В.Н. Гололобов

Экскурсия по электронике

Москва 2008

Оглавление

Предисловие

О чем эта книга, для кого эта книга?

Об электронике. И электрике, и радиотехнике, и аналоговой схемотехнике, и цифровой, и программировании, и монтаже, то есть, это книга обо всем, что связано с электричеством, для всех, кого это интересует.

Это книга для школьников, как сейчас модно говорить тинэйджеров. И для моих сверстников, которые в размышлениях о грядущем выходе на пенсию ищут, чем бы интересным занять свой досуг.

Я не хочу учить. Есть много учебников, есть много чудесных книг. Есть учебные заведения, и т.д. Я хочу рассказать о том, что занятие это очень увлекательное, способное доставить не меньше удовольствия, чем любое другое, и что это интересный выбор в качестве профессии.

Написав несколько глав книги, я (не без посторонней помощи) понял, что пытаюсь написать «правильную» книгу, начав с определений, что такое электрический ток, заряд, электрическое поле и т.д. Как будто я лучше других знаю, что такое электрический заряд! Не знаю. Знаю, что он есть, но не знаю, что это такое. Какой тогда смысл по укоренившейся со школьных времен привычке писать сочинение, пересказывая своими словами другие книги или учебники?

Я поступлю иначе. Возьму отвертку, разберу какое-нибудь устройство или прибор и постараюсь рассказать о тех электронных компонентах, которые увижу – что они такое, что из них можно сделать полезного или интересного, и как это сделать. Конечно, я буду употреблять какие-то слова (термины), которые могут быть не всем понятны. Постараюсь обозначить, что я сам вижу за этими словами. Например, я часто употребляю слово радиолюбитель, относя его не только к тем, кого интересуют радиоприемники, передатчики или любительская радиосвязь. Мне нравится это слово, в силу ли того, что оно привычно, или в силу уважения к родоначальникам этого увлечения, и я не буду выдумывать новые названия для любителей повозиться с электроникой.

Для иллюстрации своего рассказа я воспользуюсь компьютерными программами САПР (EDA), которые есть в моем распоряжении, и которые вы тоже можете использовать в своих экспериментах. Я постараюсь использовать простые понятия, ясные и простые схемы; не спешить, не пропускать деталей, очевидных для меня, но не для того, кто только начинает разбираться с предметом, и только готовится узнать о нем. И, самое главное, я очень надеюсь, что и я узнаю много нового и интересного, прежде чем закончу эту книгу.

25.07.07

Часть 1. Электроника для начинающих

Глава 1. Что на плате электронного устройства?

Резистор

Как и собирался, я с помощью отвертки, открутив винты, открыл испорченный стрелочный прибор, который нашел в своей «хламежке». Из него удалось извлечь несколько плат из стеклотекстолита с печатным монтажом; на платах много резисторов, есть конденсаторы, транзисторы, микросхемы – почти все, что мне нужно для рассказа.

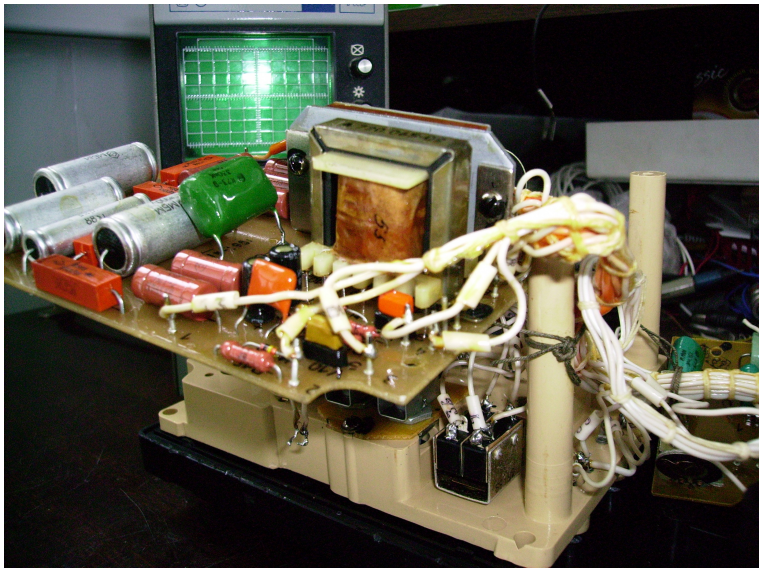


Рис. 1.1. Один из героев повествования, г-н Прибор

На самой большой плате передо мной, следовало бы посчитать, но не хочется, на первый взгляд больше всего резисторов. Резистор или сопротивление, наверное, простейший для понимания компонент любого электронного устройства...

Вот так, не успев начать рассказ, я уже сделал ошибку. Больше всего на плате не резисторов, а проводников! И, пожалуй, именно проводники самые простые компоненты. Проводники соединяют все остальные элементы устройства в сложные или простые цепи, поэтому электрическую схему я буду часто называть электрической цепью. Как правило, проводники делают из металла, вещества хорошо проводящего электрический ток. Если под током понимать *любое направленное движение электрических зарядов*, то проводники мало сопротивляются этому движению, то есть, их сопротивление обычно невелико. Свойства проводников хорошо понятны, если рассматривать атомное строение вещества, договорившись, что атомы состоят из тяжелого электрически заряженного ядра и легких электронов, субатомных частиц противоположного знака, носящихся вокруг ядра. У разных веществ заряд ядра разный, но количество электронов такое, что в целом атом электрически нейтрален. У металлов, уж так они устроены, электроны, далеко расположенные от ядра, слабо связаны с ним и могут «бродить» по металлу от атома к атому (но не могут самопроизвольно покинуть металл). Движутся они, конечно, беспорядочно, но под действием внешнего электрического поля, которое можно создать с помощью источника питания, его еще называют источником электродвижущей силы (батарейка, аккумулятор, блок питания),

их движение упорядочивается и можно говорить о протекании тока от одного полюса источника питания к другому; благодаря большому количеству носителей зарядов в металлах (электронов-бродяг), те оказываются хорошими проводниками тока. За техническое направление тока принято направление от плюса источника ЭДС (электродвижущей силы) к минусу, хотя реально в металле под действием внешнего электрического поля двигаться будут отрицательно заряженные электроны от минуса источника, поставляющего электроны в металл, к его плюсу. Если можно посчитать количество зарядов, проходящих через поперечное сечение проводника, то можно оценить силу тока – чем больше зарядов проходит через это сечение, тем больше ток. *Определяется сила тока отношением количества зарядов, прошедших за определенное время через поперечное сечение, к этому времени.* И еще о токе можно сказать, что если его величина и направление не меняется со временем, то мы имеем дело с постоянным током, иначе с переменным током. Батарейка – источник постоянного тока, а силовая сеть, куда мы подключаем пылесос или телевизор, источник переменного тока.

Создатели теории электричества, изучая эффекты, сегодня привычные нам и не всегда интересные, часами наблюдали, измеряли и искали аналогии увиденному. Так всегда удобно поступать, если что-то новое входит в ваш обиход. Приступая к освоению электричества, можно представлять себе электрический ток в виде воды в городском водопроводе: вода бежит по трубам, растекаясь по ответвлениям и к соседям справа, и слева, и сверху. Если насос, а гонит ее по трубам насос, маленький или плохой, вода уныло вытекает из открытого крана, а насос хороший и большой – хлещет и брызжет, только не догляди. И бежит вода из водохранилища, которое может обмелеть, как старая батарейка, и которое нужно наполнить, как вы заряжаете аккумулятор мобильного телефона или своего автомобиля. И пусть батарейка – и насос, и озеро в одной упаковке – устроена иначе, на первых порах вам достаточно любого представления об электрическом токе, которое поможет вам не столько понять его сущность, сколько привыкнуть к нему, чтобы в последствии не думать, что такое электрический ток, а измерять или вычислять его величину.

Все вещества по тому, как они проводят электрический ток, можно разделить на проводники, хорошо проводящие ток, изоляторы, вещества не проводящие ток, и полупроводники – «ни рыба, ни мясо», проводят ток много хуже проводников, но лучше изоляторов. Их оценивают по сопротивлению, маленькому у проводников и огромному у изоляторов. Единица сопротивления в электротехнике – Ом.

Теперь можно перейти к тому, с чего я по ошибке начал, к резисторам. Мы уже договорились, что разные вещества по-разному проводят электрический ток. Это касается и металлов. Хотя они все проводники, но одни металлы лучше проводят ток, другие хуже. Очень хорошо, например, проводят ток медь, серебро, золото. Хуже алюминий. Еще хуже сплавы металлов, как нихром, манганин, константан. Отчего зависит сопротивление проводника? От вещества, из которого он сделан, от толщины проводника и от длины проводника.

Если у вас есть мультиметр и медные провода одинаковой длины, но разного диаметра, вы можете измерить их сопротивление (провода лучше взять достаточно длинные). Если у вас найдется проводник из нихрома (от перегоревшей спирали старого нагревательного прибора) такой же длины, вы непременно заметите разницу.

Есть еще одно обстоятельство, влияющее на сопротивление проводника, это температура. При нагревании сопротивление проводника увеличивается, потому что при нагревании электроны бродяги становятся еще энергичнее в своем хаотическом движении и их труднее заставить двигаться направленно. Убедиться в том, что при нагревании сопротивление увеличивается, можно подключив к мультиметру в режиме измерения сопротивления

резистор, и нагреть вывод резистора паяльником. Если интересно, можете попробовать, только аккуратно, чтобы не испортить свой прибор. И не забывайте об этом, когда, особенно в измерительных цепях, пытаетесь получить нужную величину сопротивления, подпаивая к одному резистору другой. Обязательно дайте остыть резисторам прежде, чем оценивать результат.

Резисторы для нужд электроники изготавливают по разным технологиям и из разных материалов так, что величина их сопротивления колеблется от долей ома до десятков миллионов ом (мегаом). Сопротивление в электрической цепи может быть вредно, так получается в силовых цепях, но может быть полезно при разного рода манипуляциях с электричеством. Самое простое полезное действие электрического тока – нагрев. При протекании электрического тока по проводнику, оказывающему сопротивление, проводник нагревается. В обогревателе, где используется сопротивление, изготовленное из нихрома, такой резистор нагревается до красна. А в электрической лампочке резистор (спираль лампочки накаливания) раскаляется до бела. И в том, и другом случае сопротивление мы используем для превращения электрического тока в полезные для нас тепло и свет.

Резисторы широко используются в электронике. Есть проволочные и непроволочные резисторы, есть резисторы переменного сопротивления (потенциометры), есть терморезисторы и фоторезисторы. А такое свойство резисторов, как изменение сопротивления при механическом воздействии, находит применение в тензодатчиках.

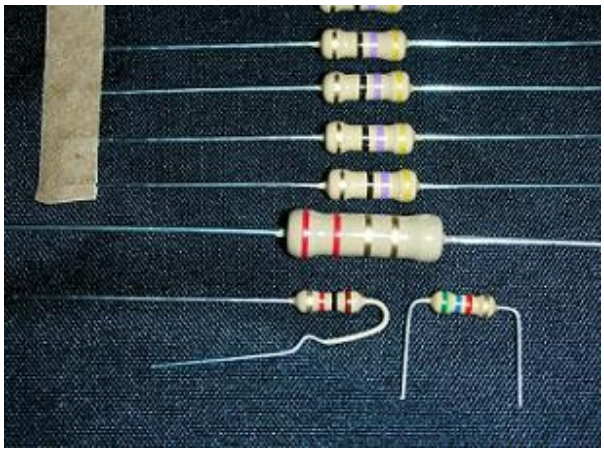


Рис. 1.2. Резисторы

Я сейчас отпаяю несколько резисторов с платы прибора и перенесу их на макетную плату, чтобы рассказать о нескольких простых, но очень полезных правилах, которые называют законами для электрических цепей. Макетная плата у меня покупная.

Рис. 1.3. Еще один из персонажей, г-н Макет

Такие платы применяют при создании прототипов электрических устройств. На макетной плате можно спаять устройство, проверить, наладить, а когда оно полностью готово, можно перейти к изготовлению образца. Очень часто макетная плата – это набор контактных площадок из меди с отверстиями для выводов компонентов электрической схемы. Плату можно изготовить самостоятельно из фольгированного листового материала, а при его отсутствии из любого жесткого листового изолятора, желательно термостойкого, чтобы плата не плавилась при пайке. При механической обработке стеклотекстолита - сверлении, распиливании, обработке напильником – следует соблюдать осторожность, поскольку пыль стекловолокна травмирует дыхательные пути. Можно использовать подходящий кусок фанеры. Контактные площадки для припаивания компонентов можно сделать из кусочков медного электрического провода без изоляции, продев из в два просверленных рядом отверстия и загнув с обратной стороны. Можно обойтись и без контактных площадок, просверлив отверстия, в которые продеваются выводы элементов схемы, а к выводам с обратной стороны припаиваются проводники. Если макетную плату снабдить стойками в 1-1.5 см, то работать с ней будет еще удобнее.

Для пайки используется паяльник (еще один пример полезного использования сопротивления), у меня паяльник на 25 ватт 220 вольт, изготовленный в Подмосковье. Сегодня можно купить хорошую паяльную станцию – паяльник с регулировкой температуры нагрева, с множеством насадок для пайки и удобной подставкой. Ручка моего паяльника сделана так, что его можно положить на ровную поверхность без подставки, но я привык использовать подставку, которую сегодня, думаю, тоже можно купить в магазине. Кроме паяльника для пайки нужен припой, лучше ПОС-61 в виде тонкого прутка, и паяльный флюс, например, канифоль, хотя я использую жидкий флюс ЛТИ-120, который держу в пузырьке из-под лака для ногтей с кисточкой, достаточно удобно. Флюс растекается по месту пайки, помогая припою лучше соединить детали, ведь пайка – это один из способов соединения деталей, кстати, не единственный, хотя в электронных изделиях наиболее широко применяемый. Кроме пайки можно использовать скрутку, одно время монтаж с помощью скрутки был очень популярен. При работе с паяльником тоже следует соблюдать осторожность и не только с тем, чтобы не обжечься. Припой, испаряясь, не принесет пользы при вдыхании. Не следует паяльник постоянно держать включенным, лучше лишний раз подождать, когда он нагреется, или собрать одну из простых схем, о которых мы поговорим позже.

Пока я все это рассказывал, я успел включить паяльник и выпаять резисторы из платы.

До того, как продолжить рассказ об электрических цепях, я хочу заметить, что описать электрическую цепь и все, что с ней связано, можно только словами, не прибегая ни к чему другому, но получается длинно, и далеко не всегда понятно. Поэтому для изображения электрических схем используют графическое представление – лучше один раз увидеть, чем сто раз услышать. Каждый компонент рисуют в виде небольшой простой картинки, а провода, соединяющие элементы схемы, изображают в виде линий. Собственно, такое графическое представление и называют схемой устройства. Простые схемы можно нарисовать так, как они будут нарисованы ниже, более сложные схемы рисуются на многих листах бумаги, а для объяснения их работы используют еще один графический вид – функциональные схемы: все устройство можно, и должно, разбить на части, функциональные узлы, как, скажем, выпрямитель, усилитель, преобразователь и т.д., которые изображаются в виде прямоугольников, связанных линиями или стрелками. К таким сложным (очень полезно, если и к простым) схемам прилагают их описание, которое может занимать несколько томов.

Графическое изображение элементов электрической схемы в разных странах, в разные годы выглядело несколько по-разному. Так для изображения батарейки используют изображение из двух черточек, одна из которых длиннее другой, с перпендикулярными к ним

выводами, аккумулятор изображали в виде нескольких таких батареек. Но, порой, в схеме не делается различия между этими двумя источниками ЭДС. В последнее время, особенно в программах, эти источники питания объединяют с другими источниками тока в общий класс источников (source) и изображают в виде кружка со значками плюс и минус. Подобные отличия есть в изображении резисторов в виде прямоугольников, обозначенных латинской буквой R с порядковым номером однотипных элементов, и в виде ломаной линии. Есть отличия в графическом изображении других элементов, о которых я постараюсь рассказать по мере их появления в книге. Обычно это не вызывает больших затруднений, но если вы будете рисовать свои схемы, лучше выбрать один стиль.

Для черчения схем и пояснения их работы я буду пользоваться компьютером, точнее программой Qucs и демонстрационной версией программы PSIM, которую скачал с сайта производителя. Последняя предназначена для разработки схем силовой электроники. Многие программы имеют свою специализацию. Если эти программы перестанут мне помогать в рассказе, я использую другие. Программа PSIM предназначена для работы на платформе Windows, но работает у меня в Linux, при этом я использую эмулятор Wine. Как это все делается я расскажу позже, а сейчас хочу заметить, что изображение резистора в программе есть только в виде ломаной линии. Второе изображение мне пришлось пририсовать в графическом редакторе.

Схема (схемы) на рисунке ниже имеют один графический элемент, о котором я еще не говорил. Это земля или общий схемный провод. Посмотрите на изображение схемы (два изображения), а потом я постараюсь ответить на вопрос об общем проводе схемы.

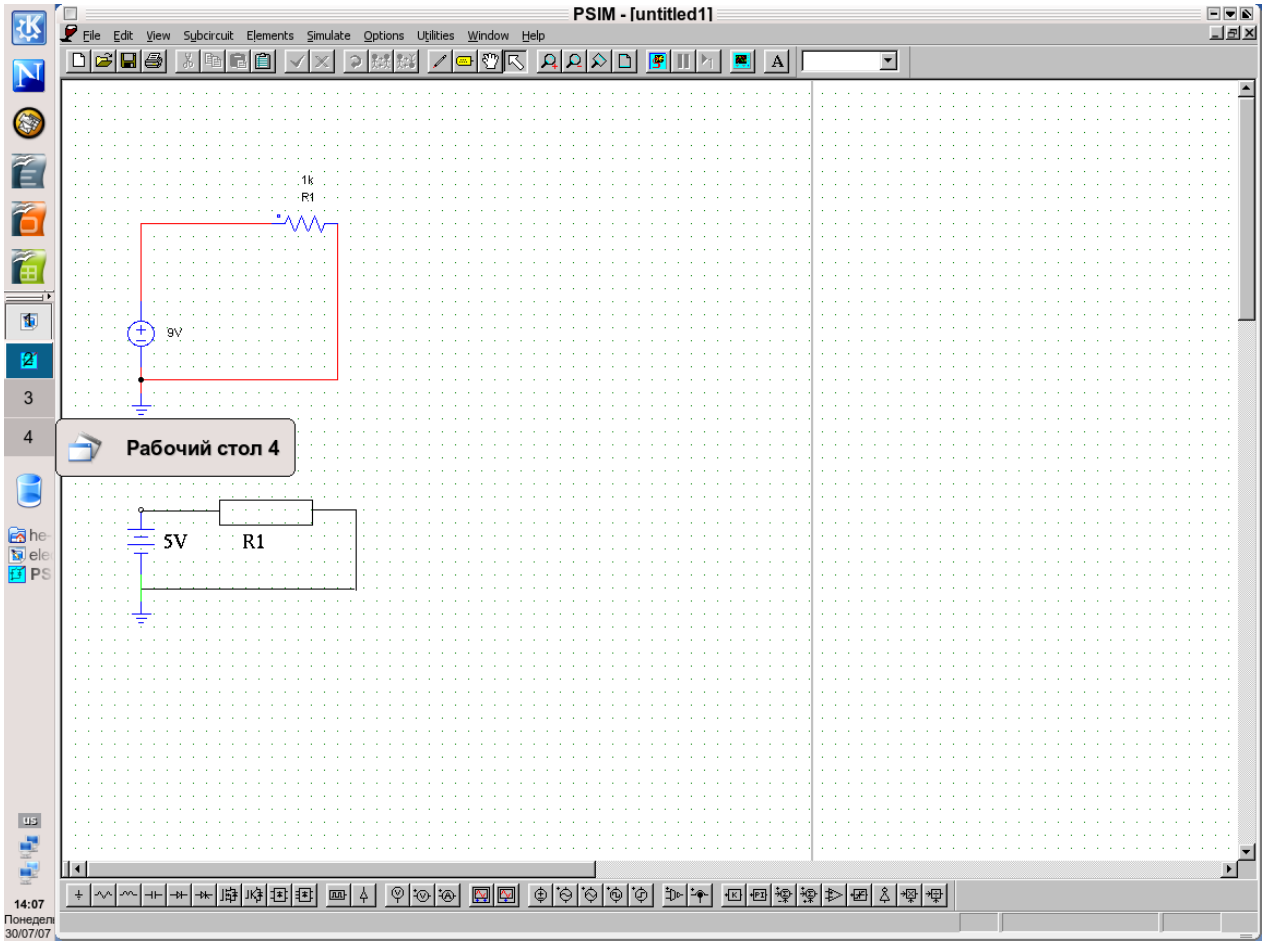


Рис. 1.4. Графическое изображение проводника, сопротивления и батарейки

Зачем же нужно вводить обозначение для общего провода схемы, который часто называют земляным проводом, или, просто, землей?

Забегая вперед, скажу, что в схеме удобно измерять напряжения относительно одной точки схемы, или одного общего проводника, удобно смотреть сигналы относительно этого проводника. Кроме того, если этот общий провод схемы соединить с землей – специально устраиваемым заземлением, имеющим хороший контакт именно с землей (грунтом, почвой) – схема меньше подвержена вредным влияниям внешних электрических полей, поэтому общий провод схемы часто называют «землей».

Мы пока познакомились только с тремя компонентами электронного устройства: батарейка, проводник, резистор. Можно ли с их помощью построить что-либо интересное?

Можно. Во-первых, можно провести несколько экспериментов для знакомства с тремя законами: закон Ома и два закона Кирхгофа. Все законы мы рассмотрим в их простейшем виде, а более сложный вид при необходимости можно найти в учебной литературе. Этих трех законов электротехники, я надеюсь, мне хватит на протяжении всей книги, и не появится необходимости в других. Как схемы удобнее изображать в графическом виде, так законы удобнее записывать в виде математических соотношений. Для этого используется латинская буква R для сопротивления, латинская буква I для тока, и латинские U или V для напряжения.

Закон Ома (для участка цепи) звучит так:

*Падение напряжения (или напряжение) на участке цепи равно произведению тока на сопротивление: U=I*R.*

Не знаю, как вас, меня удивляет, вызывает чувство уважения и восторга проницательность и ум человека, который смог подметить и определить столь простую и полезную связь между этими тремя величинами. Именно его именем названа единица сопротивления.

Соотношение между этими тремя величинами с точки зрения математики можно записать в трех видах: U=R*I, I=U/R и R=U/I. Все три вида записи можно применять на практике, но не следует забывать, что, например, последнее соотношение позволяет вычислить сопротивление, однако сопротивление проводника или резистора определяется свойствами материала и геометрией проводника, а ток, вычисляемый по напряжению и сопротивлению, как упорядоченное движение носителей заряда, вызывается источником электродвижущей силы. *Напряжение* в этом смысле можно рассматривать, как некоторое напряжение в отношениях между током, протекающим по проводнику, и материалом проводника, оказывающим сопротивление протеканию тока.

Для экспериментальной проверки соотношения между напряжением, током и сопротивлением можно собрать схему, аналогичную изображенной на рисунке 1.3, в которую следует добавить два измерительных прибора: вольтметр и амперметр (или использовать мультиметр, произведя два измерения). Вольтметр – это прибор используемый для измерения напряжения. Бывают вольтметры для измерения постоянного напряжения, которым воспользуемся мы, а бывают вольтметры переменного напряжения, о которых речь пойдет дальше. Амперметр – это прибор для измерения тока, который также бывает для измерений в цепях постоянного и переменного тока.

Эти приборы могут быть сконструированы по-разному. Я говорил о мультиметре. В его основе лежит работа специализированной и достаточно сложной микросхемы, называемой аналого-цифровой преобразователь (АЦП). Многие мультиметры могут кроме этой микросхемы не иметь других микросхем, лишь вспомогательные резисторы, переключатель и дисплей, отображающий цифры. Возможно, к концу книги мы рассмотрим работу такого прибора, а сейчас, все-таки опять забегая вперед, я немного расскажу о стрелочном

измерителе тока. Достаточно часто в стрелочных измерительных приборах используют следующую конструкцию: рамку с намотанным на нее проводом помещают в магнитное поле, создаваемое постоянными магнитами, и крепят с помощью спиральных пружинок на оси так, чтобы стрелка, прикрепленная к этой рамке, находилась в нулевом положении, отмеченном на шкале. Когда по рамке протекает постоянный ток, то магнитном поле действует на рамку, заставляя ее поворачиваться. Это происходит до тех пор, пока сила, вызываемая взаимодействием тока с магнитным полем, не уравновесит возвращающую силу пружинок. Стрелка останавливается в положении, которое пропорционально протекающему току, значение которого считывается со шкалы прибора. Если у вас есть тестер со стрелочной измерительной головкой, то она может быть устроена именно так или похожим образом: часто оси вращения и пружинки возврата стрелки в ноль заменяют «растяжками». То есть, к рамке с проводом прикрепляют очень тонкую полоску из пружинящего материала. Растянутая на двух таких полосках рамка под их действием занимает «нулевое» положение, а при протекании тока поворачивается, отчего полоски упруго скручиваются, образуя силу, возвращающую рамку со стрелкой в первоначальное положение, когда ток перестает протекать по рамке. Но о механизме взаимодействия провода с током и магнитного поля я ничего вам не говорил, поэтому будем считать, что ничего не говорил и о стрелочных приборах. Хотя, пожалуй, добавлю к тому, о чем не говорил, что измеритель тока легко превратить в измеритель напряжения, добавив к нему резистор.

Итак, посмотрим, на закон Ома, который я постараюсь проиллюстрировать в программе PSIM. Кроме того, я спаяю схему на макетной плате, используя один из резисторов, которые выпаял из платы, вдруг я что-то напутал и ввел вас в заблуждение. Доверяй, но проверяй.

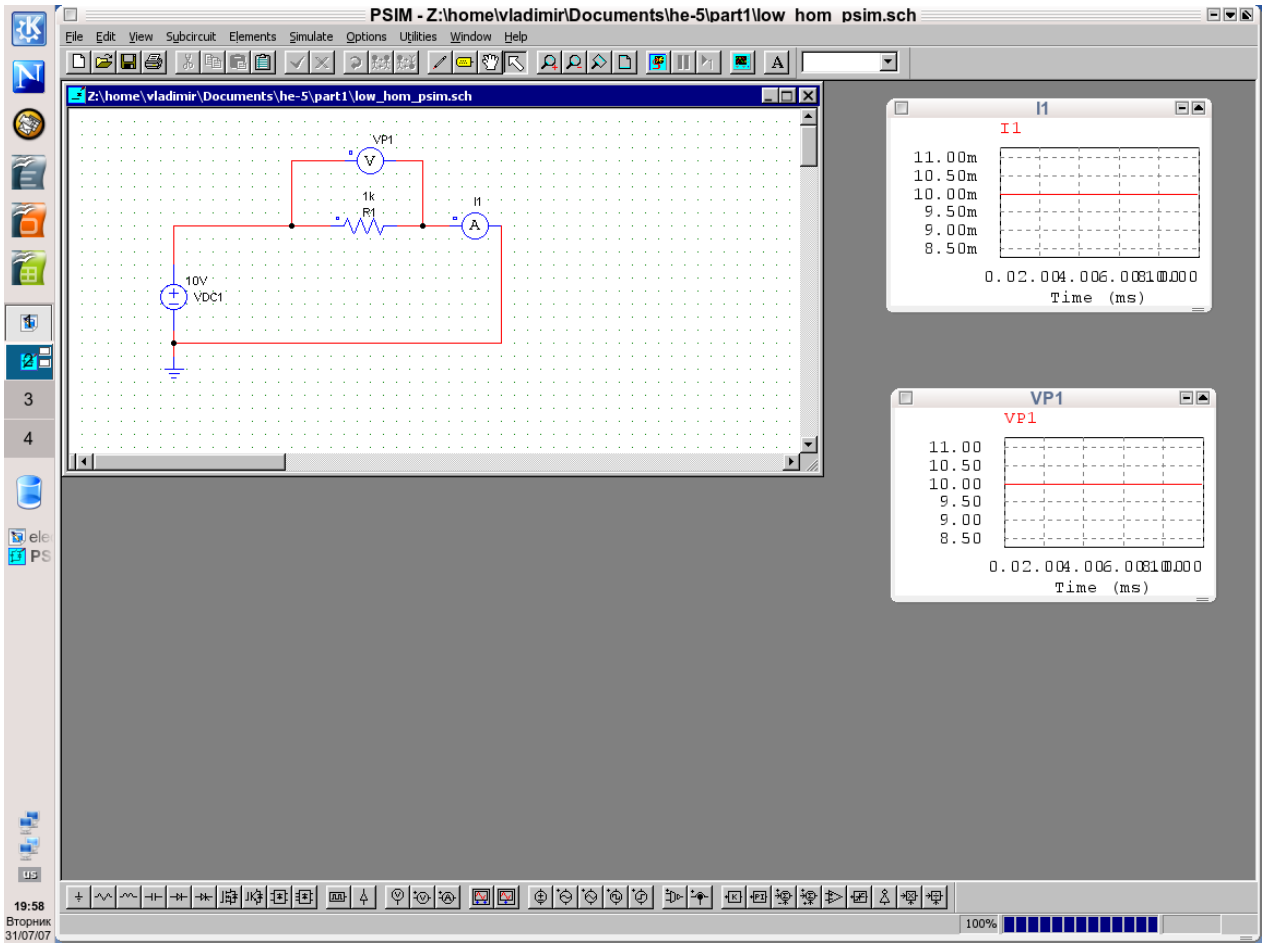


Рис. 1.5. Эксперимент, иллюстрирующий закон Ома в программе PSIM

Как видно из рисунка, к источнику питания 10V (10 Вольт, на схеме обозначен VDC1) подключен резистор 1 кОм (1000 Ом), параллельно которому включен вольтметр VP1, и последовательно с которым включен амперметр I1. График измерения тока показывает, что ток равен 10.00m (10 миллиампер) или 0.01 А. Если умножить сопротивление на ток, то есть, 1000 Ом умножить на 0.01 А, то получится падение напряжения в 10 вольт, которые и показывает второй график, отображающий показания вольтметра. Для получения правильных значений при расчетах следует пользоваться основными единицами, в данном случае ампер, ом и вольт.

А вот, что получилось с макетом. На батарейке, которую я использую, написано 9V, на резисторе 1к, ток должен получиться 9мА (9 миллиампер, 0.009А). Измеренный ток 8мА.

В чем ошибка? Во-первых, я не измерил ЭДС (напряжение на батарейке), во-вторых, не измерил сопротивление. Реальное сопротивление резистора, если его измерить, не 1 кОм (килоом, 1000 Ом), а 910 Ом. А ЭДС батарейки после подключения резистора оказывается равной 7.31 вольт. Отсюда и расхождение, скорее не в теории и практике, а в моем представлении о том, что я делаю, с тем, что делаю в действительности. Доверяй, но проверяй!

От закона Ома можно плавно перейти ко второму закону Кирхгофа. Почему ко второму, а не к первому? Мне так удобнее.

Закон Кирхгофа в упрощенном виде:

ЭДС в замкнутом контуре равна сумме падений напряжений.

Действительно, на рисунке выше ЭДС (источника питания VDC1) 10 В, а напряжение на резисторе тоже 10 В. Можно изменить схему, включив последовательно два резистора, например, по 500 Ом, измерить на них напряжения и убедиться, что на каждом из них будет падение напряжения 5 В, а сумма этих напряжений получится 10 В. Попробуйте включить несколько резисторов разной величины, несколько вольтметров и амперметров, и если вы при этом получите, что сумма напряжений на всех резисторах не будет равна ЭДС, обязательно сообщите мне об этом, я перепишу всю главу наново.

Здесь уместно добавить, что более верно этот закон звучал бы так: *алгебраическая сумма всех ЭДС в замкнутом контуре равна алгебраической сумме падений напряжений.*

Первоначально, написав эту главу, я не стал добавлять пояснений, но, занимаясь правкой, ошибок я делаю много, и, подчас, они носят принципиальный характер, занимаясь правкой я решил дополнить главу еще одним примером, правда, использовал программу Qucs, но надеюсь, это не помешает понять происходящее.

Если источников ЭДС несколько, то в одном контуре могут протекать токи разных направлений, создавая падения напряжения разных знаков, а источники питания могут быть включены встречно друг другу. В следующем примере источник ЭДС V1 приводит к появлению тока в 10 мА, что определяется суммой сопротивлений R1 и R2. Но второй источник V2 создает встречный ток в 5 мА. В результате по цепи протекает ток равный разности этих токов 10 — 5 = 5 (мА), создающий два падения напряжения на резисторах по 2.5 В. Алгебраическая сумма всех ЭДС, а источники питания включены встречно, будет равна 10 + (- 5) = 5 В. В данном случае цепь очень простая, бывают и более сложные, но не думаю, что вам часто придется иметь с ними дело.

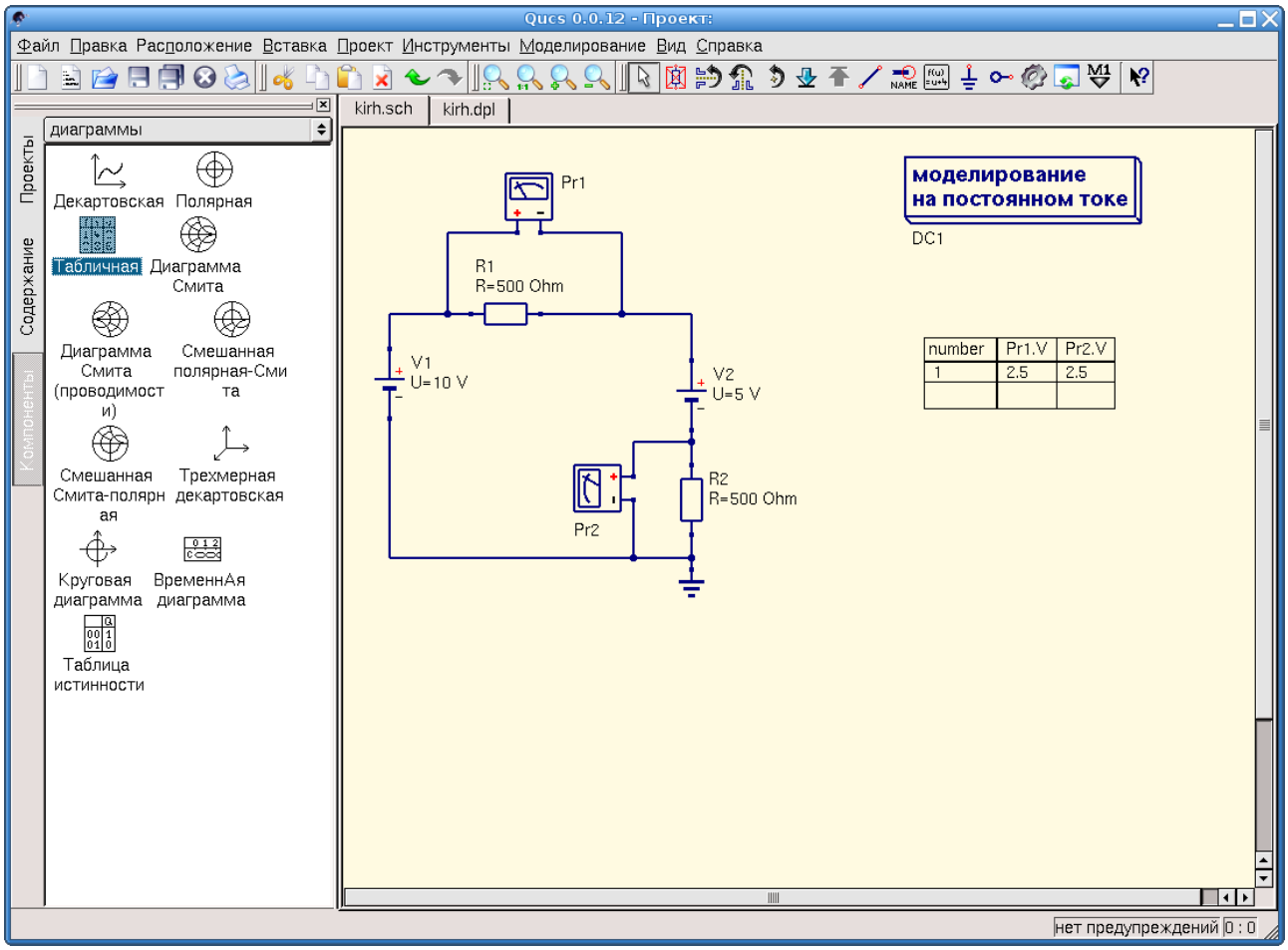


Рис. 1.6. Два источника ЭДС в одной цепи

Хотя для разговора о первом законе Кирхгофа следовало бы нарисовать другую схему (другую цепь), я использую схему рисунка 1.4. Выше я говорил о том, что часто вольтметр – это тоже измеритель тока, то есть, ток протекает не только через резистор R1, но и через вольтметр. Таким образом, в точке соединения вольтметра и резистора ток разветвляется, одна его часть протекает по резистору, другая по вольтметру, а затем обе эти части соединяются и протекают через амперметр.

Сумма токов, вытекающих из узла электрической цепи при ее ветвлении, равна току, втекающему в этот узел.

Так в упрощенном виде звучит первый закон Кирхгофа. Соответственно, если в узел втекают токи, а из него вытекает ток, то он будет равен сумме втекающих токов. В реальной схеме это можно было бы проверить, включив еще два амперметра в ветвях схемы, одна из которых относится к резистору, а вторая к вольтметру.

На практике, проводя измерения, всегда следует помнить, что вольтметр имеет некоторое внутреннее сопротивление, которое может повлиять на результаты измерения. Так достаточно хороший вольтметр может иметь внутреннее сопротивление в 100 кОм. Много это или мало? Это не много и не мало, но ровно столько, сколько есть. Как это может повлиять на ваше понимание происходящего? Положим, у вас есть вольтметр и амперметр, и вы хотите провести измерения. Вольтметр имеет внутреннее сопротивление равное 100 кОм. Вы хотите определить, используя закон Ома, величину сопротивления, маркировка которого стерлась от времени (но это был резистор в 100 кОм, о чем ни вы, ни я не знаем). В схеме на

рисунке 1.4 я заменю вольтметр (напряжение на резисторе R1 при измерении будет равно 10 В) сопротивлением в 100 кОм, и такое же сопротивление будет иметь резистор R1. По результату измерения тока в цепи определим величину сопротивления.

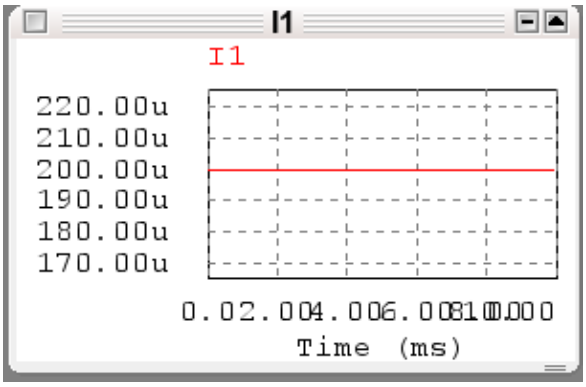


Рис. 1.7. Ток в цепи при определении величины сопротивления по закону Ома

Как видно из рисунка, ток равен 200 мкА. При падении напряжения в 10 В, величина сопротивления определится делением этого напряжения на ток и будет равна 50 кОм. Так мы измерили неизвестное сопротивление без учета сопротивления вольтметра. И ошиблись в два раза. А это уже не мало. В данном конкретном случае нам помогло бы проведение двух измерений с помощью тестера или мультиметра. Первый раз мы могли бы измерить напряжение, а второй раз ток. Думаю, мы получили бы правильный результат. Но... но только в этом случае. Если немного усложнить цепь, скажем, включив последовательно с измеряемым сопротивлением еще одно такой же величины, то и методика двух измерений может дать неверный результат.

Имея опыт работы с измерительными приборами, достаточно быстро соотносишь полученные результаты с параметрами прибора. Иногда это происходит еще до измерения. Но, даже имея опыт работы, и начиная пользоваться прибором, с которым раньше не работал, можно ошибиться. Чтобы избежать подобных ошибок, следует разумным образом организовывать свою самостоятельную работу, и не спешить с выводами, обдумывая полученные результаты. Если вы повторяете готовую схему, имеющую подробное описание, то обратите внимание на рекомендации по наладке схемы, зачастую там есть указания на то, с помощью каких приборов проводились измерения.

Несколько советов по организации своих разработок я постараюсь дать в конце книги. Но главный совет — не спешить. Увлекаясь, в надежде получить быстрый результат забываешь проверить даже состояние источника питания, а когда результаты измерений плохо отвечают ожиданиям, начинаешь что-то менять в схеме. Это, обычно, плоды не столько небрежности, сколько поспешности.

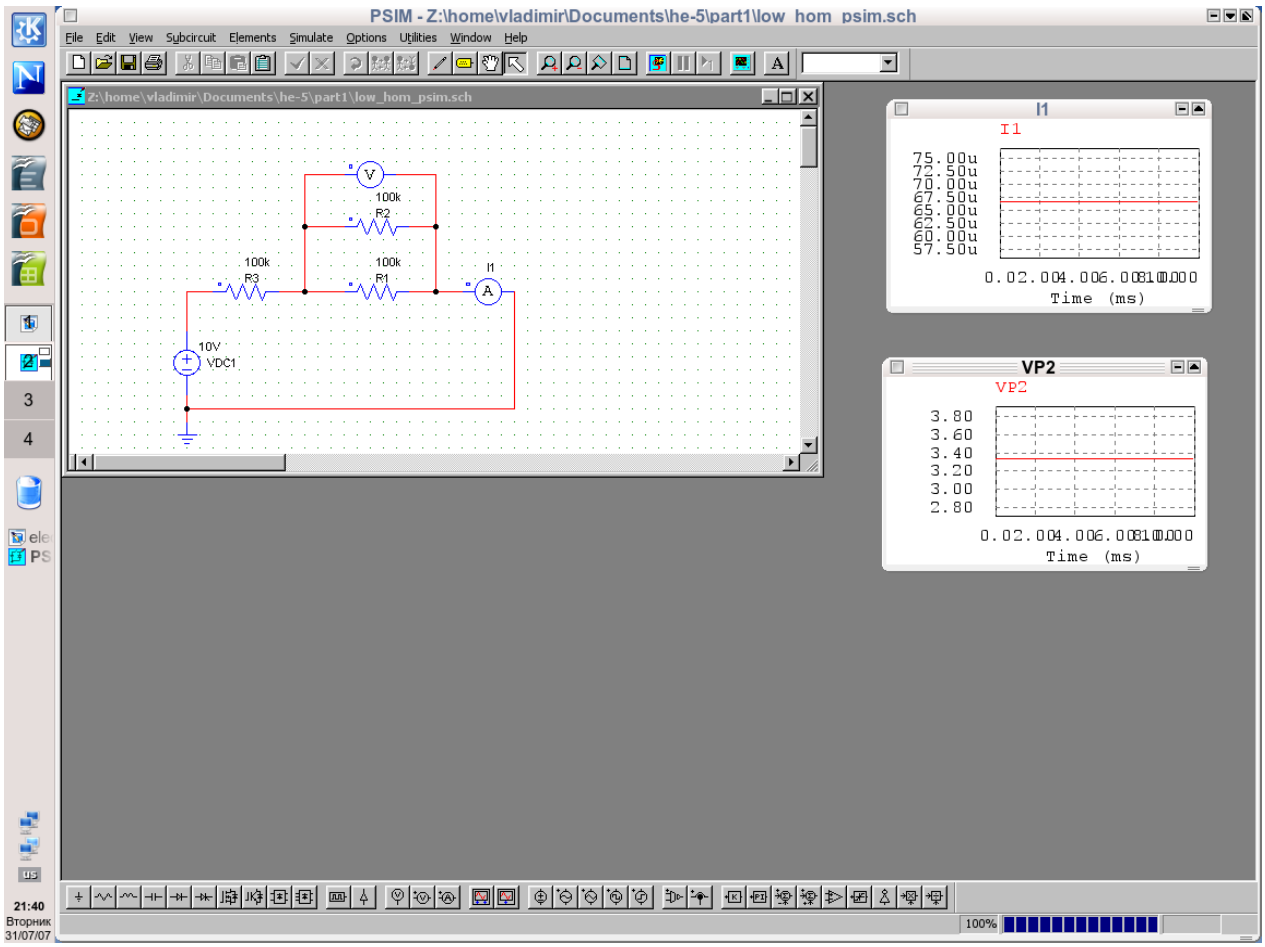


Рис. 1.8. Измерение величины сопротивления без учета сопротивления вольтметра

По графикам трудно определить точные значения, но приблизительно это будет напряжение 3.35 В и ток 66 мкА. В результате деления получается значение около 51 кОм. Такое измерение в одних случаях может только затруднить понимание реального положения дел, но в других может привести к неприятностям, которых лучше избежать, если помнить, что каждый прибор имеет определенные параметры, и их следует учитывать при пользовании прибором.

Можно объединить все три закона в одно правило, если кому-то удобно запомнить его.

Сумма падений напряжений в электрической цепи, равных произведению сопротивлений на ток, протекающий через них, равна сумме ЭДС в этой цепи, а сумма токов, вытекающих из узла по всем ветвям цепи, равна втекающему в узел току.

И следует помнить, что на всех компонентах электронных устройств проставляют номинальное значение, которое, в зависимости от изготовления, может отличаться от реального на 5-20%, хотя есть и компоненты, изготовленные с большей точностью, 0.1-1%, но они применяются реже и стоят дороже.

Напомню, что за техническое направление протекания постоянного тока принято направление от плюса к минусу. Цифровые мультиметры, как правило, показывают знак измеряемого напряжения и тока, и они не так чувствительны к подключению с неверной полярностью, как стрелочные приборы. Последние зашкаливают «в обратную сторону» и могут от этого пострадать. Всегда следует проверять полярность подключения стрелочных приборов. Минус у тестеров при измерении напряжения и тока зачастую помечают значком в

виде звездочки. В простых цепях, изображенных на рисунках, все достаточно очевидно, но в сложных цепях, когда есть много сопротивлений, включенных сложным образом, когда есть несколько источников питания, тогда определение напряжений и токов усложняется. Конечно, существуют математические методы расчета подобных цепей, описанные в учебниках, однако я сомневаюсь в их популярности в любительских кругах, поскольку удобнее измерить интересующую величину, чем рассчитать ее, но очень важно иметь ясное представление об основных процессах в электрической цепи. Подключая прибор, следует начинать измерение с безопасного для прибора предела измерения – самого большого напряжения или тока, позже его можно изменить.

Законы Ома и Кирхгофа помогают легче понять, что происходит при последовательном и параллельном включении резисторов. При последовательном включении резисторов ясно, общее сопротивление увеличивается. Но при параллельном, возьмем для простоты эксперимента два резистора, по закону Кирхгофа падение напряжения на общем сопротивлении равно ЭДС, значит каждый из резисторов ведет себя так, как если бы второго не было, то есть, ток через него, по закону Ома, такой же что и при его одиночном включении. А по другому закону Кирхгофа ток в точке ветвления должен быть равен сумме токов, растекающихся по двум параллельно включенным резисторам, то есть ток увеличивается так, как если бы вместо двух резисторов был включен один, меньшей величины. При параллельном включении резисторов удобно складывать величины обратные их сопротивлению (проводимости), а результирующее сопротивление будет обратной величиной к полученному результату.

Пример с резисторами может создать ложное впечатление, что законы Ома и Кирхгофа относятся только к ним. Отнюдь. У транзистора типа p-n-p, о самом транзисторе речь пойдет ниже, ток, втекающий в эмиттер, разветвляется на токи базовой и коллекторной цепи. Сопротивление в цепи базы, как правило, значительно больше, чем сопротивление в цепи коллектора, и ток в коллекторной цепи значительно больше базового. Токи базы и коллектора связаны определенной зависимостью, а в цепи коллектора может не быть резистора, но сумма тока базы и тока коллектора всегда будет равна току эмиттера. То же можно сказать о напряжениях. Если транзистор включен, как это показано на рисунке 1.28 (много дальше), то можно сказать, что ЭДС равна сумме падения напряжения на переходе эмиттер-база и падению напряжения на резисторе в цепи базы, а для коллекторной цепи – ЭДС равна сумме падения напряжения на транзисторе (эмиттер-коллектор) и на сопротивлении нагрузки в цепи коллектора.

В любой мало-мальски интересной схеме можно встретить элементы, о которых выше не было ни слова, и позже мы поговорим об этом, но я постараюсь показать, что достаточно сложные, если в них вникнуть, процессы можно в любительской практике свести к таким понятиям, как сопротивление, напряжение и ток.

При работе с электрическими цепями важно учитывать мощность, определяемую произведением напряжения на ток. Если мы измеряем ток, уходящий от источника питания, и умножаем его на ЭДС (напряжение) источника питания, то мы можем говорить о мощности, потребляемой схемой. Если мы измеряем напряжение на сопротивлении и ток, протекающий через него, то можем говорить о мощности, потребляемой этим сопротивлением и обычно выделяющейся на нем в виде тепла. Естественно, что резистор при этом нагревается, и если неправильно выбрать такой его параметр, как допустимая мощность рассеяния, то сопротивление перегреется и может сгореть. Обычно на схеме указывается мощность любого сопротивления, или она указывается в спецификации – перечне всех элементов схемы с их параметрами. По параметру допустимой мощности сопротивления делятся на ряд значений, из которых наиболее часто употребляемые в схемах – это резисторы 0.25 Вт и 0.5 Вт. Мощность, рассеиваемая резистором, не должна превышать этой величины. Вместе с тем,

следует помнить, что нагрев сопротивления приводит к изменению его величины. Чем ближе допустимая мощность рассеивания резистора к мощности, выделяемой на нем, тем сильнее он будет разогреваться. Если вам важно сохранить величину сопротивления, то следует выбрать более мощное сопротивление. В измерительных приборах, равно как любых цепях, относящихся к измерению, там где значение сопротивления очень важно, кроме сопротивлений с более высокой допустимой мощностью рассеяния применяют специальные сопротивления, мало меняющие свое значение при нагреве.

Есть еще несколько интересных, и как мне кажется, важных моментов, относящихся к сопротивлению. Многие неисправности в электронных устройствах связаны с проблемами источников питания. Батарейки, например, со временем «садятся». Простейший способ проверить батарейку – измерить ток, который она может отдавать. Для этого достаточно включить мультиметр (или тестер) в режим измерения максимального для конкретного прибора постоянного тока, мой мультиметр измеряет токи до 10 А, и подключить амперметр к батарейке. Свежая батарейка, в зависимости от типа, покажет ток в несколько ампер, тогда как разряженная сможет дать только десятки миллиампер. Зная закон Ома, мы можем определить ожидаемый ток до проведения измерения. Положим батарейка имеет напряжение 1.5 вольта. Амперметр имеет сопротивление 0.1 Ом. Тогда мы должны получить ток 15 А. Проверим это утверждение сначала в программе PSIM.

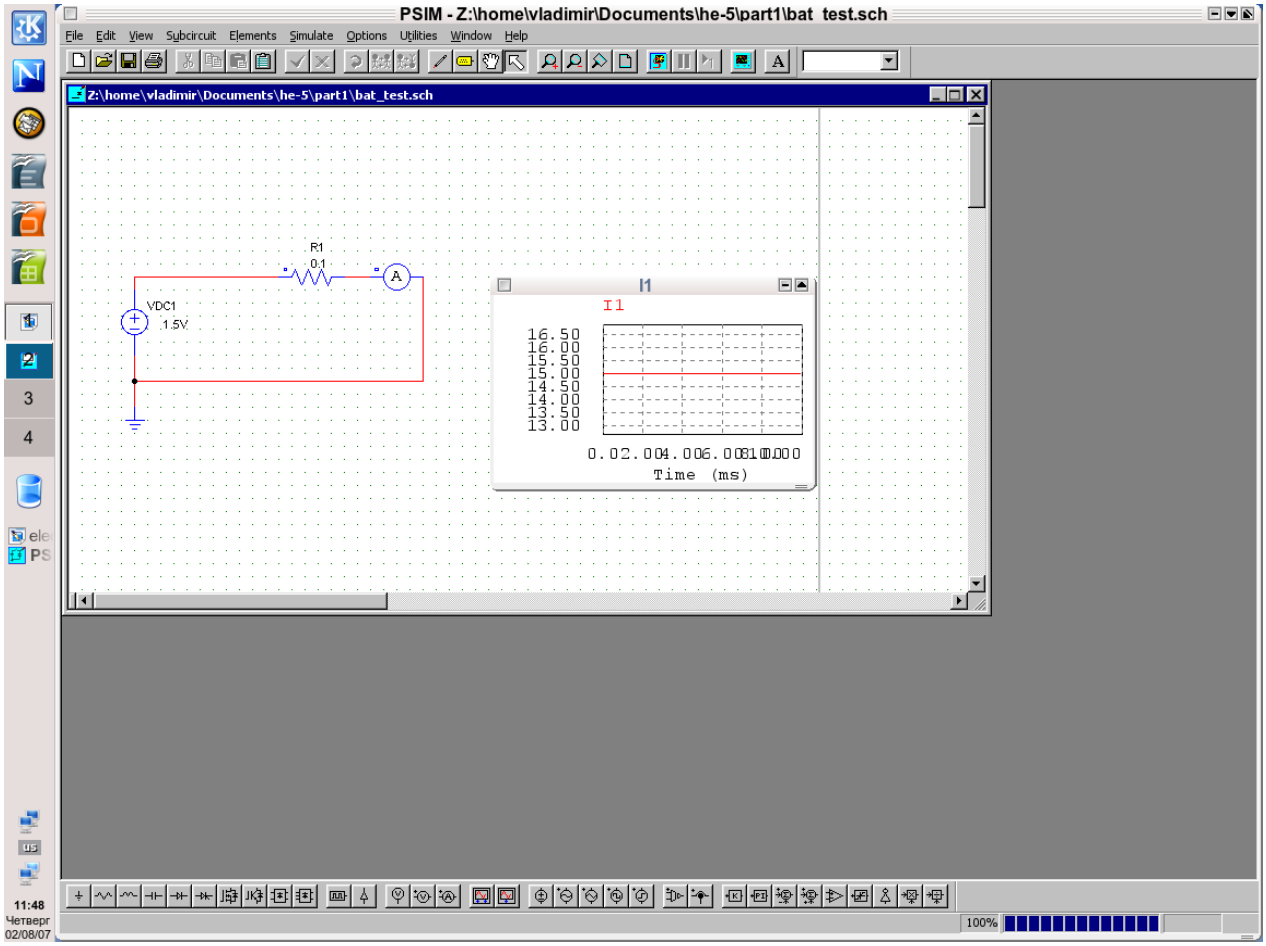


Рис. 1.9. Проверка батарейки на пригодность

В схеме на рисунке сопротивление R1 – это сопротивление реального измерительного прибора. Как мы и рассчитали, при измерении протекает ток в 15 А. Но едва ли вы увидите такой ток, если проведете реальное измерение. И дело не в плохой батарейке. Дело в том, что

реальная батарейка, как любой источник питания, это не идеальный источник ЭДС. Любой реальный источник питания имеет, как и реальный измерительный прибор, внутреннее сопротивление. Небольшое, зависящее от типа источника, но это внутреннее сопротивление есть. Оно-то и уменьшает ток через батарейку. То есть, в схеме на рисунке 1.9 следовало бы нарисовать последовательно с источником питания еще одно сопротивление, а при расчете по закону Ома использовать сумму двух резисторов. Однако внутреннее сопротивление батарейки – это параметр, который не увидишь на этикетке. Плохо ли, что батарейка имеет внутреннее сопротивление, да еще и меняющееся со временем? С одной стороны плохо. А с другой... современные мобильные телефоны имеют аккумуляторы с очень низким внутренним сопротивлением. Они могут отдавать большой ток. По этой причине многие из них приходится снабжать специальным устройством, ограничивающим этот ток. И проблема не в том, что если этого не сделать, и вы коснетесь выводов аккумулятора, то вас «тряхнет» током. Нет. Проблема в том, что если этого не сделать, и вы решите почистить выводы аккумулятора безопасной бритвой, бывает такое, то при замыкании выводов бритвой большой ток через нее может расплавить бритву (вспомните про нагревание резистора), а расплавленный металл вызвать серьезные ожоги. При этом процесс происходит настолько быстро, что расплавленный металл разбрызгивается в разные стороны. Что хорошо в одних случаях, плохо в других. На практике, если внутреннее сопротивление источника на порядок меньше сопротивления цепи, то им можно пренебречь.

Коль скоро я упомянул внутреннее сопротивление батареек, хочу немного рассказать о граблях, на которые сам временами наступаю. Схему на рисунке ниже я выберу самую простую.

Вы все знаете, что часто в пультах управления, плеерах и т.п. ставят две батарейки, включая их последовательно. Иногда две батарейки включают параллельно, соединяя их положительные и отрицательные выводы. В этом случае они могут дать больший ток, или служат вдвое дольше. Что будет, если при параллельном включении соединить их разнополярно: положительный вывод одной батарейки соединить с отрицательным другой, а отрицательный с положительным? Согласно закону Кирхгофа сумма ЭДС должна быть равна сумме падений напряжений. Но сумма ЭДС (алгебраическая, то есть, с учетом знака) равна нулю. Значит сумма падений напряжений (напряжение) тоже будет равно нулю. А ток?

Если я отвечаю не подумав, то скажу – нет напряжения, нет и тока. Но это не так. Нарисуем эту схему в программе PSIM, добавив внутренние сопротивления батареек, амперметр для измерения тока и вольтметр для измерения напряжения. Параметры измерительных приборов учитывать в данном случае не будем, чтобы не усложнять общую картину.

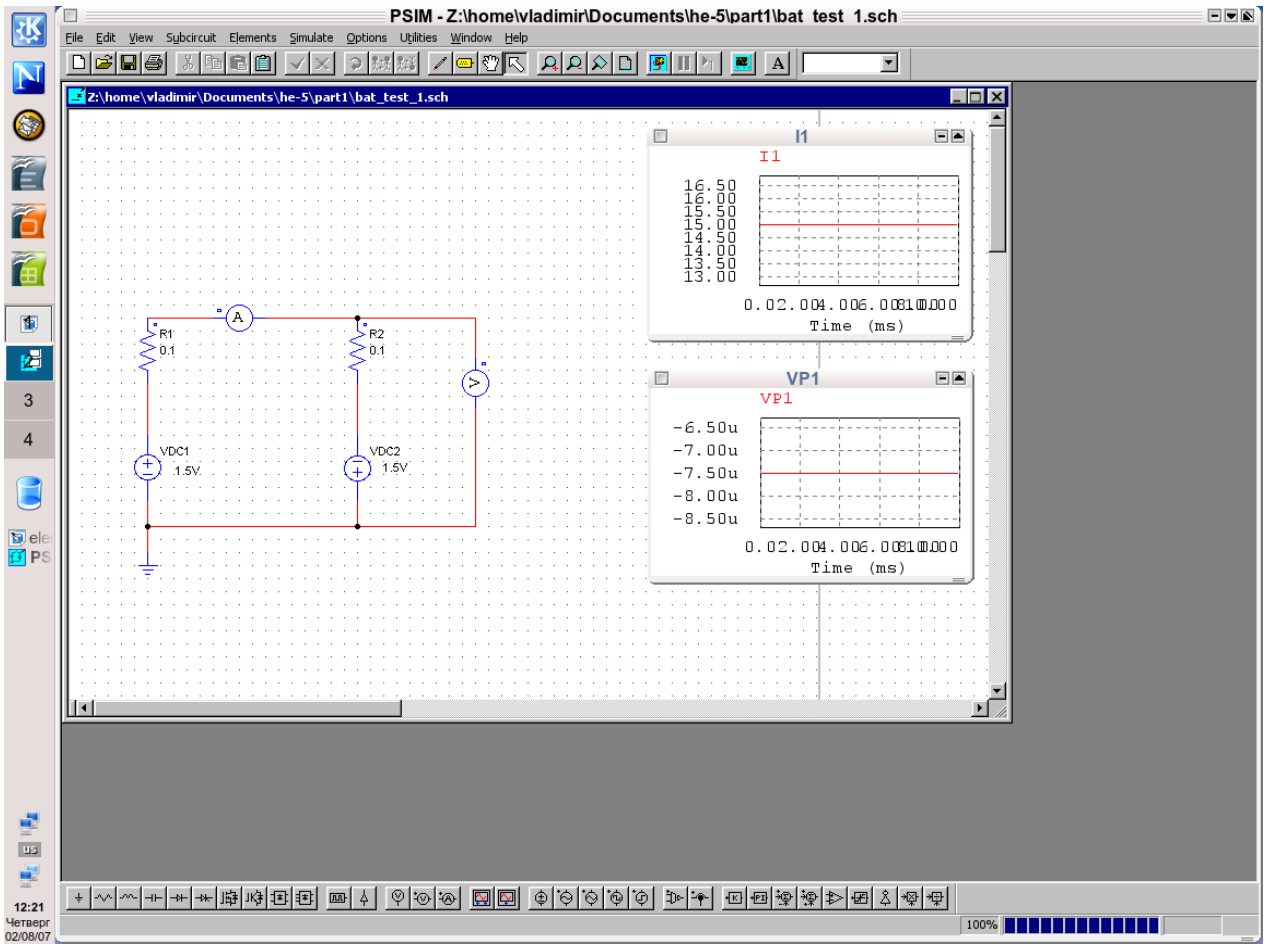


Рис. 1.10. Встречное включение батареек

Как видно на рисунке напряжение, измеряемое с помощью VP1, близко к нулю и равно нескольким микровольтам, тогда как ток, измеряемый I1, равен 15 А. Действительно, если убрать общий провод (землю) и повернуть схему, то можно рассматривать два источника питания, как включенные последовательно и нагруженные на цепь, состоящую из резисторов (внутренние сопротивления батареек) R1 и R2. Тогда ЭДС схемы будет 3 В, сопротивление цепи 0.2 Ома, ток по закону Ома 15 А, а сумма падений напряжений U1=15A*0.1 Ом плюс такое же U2. Падение напряжения происходит на внутренних сопротивлениях источников питания. Если бы внутреннее сопротивление не было скрыто от глаз, было бы проще ответить правильно, но, что глаз не видит, того, вроде бы, и нет.

Две батарейки по 1.5 вольта в реальном эксперименте показали 0.2В. Отчего так? Попробуйте менять внутреннее сопротивление R1 в схеме на рисунке 1.10.

Есть еще один интересный эксперимент, который легко провести, чтобы понять, что в электротехнике называют источником тока в отличие от источника напряжения. Представим, что внутреннее сопротивление батарейки очень велико. Скажем, 100 кОм. Напряжение батарейки для определенности пусть будет 10 В. Тогда максимальный ток, который батарейка может отдавать во внешнюю цепь будет не более, чем 10В/100кОм = 0.0001А (или 100 мкА). Если мы к такой батарейке подключим сопротивление в 1 кОм, то ток, практически, не изменится. То есть, в достаточно широком диапазоне изменений сопротивления внешней

цепи ток, протекающий по этой цепи, не будет зависеть от сопротивления цепи. Конечно, напряжение такой батарейки будет меняться очень сильно, но ток нет, что и находит применение на практике, конечно, не в виде батареек с большим внутренним сопротивлением, а в виде специальных генераторов тока. Так например, в мультиметре при измерении величины сопротивления фактически измеряется падение напряжения на испытуемом резисторе, а ток поддерживается постоянным в широком диапазоне измеряемых сопротивлений с помощью источника тока (а не источника напряжения).

Источник питания, напряжение которого мало зависит от сопротивления внешней цепи, мы будем называть источником напряжения, а тот, ток которого мало зависит от сопротивления нагрузки, источником тока.

Итак. Всего несколько понятий: ЭДС, напряжение, ток и сопротивление; всего три закона электротехники: закон Ома и два закона Кирхгофа, – дали нам возможность провести ряд интересных экспериментов. И это далеко не все интересные эксперименты, которые можно было бы провести. Попробуйте составлять цепи из многих батареек и сопротивлений, включая их разными способами, и попытайтесь ответить на вопрос о падении напряжения на любом из резисторов и токе через него! Уверен, вы найдете много интереснейших вариантов.

Прежде, чем продолжить рассказ, я хочу еще раз обратить ваше внимание на то, что электричество всегда несет с собой некоторую опасность. Я уже говорил, что некоторые источники питания при неаккуратном обращении с ними могут привести к травмам, как аккумулятор сотового телефона, но речь шла об ожогах при коротком замыкании. Теперь я хочу сказать о других опасностях. Некоторые аккумуляторы, источники питания многократного применения, очень похожи на батарейки. Если не обратить внимания на предостерегающие надписи, если попытаться заряжать батарейки обычным зарядным устройством, то это может вызвать вытекание электролита, а в качестве электролита может использоваться щелочь. Попадая на руки, электролит тоже вызовет болезненный и долго не заживающий ожог. Прежде, чем пытаться заряжать что-либо, следует проверить, подлежит ли оно заряду? И еще немного о поражении электрическим током. Если через человека проходит очень небольшой ток в несколько десятков миллиампер, то это может вызывать, если и не смертельное, то весьма опасное поражение электрическим током. Всего несколько десятков миллиампер! А батарейка для фонарика может дать ток в несколько ампер! Опасна ли она? Здесь следует вспомнить закон Ома. Если напряжение батарейки 1.5 В, а сопротивление внешней цепи (то есть, человека, который взялся одной рукой за один вывод батарейки, а другой рукой за другой вывод) несколько десятков тысяч ом, то ток будет измерятся единицами микроампер и вреда не принесет. Сопротивление человека в основном зависит от многих внешних и природных факторов и составляет несколько тысяч ом, поэтому при маленьких напряжениях можно не слишком беспокоиться о поражении электрическим током. Но такой подход может сыграть злую шутку, если небольшое напряжение, безопасное с точки зрения расчетов, пройдет через человека неудачным образом. Лучше сразу выработать привычку, даже работая с батарейками от карманного фонаря, никогда не касаться двух полюсов одновременно, если есть возможность, то не проводить измерений под напряжением. Нужно измерить напряжение в схеме, выключите питание, подключите прибор и включите питание. Небрежность, пренебрежение правилами безопасности могут навсегда отбить охоту к работе со схемами. Но небрежность, всегда небрежность. Ни электроника, ни схема ни в чем не виноваты.

Резюмируя свои предостережения, я хочу посоветовать начинающим использовать в экспериментах либо простейшие батарейки, либо качественные (с защитой) блоки питания, а все сомнительные эксперименты вначале проводить за компьютером, используя программы САПР, такие как PSIM и Qucs. Насколько они полезны станет понятно, когда мы начнем говорить о переменном токе.

Конденсатор и индуктивность

Но прежде, чем расстаться с постоянным током, я хочу немного рассказать о конденсаторе. Любая схема (или почти любая) электронного устройства содержит хотя бы один конденсатор. Что он собой представляет?

Рис. 1.11. Конденсаторы

Возьмем две металлические пластины, положим между ними тонкую изолирующую пластину и получим конденсатор. На схеме конденсатор так, примерно, и изображают: две пластины (в профиль), к которым подходят два проводника.

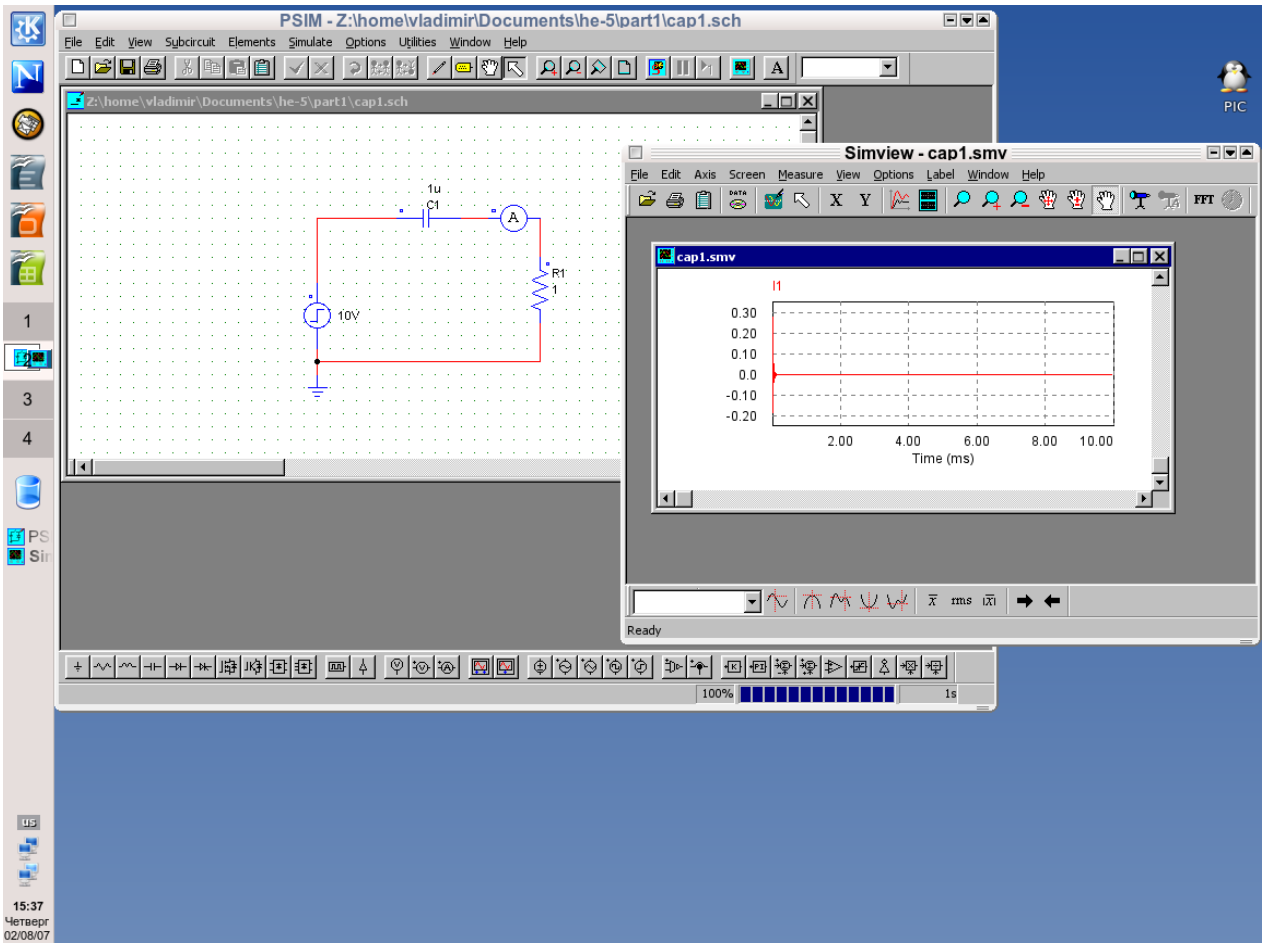


Рис. 1.12. Конденсатор в цепи постоянного тока

Поскольку между пластинами изолятор, не проводящий постоянный электрический ток, то зачем бы нам конденсатор в цепи постоянного тока?

Несколько предвосхищая события, я вместо источника напряжения на схеме использовал нечто вроде его комбинации с выключателем. Но, если вы, как люди воспитанные, сделаете вид, что не заметили этого и повторите эксперимент, используя обычную батарейку и выключатель, то результат должен быть таким, как показано на графике I1, то есть ток в цепи, как показывает амперметр, равен нулю. Цепь постоянного тока разрывается изолятором конденсатора и ток через конденсатор не протекает. Что и зачем я использовал в схеме вместо батарейки? Это генератор ступенчатого напряжения. В начальный момент времени напряжение на его выводах равно нулю, затем сразу становится равным 10 В, как если бы был включен выключатель между цепью из батарейки, конденсатора, амперметра и резистора 1 Ом. Я так и хотел сделать, но не нашел в программе выключателя. В некоторых программах он есть, здесь я не нашел. Но это не существенно.

Зачем мне понадобился выключатель? Если присмотреться к графику, то в самом его начале в момент времени близкий к начальному, на графике видна небольшая размытость. Увеличим ее, поскольку программа PSIM в окне просмотра результатов симуляции Simview позволяет манипулировать с изображением. Сразу хочу предупредить, что теоретический вид результата эксперимента должен отличаться от приведенного ниже, а реальный результат эксперимента будет зависеть от нескольких факторов, о которых, возможно, речь пойдет дальше, когда мы, и если, будем говорить о линиях, паразитных параметрах цепей и т.п.

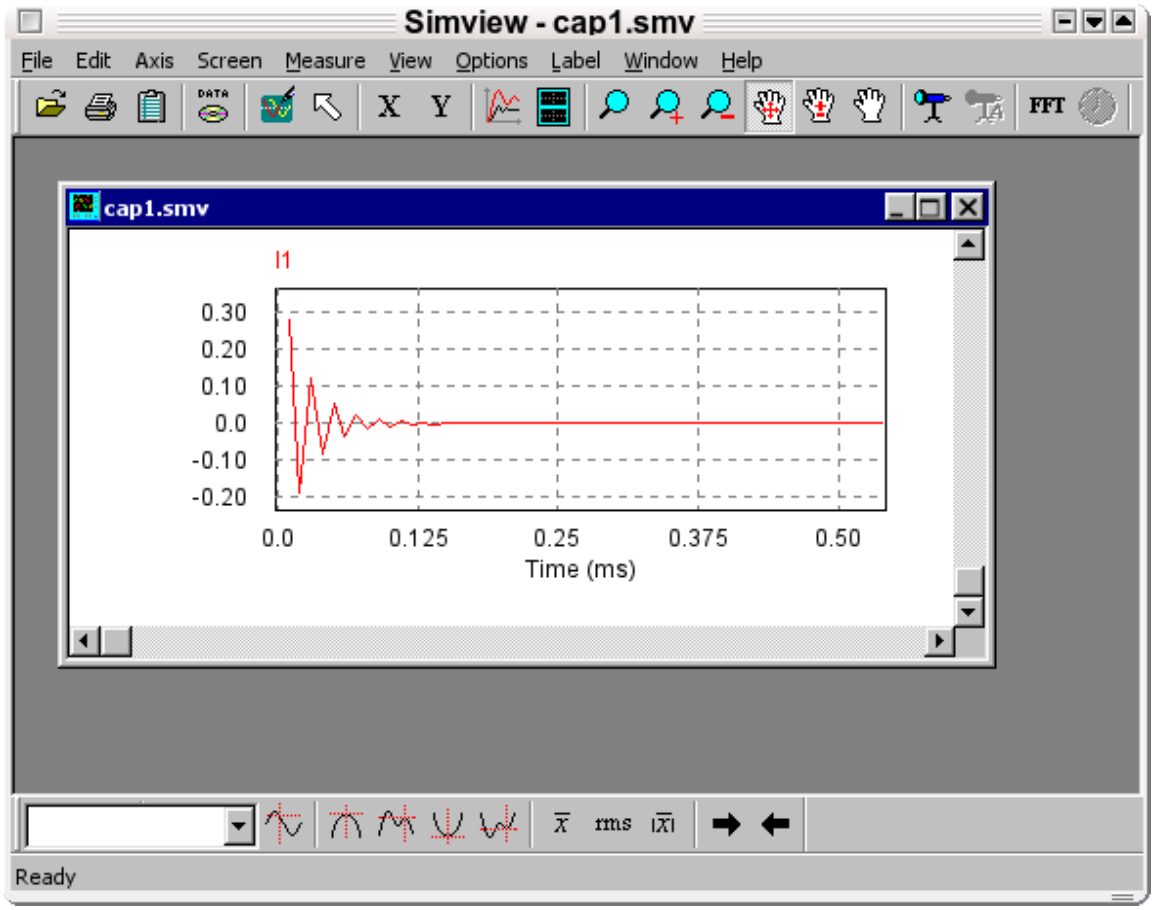


Рис. 1.13. Цепь постоянного тока с конденсатором в начальный момент времени

Как видно, небольшая размытость – это некоторое изменение тока. То есть, хотя постоянный ток в цепи, вообще говоря, не течет, в небольшой промежуток времени при

включении, он все таки умудряется протекать. Такие процессы в короткие промежутки времени при резком изменении параметров (скачком) называются переходными.

Как можно себе представить, что происходит, когда мы включаем выключатель (которого нет на рисунке)? Мы подключаем источник ЭДС к пластинам конденсатора. В металле, из которого эти пластины сделаны, электроны смещаются к поверхности пластины, удаленной от полюса источника, образуя электрическое поле, которое проходит через изолятор к другой металлической пластине, вызывая смещение ее электронов к другому полюсу источника питания. Вот эти-то перемещения (или смещения), как любое направленное движение зарядов, и образуют кратковременный ток, при этом конденсатор заряжается от источника питания.

Конденсатор не пропускает постоянный ток, но может реагировать на изменения напряжения. Оставим себе это на заметку. А так же вспомним, что мы говорили о переменном токе, как токе изменяющемся по величине, в отличие от постоянного. Именно такой ток, изменяющийся по величине (в самом начале) мы наблюдаем на рисунке 1.13. И это заставляет думать, что в цепи переменного тока конденсатор может себя вести иначе, чем в цепи постоянного тока. Чтобы закончить с конденсатором в цепи постоянного тока немного уточним, что я имел в виду, когда выше написал: *конденсатор заряжается...* Буквально то, что написал. Конденсатор накапливает (конденсирует) заряд. Если после его подключения к батарейке отключить батарейку и измерить напряжение на конденсаторе, то прибор покажет напряжение равное напряжению батарейки (в идеальном случае). Если конденсатор очень большой емкости, то к нему можно подключить лампочку, и лампочка будет какое-то время светиться. Пока конденсатор не разрядится. И, кстати, напомню, отчего светится лампочка — то самое рассеивание мощности на резисторе, в данном случае на спирали, от которого резистор может разогреться до «красного каления», а лампочка разогревается до «белого каления».

Емкость конденсатора измеряется в фарадах, но на практике применяют конденсаторы значительно меньшей емкости: мкФ (одна миллионная фарады, микрофарада), нФ (одна тысячная микрофарады, нанофарада), пФ (одна тысячная нанофарады, пикофарада). Емкость конденсатора зависит от площади пластин, чем она больше, тем больше емкость, и толщины изолятора (и материала изолятора), чем он тоньше, тем больше емкость конденсатора. Конденсаторы в виде пластин с изолятором применяют только при изготовлении конденсаторов переменной емкости. Постоянные конденсаторы изготавливают с использованием разных технологий, так в конденсаторах большой емкости, в десятки и тысячи микрофарад, применяют электролит, а такие конденсаторы называют электролитическими. Как правило они полярны, один из выводов помечен плюсом, и его следует подключать к плюсу постоянного напряжения. Вообще, допустимое напряжение очень важный для конденсатора параметр. Если напряжение на конденсаторе превышает это значение, то конденсатор может выйти из строя.

Самый простой способ получить полностью переменное напряжение – это подключить батарейку, скажем, как в схеме на рисунке 1.12, но через переключатель, чтобы в одном положении переключателя батарейка была подключена как на рисунке, а в другом полярность ее подключения менялась на противоположную. Мы получим изменяющееся по величине (в какой-то момент схема отключена от батарейки) и по направлению напряжение, а в цепи пропорционально меняющийся ток, то есть, переменный ток.

Под переменным током будем понимать такой ток, который меняется по величине или/и по направлению.

Если переключателем щелкать очень равномерно, то получится периодический переменный ток. А если описать его с помощью математического выражения, то можно

говорить о законе, по которому меняется ток. На практике очень часто используется переменный ток, меняющийся по закону синуса – синусоидальный переменный ток. Математическое выражение для него выглядит просто: y=A*Sin(x). Конечно, получить из батарейки с помощью переключателя такой вид переменного тока слишком сложно, но такой вид имеет бытовое силовое напряжение, и такой вид переменного напряжения дают многие генераторы, используемые при работе с электронными приборами.

На рисунке ниже я использовал источник синусоидального переменного напряжения V1. Функция синуса периодическая, то есть, существует отрезок времени, через который график функции повторяет свою форму, вновь и вновь. В электротехнике период измеряют в секундах. Величина, обратная к периоду, показывающая количество колебаний в секунду, называется частотой и измеряется в герцах.

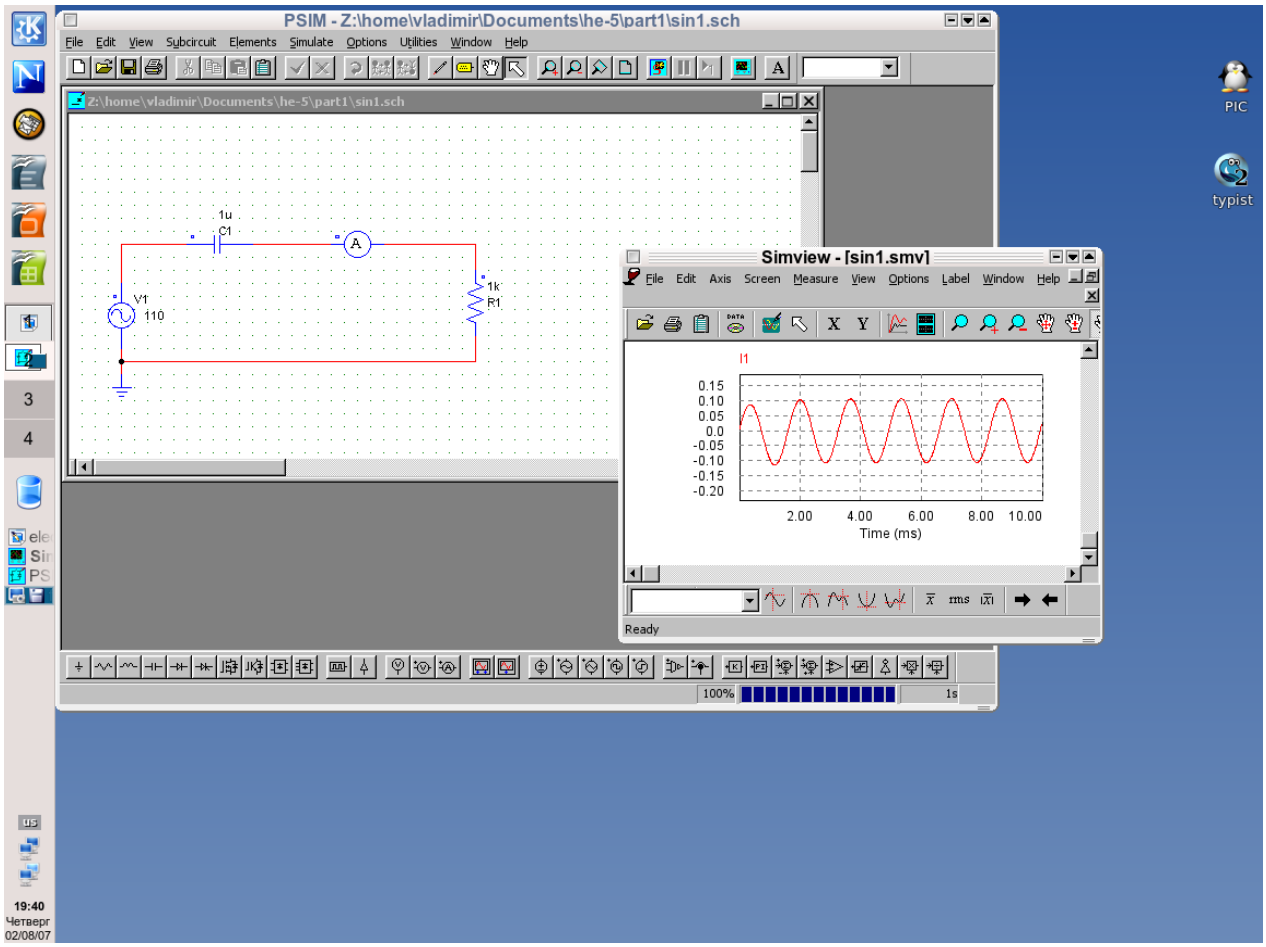


Рис. 1.14. Вид синусоидального переменного тока

Период переменного тока на рисунке чуть меньше двух миллисекунд, это соответствует частоте около 500 Гц.

Мы подозревали, что конденсатор ведет себя по отношению к переменному току иначе, чем к постоянному. И действительно, если постоянный ток конденсатор не пропускает, то переменный ток проходит через него. Этим свойством конденсатора часто пользуются для того, чтобы отделить, как говорят, постоянную составляющую от переменной, и вы можете встретить такое название конденсатора, как разделительный конденсатор, что относится к функции, выполняемой конденсатором.

Если в программе PSIM воспользоваться не амперметром, а осциллографом, прибором для

наблюдения в первую очередь за переменным напряжением, который подключить параллельно резистору R1, то мы сможем наблюдать переменный ток и более высоких частот. Дело в том, что многие амперметры, измеряющие переменный ток, хорошо работают только с токами низких частот. А осциллограф отображает процессы и очень высоких частот. Современные осциллографы, доступные любителю, могут работать до частот в сотни мегагерц.

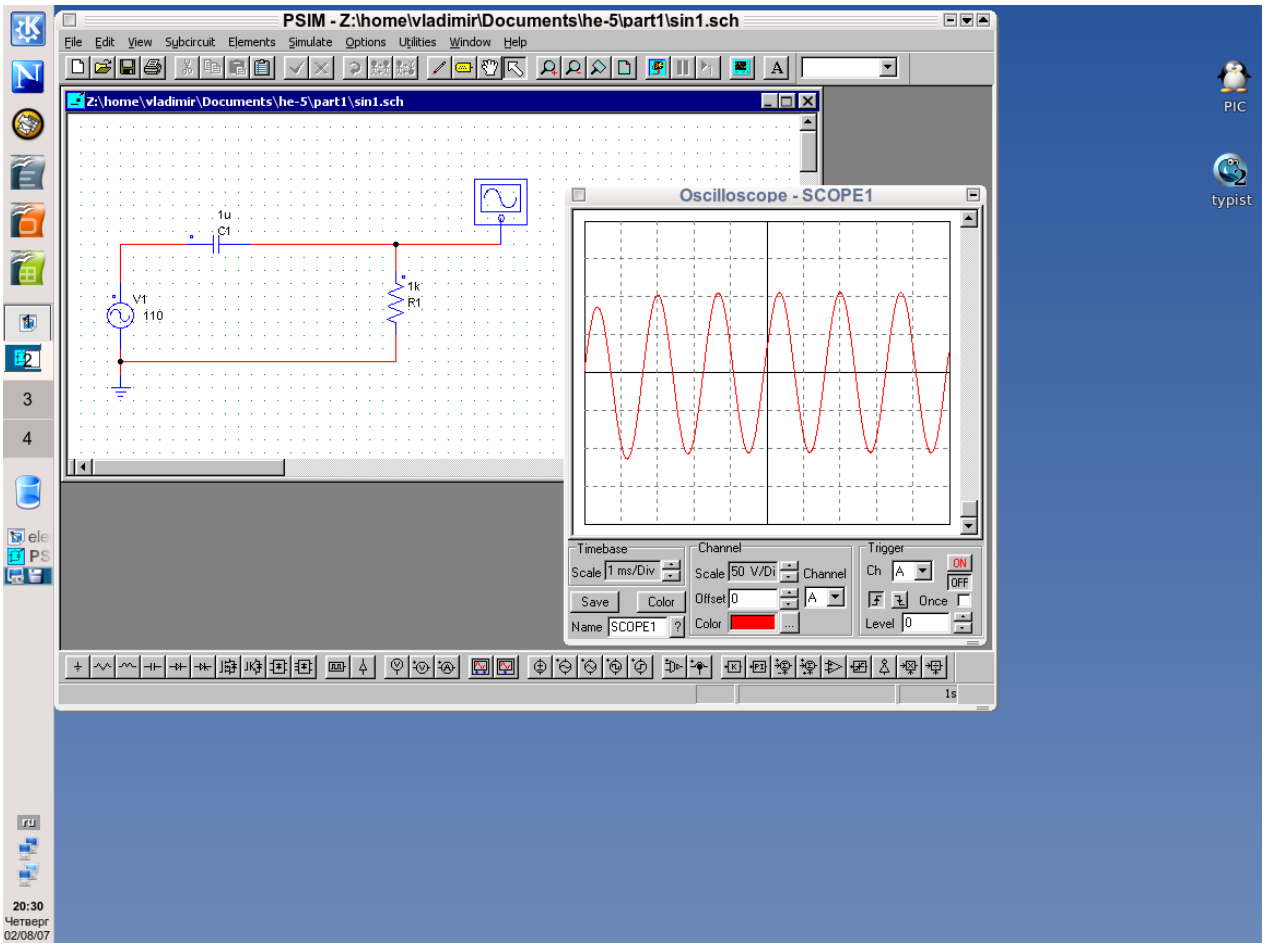


Рис. 1.15. Наблюдение переменного напряжения в цепи конденсатор-резистор

Если посмотреть на осциллограмму напряжения и по виду осциллографа (по надписям) понять, что каждое вертикальное деление сетки на экране – это напряжение в 50 В, то амплитуда – наибольший «размах» напряжения от нуля или середины – получается около 100 В, тогда как амплитуда напряжения источника переменного тока V1 равна, как это показано на схеме, 110 В. Похоже, что как резистор оказывает сопротивление постоянному току, так и конденсатор оказывает сопротивление прохождению переменного тока, и на нем появляется падение напряжения. От чего зависит это сопротивление?

Проведем несколько экспериментов. Я использую программу PSIM для их описания, а макетную плату для проверки этого описания. Правда, в отличие от программы, мой низкочастотный генератор позволяет получить напряжение на выходе только 2 вольта, но мультиметр может измерять переменное напряжение такой величины. Если между компьютерным и «живым» экспериментом появятся различия, я о них скажу.

Попробуем изменить частоту (уменьшить) источника тока:

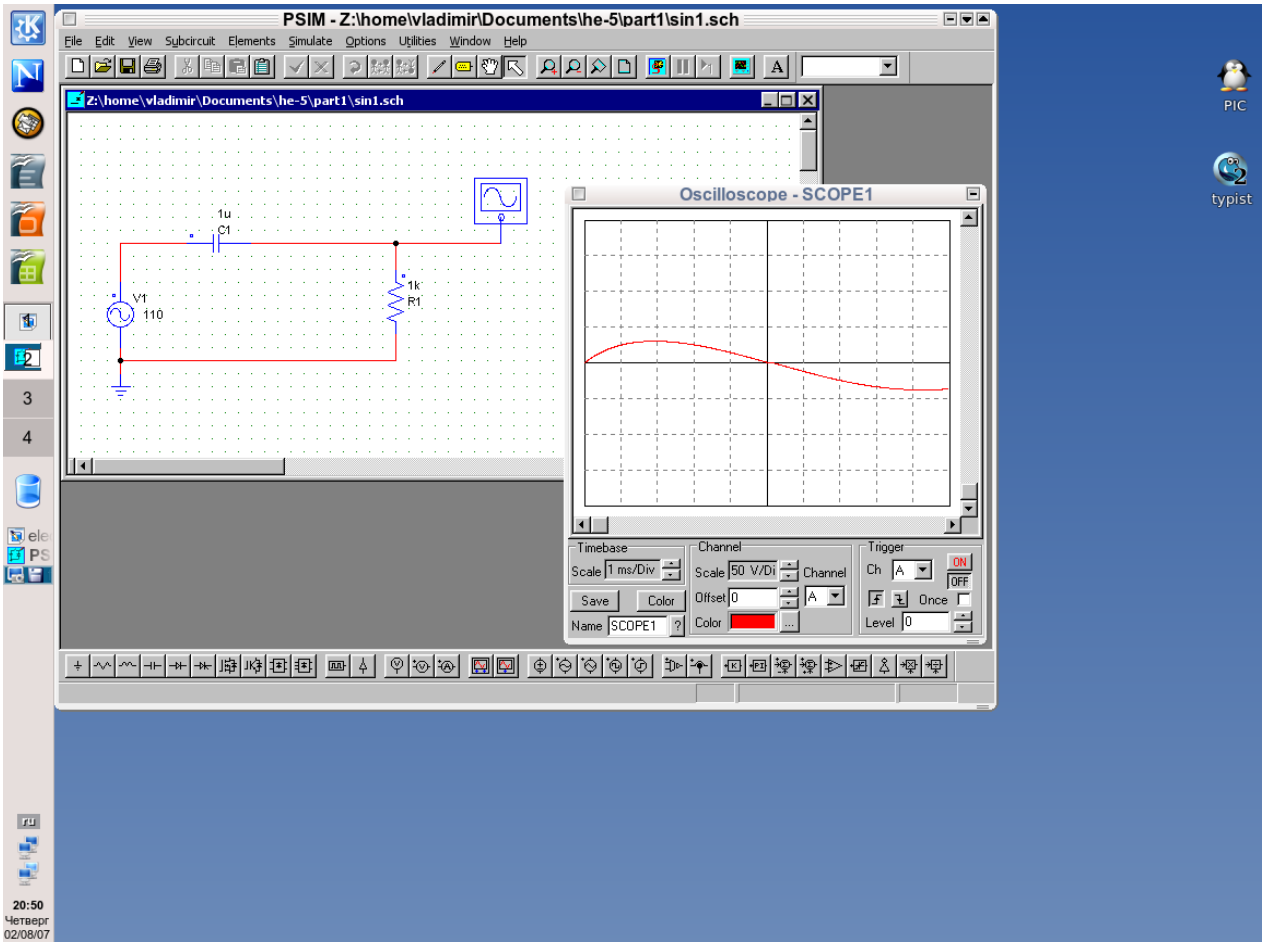


Рис. 1.16. Изменение осциллограммы при уменьшении частоты в 10 раз

На рисунке видно, что амплитуда напряжения уменьшилась существенно, став менее 50 В. Значит, сопротивление зависит от частоты переменного тока. А увеличив значение конденсатора в 10 раз (до 10 мкФ, микрофарад) мы почти восстановим амплитуду. Значит, сопротивление конденсатора зависит от его величины.

И действительно, сопротивление конденсатора синусоидальному переменному току равно $1/2\pi fC$, где f – частота в Гц, C – емкость в Ф (фарадах). Если сопротивление резистора называют активным сопротивлением, то сопротивление конденсатора переменному току называют реактивным (реакция на изменения тока).

В электронике, как правило, нельзя обойтись без источника постоянного тока – источника питания любой электрической схемы, будет ли это схема автоматики или радиоприемника. А переменный ток, чаще играет роль сигналов, несущих полезную информацию. Чаще переменный ток в схемах бывает током, изменяющимся по величине, чем током изменяющимся по направлению. Работа электронного устройства в целом связана с преобразованиями сигналов для придания им либо необходимой величины при заданной форме, либо необходимой формы и величины, например, для переноса информации. Когда вы отправляете с компьютера письмо своему приятелю (или подруге), то по электрической цепи (возможно, местами в виде радиоволны) вы отправляете сигнал очень сложной формы, который и переносит ваше письмо к компьютеру приятеля (или подруги), где сигнал преобразовывается в понятные им (приятелю или подруге) буквы и цифры, и даже рожицы в конце фраз. Когда вы разговариваете по мобильному телефону, ваша речь преобразуется телефоном в радиосигналы, долетающие до телефона подруги (или приятеля), где они вновь

преобразуются в звук, несущий информацию о том, что вы скучаете, не зная, чем заняться. Если ваш приятель радиолюбитель, он посоветует вам заняться этим увлекательным делом.

Но о сигналах мы поговорим позже.

А сейчас вспомним, что закон Ома для цепи постоянного тока был весьма полезен. Нельзя ли использовать его для цепей переменного тока. В простейшем виде, напомню, закон Ома звучит так: напряжение на участке цепи равно произведению тока, протекающего по этому участку, на сопротивление этого участка постоянному току. Однако как быть с переменным током, если его величина все время меняется? Можно сказать, что закон Ома будет справедлив для каждого мгновения, но удобнее ввести понятие действующего значения переменного тока.

Под действующим значением переменного тока будем понимать такое его значение, которое оказывает такое же тепловое действие, что и постоянный ток той же величины за время равное периоду переменного тока.

Выше я упоминал о лампочке, доведенной до «белого каления». Довести ее до этого можно с помощью батарейки, дающей постоянный ток. Но если мы доведем ее до того же состояния с помощью переменного тока, то его величина будет иметь действующее значение, тоже измеряемое в амперах, численно равное величине постоянного тока. А напряжение (действующее), измеряемое тоже в вольтах, численно будет равно постоянному напряжению. И закон Ома будет звучать также, как для постоянного тока. И оба полезных закона Кирхгофа будут иметь место для переменного тока, с учетом, конечно, его переменного характера. Я не буду уверять вас, что я абсолютно прав, напротив, советую вам проверить это, используя либо программу PSIM (или аналогичную программу САПР), что удобнее и спокойнее всего на «первых порах», или проверьте это на макетной плате, используя мультиметр и генератор синусоидального напряжения в качестве источника переменного тока. Я не советую использовать силовое напряжение по причине опасности работы с ним, даже если вы примените понижающий трансформатор. Лучше собрать простейший генератор на цифровой микросхеме, взяв схему из этой книги дальше. Можно быстро собрать генератор прямоугольных импульсов. Он не будет генератором синусоидального напряжения, и мультиметр будет показывать не вполне правильные значения, но уверен, что проделанные эксперименты, каковы бы ни были результаты, доставят вам удовольствие. Используя генератор, мультиметр, несколько конденсаторов и резисторов вы можете придумать множество интересных экспериментов, помимо тех, о которых шла речь.

Когда я заговорил о законе Ома для переменного тока, я упоминал мгновенные значения переменного тока. Можно сказать, что в каждое мгновение переменный ток находится в *определенной фазе* своего изменения: вот он равен нулю, затем растет, достигает наибольшей величины, начинает уменьшаться и т.д. И одним из отличий активного сопротивления от реактивного будет то, как изменяются напряжение на них и ток через них. Я немного изменю схему, применив двухканальный осциллограф, есть такая возможность в программе PSIM, возможно у вас есть «физический» двухканальный осциллограф, для наблюдения за напряжением на конденсаторе и напряжением на резисторе (пропорциональном току через конденсатор), что заменит мне наблюдение за током через конденсатор. На рисунке ниже канал A показывает напряжение, а канал B ток через конденсатор. Кстати, на схеме осциллограф подключен только одним проводом, что удобно при работе с программой. При этом подразумевается, но не отображается, что общий провод осциллографа соединяется с общим проводом схемы. Кроме того, на схеме я уменьшил величину резистора R1 до 10 Ом, с тем чтобы получить наиболее яркую иллюстрацию того, что хочу отметить по поводу фаз.

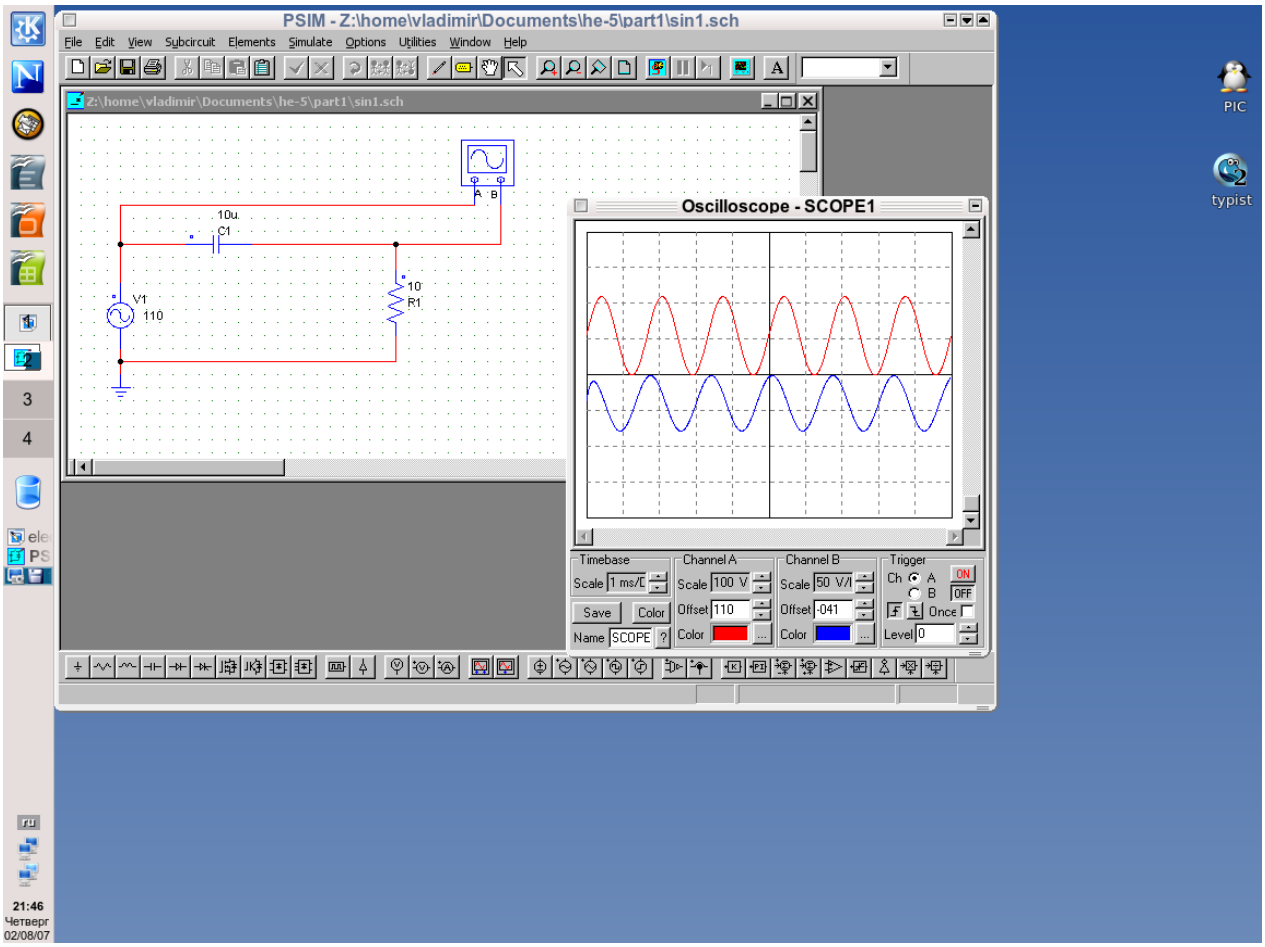


Рис. 1.17. Наблюдение напряжения и тока через конденсатор

Два сигнала в центре экрана осциллографа, как видно на рисунке, находятся в разных фазах изменения: напряжение на конденсаторе и ток через него находятся в разных состояниях в любой момент времени. Говорят, что они не совпадают по фазе. Если вы замените конденсатор обычным резистором того же сопротивления, то оба сигнала совпадут по фазе. Напряжение и ток в случае активного сопротивления совпадают по фазе. Для проведения эксперимента на макетной плате мне приходится включать осциллограф, да еще придумать, как на одноканальном осциллографе увидеть сдвиг фаз. Это не так наглядно, как на двухканальном. Когда мы будем разговаривать о схемах, сигналах, обратной связи, я постараюсь не забыть и расскажу, как это сделать. Или приведу схему, которую сам некогда собрал, чтобы превратить одноканальный осциллограф в двухканальный.

Еще одним представителем реактивного сопротивления является индуктивность. Хотя индуктивностью, впрочем как и емкостью, обладают многие элементы схем, обычно индуктивность – это катушка с проводом. Если такая катушка имеет сердечник из специального материала, например, феррита или стали, то ее индуктивность больше. Как и конденсатор, индуктивность обладает сопротивлением переменному току, но в отличие от конденсатора это сопротивление растет с ростом частоты, то есть, индуктивное сопротивление прямо пропорционально частоте и индуктивности. Раньше для получения индуктивности на цилиндрический каркас наматывали провод, сегодня появились новые технологии изготовления индуктивностей, в основном из-за того, что частоты устройств стали очень высокими. Технология намотки провода на каркас сохранилась там, где индуктивность применяется на низких частотах, да при изготовлении трансформаторов. Трансформатор – устройство, использующее тот факт, что две катушки на одном каркасе

оказываются связаны индуктивно, и если на одну, их называют обмотками, если на одну обмотку трансформатора подать переменный ток, то вторая обмотка ведет себя как источник переменного тока. Если количество витков обоих обмоток трансформатора одинаково, то и напряжение на второй обмотке будет равно напряжению на первой. Если количество витков второй обмотки меньше, чем первой, то напряжение на ней будет меньше. Таким образом, мы получим *понижающий* трансформатор. Его обмотки принято называть первичными и вторичными.

В сегодняшних бытовых устройствах все чаще используют импульсные блоки питания. Они много легче старых трансформаторных, отчего меньше весит телевизор или DVD-проигрыватель, но и они, зачастую, имеют в своем составе трансформатор. Суть перемен в преобразовании частоты питающей сети, 50 Гц в нашей стране, в более высокую частоту. Поскольку индуктивное сопротивление зависит от частоты, трансформатор, работающий на более высокой частоте, имеет меньшие габариты и вес, чем пятидесятигерцовый. Так используется зависимость индуктивного сопротивления от частоты. Но не только так.

И конденсатор, и индуктивность реагируют на изменение частоты. Изменяется их сопротивление переменному току. Но у конденсатора сопротивление уменьшается с ростом частоты, а у индуктивности сопротивление увеличивается. А что, если включить их вместе, конденсатор и индуктивность? Вместе их можно включить либо последовательно, либо параллельно. В любом случае возникает некая особая ситуация, когда на какой-то частоте их сопротивления переменному току будут равны. Проведем такой эксперимент:

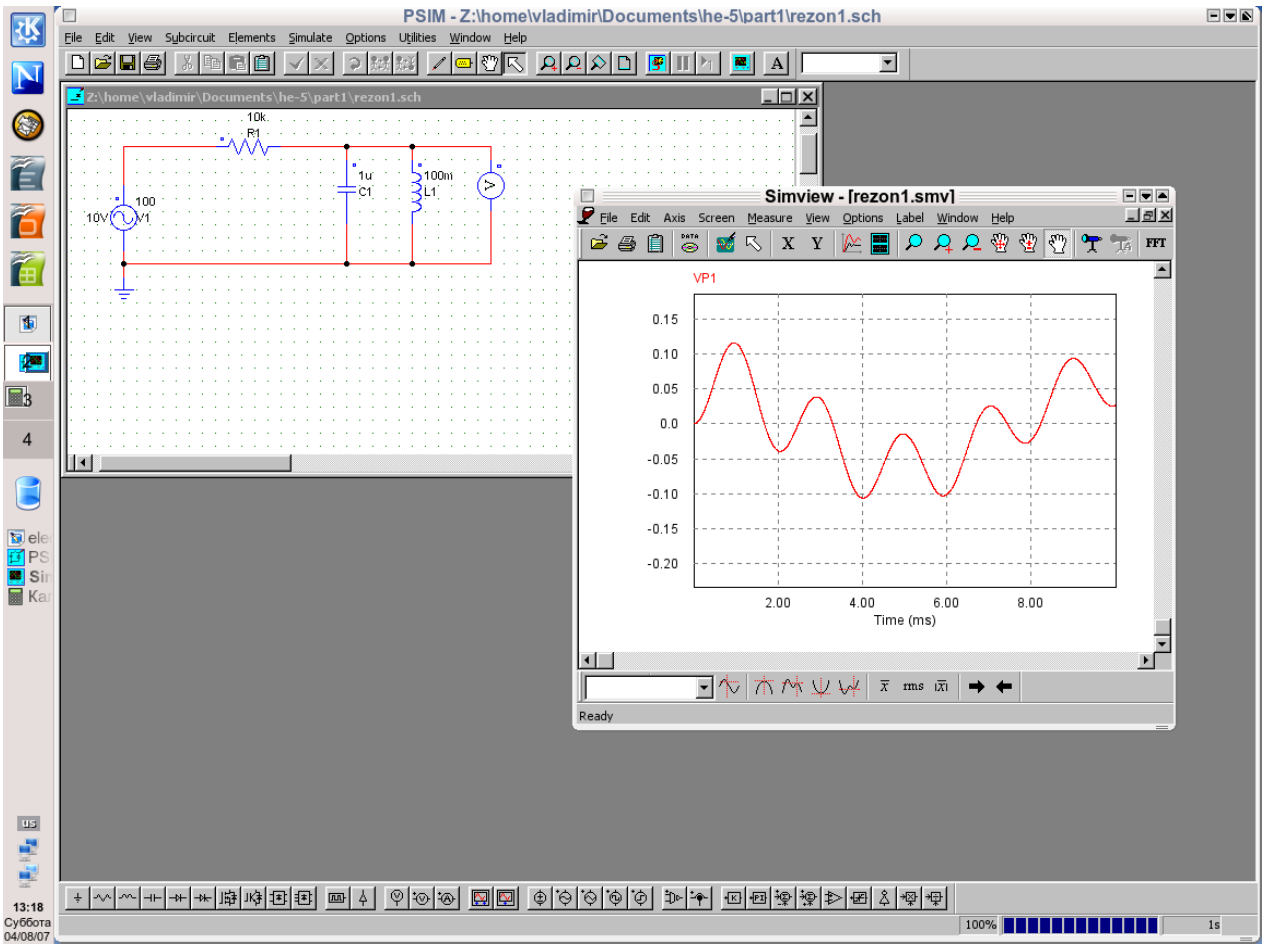


Рис. 1.18. Параллельное включение конденсатора и индуктивности

Поскольку меня в данный момент интересует сопротивление параллельно включенных конденсатора и индуктивности, на схеме они включены последовательно с резистором 10 кОм. Их сопротивление я отслеживаю по падению напряжения на них с помощью вольтметра. Источник переменного тока (или, точнее, напряжения) V1 позволяет мне менять частоту, которая в данный момент равна 100 Гц. Напряжение источника 10 В. При емкости конденсатора 1 мкФ и индуктивности катушки 100 мГн (генри – единица индуктивности) напряжение на паре конденсатор-индуктивность, как видно из графика, порядка 0.1 В.

Изменив частоту источника переменного напряжения, кстати, синусоидального, до 1000 Гц, я получу новый график:

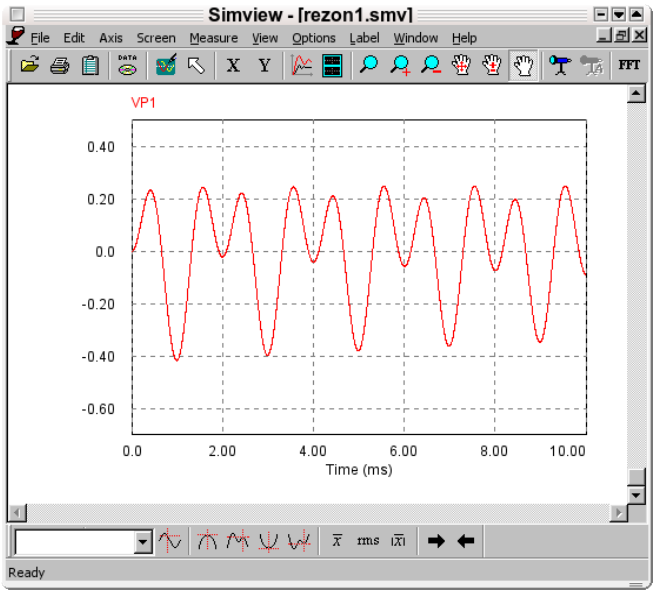


Рис. 1.19. График напряжения на контуре при частоте 1000 Гц

А повторив эту операцию для частоты 500 Гц, получу такой вид графика:

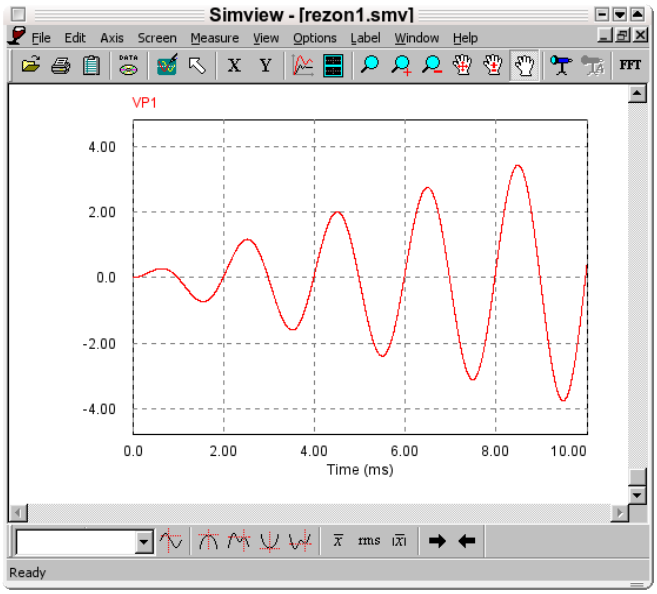


Рис. 1.20. График напряжения на частое 500 Гц

Как видно из рисунка, напряжение на частоте 500 Гц возрастает до 4 В, а это означает, что сопротивление на частоте 500 Гц резко увеличивается. Именно эту особенность – реакцию на определенную частоту – я и хотел отметить. Свойство выделять определенную частоту переменного тока у пары конденсатор-индуктивность называют резонансом. Для каждого значения конденсатора и индуктивности есть своя резонансная частота. Ее значение равно единице, деленной на корень квадратный из произведения этих значений, умноженный на коэффициент 2π. Я не буду приводить расчетную формулу, которую легко получить из исходной предпосылки равенства емкостного и индуктивного сопротивлений, но отмечу, что это свойство достаточно широко применяется для выделения определенной частоты из множества других, особенно в радиотехнических схемах.

Я использовал параллельное включение конденсатора и индуктивности (катушки индуктивности), в этом случае говорят о параллельном резонансе, но можно включить их последовательно, получив последовательный резонанс. Выше я говорил о том, что напряжение на конденсаторе и ток через него не совпадают по фазе. Посмотрите, что происходит с напряжением и током резонансного контура (так называют пару конденсатор-индуктивность), используя схему аналогичную той, что я приводил для конденсатора. Очень интересный эксперимент.

Диод и транзистор

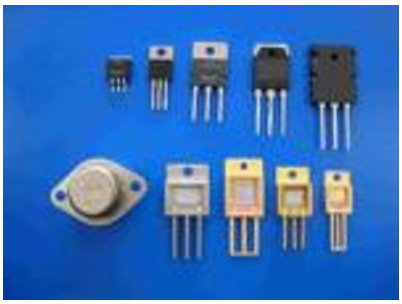


Рис. 1.21. Транзисторы

На плате передо мной есть несколько диодов и транзисторов, немного, но они есть. Их, в отличие от резисторов, конденсаторов и катушек индуктивности, относят к активным элементам электрической цепи. Без них не обходятся ни усилители, ни генераторы. Выпрямители, стабилизаторы, индикаторы, фотоприемники – вот неполный перечень применений диодов. А если разобрать любую микросхему, аналоговую или цифровую, то можно убедиться в том, что это царство транзисторов. Диоды и транзисторы изготавливаются особым образом из полупроводниковых материалов. Напомню, что полупроводником называют материал, который по свойству проводить электрический ток, занимает промежуточное положение между проводниками и изоляторами. Когда-то по этой причине они были мало интересны, провода из них не сделаешь, слишком большое сопротивление, а в качестве изолятора лучше использовать резину или текстолит. Причина плохой проводимости тока у полупроводников в их строении. Количество электронов, способных перемещаться по материалу, много меньше, чем у металлов, но они есть, что мешает использовать материал в качестве изолятора. Мало того, у одних типов полупроводников, как и у металлов, есть электроны-бродяги, а у других типов полупроводников все еще путаней – вместо добропорядочных носителей зарядов есть вакансии для неприкаянных электронов, которые называют «дырками». Интерес к полупроводникам появился тогда, когда из полупроводников двух типов сделали двухслойную конструкцию, у которой обнаружилось прелюбопытное свойство – пропускать постоянный ток в одном направлении, и не

пропускать в другом.

Проведем два эксперимента с полупроводниковым диодом: соединим последовательно диод и резистор (чтобы ограничить ток), добавим амперметр, подключим все это к батарейке. В первом случае включим батарейку в одной полярности, а во втором в противоположной, и посмотрим, что у нас происходит с постоянным током, проходящим по нашей схеме.

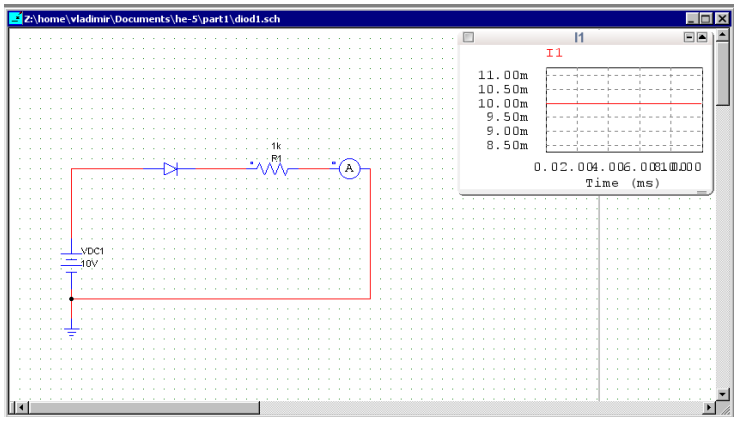


Рис. 1.22. Диод (в прямой полярности подключения) в цепи постоянного тока

В этом случае ток в цепи равен 10 мА (тысячным долям ампера), что мы, зная закон Ома, можем получить расчетным путем: ток равен напряжению, деленному на сопротивление. Разделим напряжение (ЭДС) 10 В на сопротивление (резистора) 1 кОм (тысяча Ом) и получим ток в 10 мА (десять тысячных ампера). Диод ведет себя себя как проводник, то есть, так, как будто его почти совсем нет. Изменим полярность батарейки на противоположную.

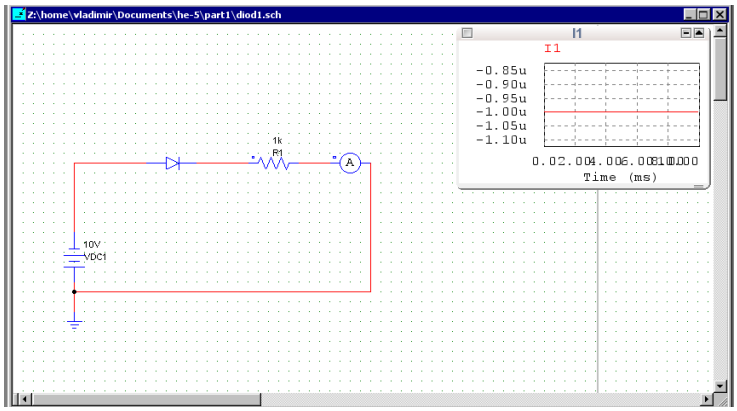


Рис. 1.23. Диод (в обратной полярности) в цепи постоянного тока

Ток через амперметр в этом случае равен 1 мкА (одной миллионной доли ампера), а мы можем рассчитать сопротивление цепи, разделив напряжение на ток: 10 В разделим на ток 1 мкА и получим сопротивление 10 МОм (десять миллионов Ом). Этот расчет можно проверить опытным путем – замените диод резистором в 10 МОм и, если получите ток через амперметр I1 равным 1 мкА, то расчет сделан верно.

Что полупроводниковый диод чувствителен к полярности приложенного напряжения можно убедиться с помощью мультиметра. У него есть режим измерения, который помечен значком диода (диод обозначается на схемах в виде треугольника, упирающегося в стенку), подключая диод к мультиметру в этом режиме, можно увидеть, что тот показывает сопротивление порядка нескольких сотен ом при одной полярности включения и показывает

перегрузку (горит только единичка в старшем разряде) при смене полярности. Именно так проверяют работоспособность полупроводникового диода мультиметром.

Как объясняется это свойство двухслойной конструкции из полупроводниковых материалов разного типа? Представим, что у нас есть две маленькие тонкие пластинки из полупроводников разного типа. Поверхности пластинок так идеально отшлифованы, что соединив их вместе, мы получим, как бы, единую пластинку. В этом случае электроны-бродяги из полупроводника одного типа (его называют полупроводником типа «n») могут перемещаться через границу, попадая в полупроводник другого типа (его называют полупроводником типа «p»), где электроны охотно занимают вакантные места в атомах материала (заполняют «дырки»). Но до этого электрически нейтральные атомы на границе раздела материалов становятся заряженными – те, что потеряли электроны, положительно, присоединившие электроны, отрицательно. Между этими заряженными атомами на границе раздела возникает электрическое поле, которое теперь уже мешает электронам из материала типа «n» попадать в материал типа «p». В целом наша конструкция остается электрически нейтральной, сколько было каких зарядов, столько их и осталось. Но, когда мы подключаем к нашей конструкции внешний источник ЭДС, то создаваемое им поле может ослабить поле на границе раздела при прямой полярности включения, и тогда электроны от одного полюса источника питания могут двигаться по диоду, как по обычному полупроводнику. Полупроводник будет проводить ток хуже проводника, но достаточно хорошо. Если мы изменим полярность источника ЭДС, то внешнее поле усилит поле на границе раздела и электроны от одного полюса к другому почти не смогут перемещаться из-за противодействия результирующего электрического поля. Наша конструкция почти не проводит ток, как изолятор.

Свойство диода столь разно проводить ток разной полярности используется для «выпрямления» переменного тока. Заменим батарейку на схеме источником переменного тока, чтобы понаблюдать за током. Как выглядит переменный ток, будем считать, мы знаем.

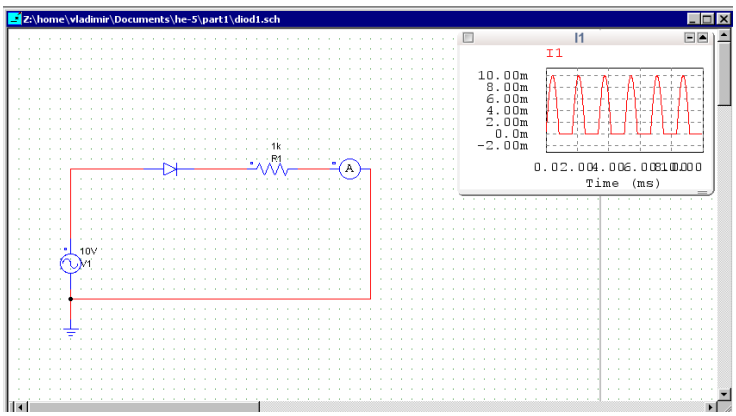


Рис. 1.24. Диод в цепи переменного тока

Диод как бы «отрезает» отрицательную полу-волну синусоидального переменного напряжения. Если поменять полярность включения диода, то диод будет отрезать другую полу-волну.

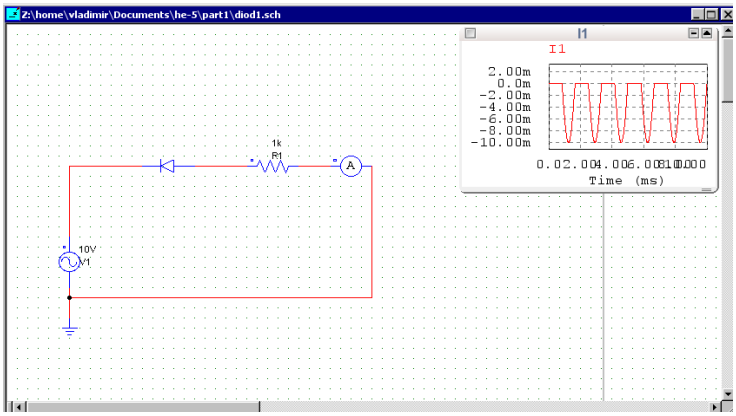


Рис. 1.25. Обратная полярность включения диода

То, что пропускает диод, остается переменным током, но его направление почти не меняется, меняется только величина. И если в цепь добавить конденсатор (для конденсации «результатов»), то мы получим почти постоянный ток из переменного.

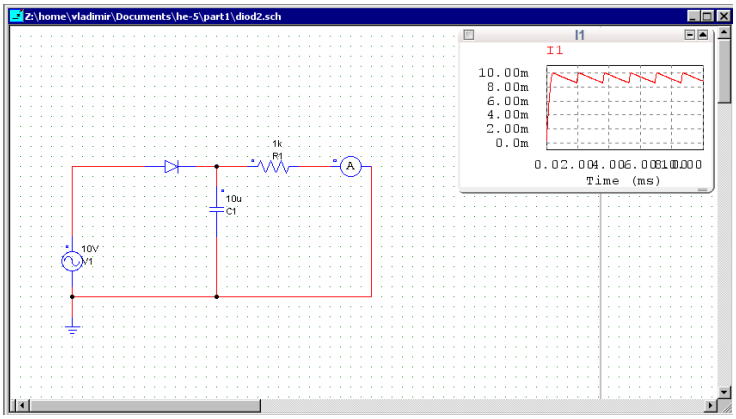


Рис. 1.26. Выпрямление переменного тока

Результирующий ток будет тем больше похож на постоянный, чем больше величина конденсатора C1 в нашей схеме выпрямителя. Ведь, в сущности, мы «выпрямили» переменный ток.

Вот каким полезным свойством обладает простейшая конструкция из полупроводников разного типа проводимости. И почти так работаю блоки питания электронных устройств от силовой сети 220 В. Конечно с участием понижающего трансформатора, поскольку напряжение питания многих электронных устройств 5-10 В, а не 220. Как именно устроен простейший блок питания? Добавьте трансформатор к схеме, и все. Правда, схема, показанная выше, осуществляет *однополупериодное* выпрямление. То есть, выпрямляется только один полу-период переменного напряжения. Но можно увеличить количество диодов, получив *двухполупериодное* выпрямление, либо с мостовой схемой при четырех диодах, либо с двумя диодами и двумя выходными обмотками трансформатора. Думаю, с этим вы разберетесь без меня, а я хочу только отметить, что конденсатор, добавленный к схеме, заряжается, когда присутствует полу-волна, и разряжается, когда она отсутствует, играя роль источника тока для схемы в это время и «сглаживая» напряжение на выходе блока питания, поэтому конденсатор в схеме выпрямителя часто называют *сглаживающим*, а напряжение на выходе *пульсирующим*.

А я хочу продолжить рассказ о конструкциях полупроводников.

Положим, мы возьмем не две пластинки из полупроводниковых материалов разного типа, а три. Соединим эти три кусочка полупроводника, так, чтобы между двумя пластинками одного типа была тоненькая пластинка другого. В итоге мы получим конструкцию, имеющую три вывода, и называющуюся биполярный транзистор. В зависимости от выбора типа полупроводника средней пластинки мы получим транзистор «n-p-n» или «p-n-p» типа.

Итак, биполярный транзистор типа «n-p-n». Он имеет две области с электронным типом проводимости, между которыми заключена область полупроводника с «дырочным» типом проводимости. Как и у диода на границах областей образуются пограничные слои, но теперь их два. Как и у диода, в зависимости от полярности приложенной ЭДС эти пограничные слои будут влиять на пропускную способность, оказывая сопротивление постоянному току, зависящее от приложенного напряжения. Такую конструкцию можно было бы представить в виде двух диодов, включенных встречно, если бы ни одно «но!». Область, заключенная между двумя материалами с одним типом проводимости, конструктивно очень тонкая. Вывод, подключенный к ней, называется у транзистора *базой*. Конструкция транзистора такова, что области одинаковой проводимости не равнозначны, одна из них играет роль поставщика носителей тока и называется *эмиттер*, другая роль сборщика носителей и называется *коллектор*. Возникающие в отсутствии источников ЭДС два пограничных слоя, чем-то похожие на заряженные конденсаторы, препятствуют перемещению носителей тока из эмиттера в базу и из коллектора в базу, но для носителей, прошедших из эмиттера в базу поле перехода база-коллектор (пограничный слой называют переходом) становится «попутным», помогающим им перейти в область коллектора. Два источника питания транзистора включают так, что переход эмиттер-база *смещен в прямом направлении*, то есть, поле пограничного слоя компенсируется, а переход коллектор-база *смещен в обратном направлении*, его поле усиливается внешним. Ток двух источников питания будет частично протекать по базовому выводу, но основная масса носителей от их поставщика, эмиттера (из-за того, что область базы очень тонкая), будет попадать в «попутное» поле перехода коллектора-база. Вот неполная картина происходящего в транзисторе. В дальнейшем я, надеюсь, не буду обращаться к ней, но какую-то картинку происходящего полезно иметь перед глазами.

Мне кажется, что для работы с транзистором удобнее рассматривать его, как распределитель токов: ток эмиттера разветвляется в базу и коллектор (для p-n-p транзистора), а между токами базы и коллектора существует строгая взаимосвязь – ток коллектора почти всегда равен произведению тока базы на постоянную, которую называют (статическим) коэффициентом усиления транзистора по току. Этот коэффициент у разных транзисторов меняется от нескольких десятков до сотен единиц. Именно благодаря этому свойству транзистор имеет то широкое применение, которое он имеет.

На практике редко применяют включение двух источников постоянного напряжения для питания базовой и коллекторной цепей, но лучше все-таки это нарисовать, чтобы легче было понимать, как на практике включают транзистор, чтобы использовать его «активное» свойство усиливать ток, втекающий в базу транзистора (или из нее вытекающий у транзистора другого типа).

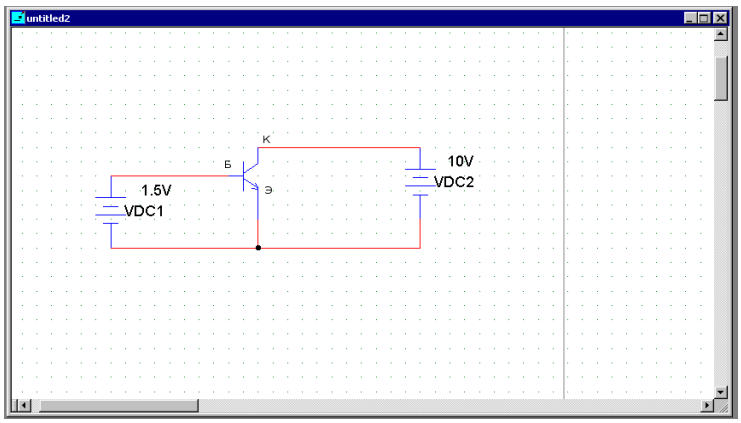


Рис. 1.27. Схема включения транзистора

Хочу сразу заметить, что не следует пытаться проводить подобный эксперимент, впаяв транзистор на макетную плату. Дело в том, что переход транзистора база-эмиттер (эмиттер обозначен стрелкой) ведет себя подобно диоду, то есть при том включении (и напряжении источника VDC1), которое изображено выше, ток через переход будет весьма большим, вызывая разогрев перехода, который попытается «засветиться» подобно лампочке при нагреве, но сгорит быстрее, чем вы успеете заметить это свечение. Я даже не стал рисовать землю, чтобы показать, что не запускал симуляцию схемы. Впрочем, на компьютере вы можете провести любые эксперименты без опасений за целостность ваших компонентов, чем компьютер и полезен.

Итак, источник тока VDC1 создает на переходе база-эмиттер электрическое поле, ослабляющее поле пограничного слоя. Носители, выходя из области эмиттера попадают под действие разгоняющего их поля источника VDC2, и почти все «улетают» в область коллектора. Влияние поля, создаваемого источником VDC1 на ток коллектора очень велико, а роль источника VDC2 можно назвать «направляющей», направляющей ток к коллектору, чтобы он весь не уходил в базу. Если убрать этот источник тока, то весь ток, выходящий из эмиттера, пройдет в базовую цепь транзистора.

Если для транзистора типа n-p-n удобно говорить о токе, образованном электронами, рассуждения о токе эмиттера, разветвляющегося на ток базы и ток коллектора, то для транзисторов типа p-n-p вполне уместно повторить те же рассуждения, считая, что носителями тока в этом случае выступают «дырки». Ход рассуждений получается одинаков.

Для экспериментов, на макетной плате или за компьютером, удобнее следующая схема:

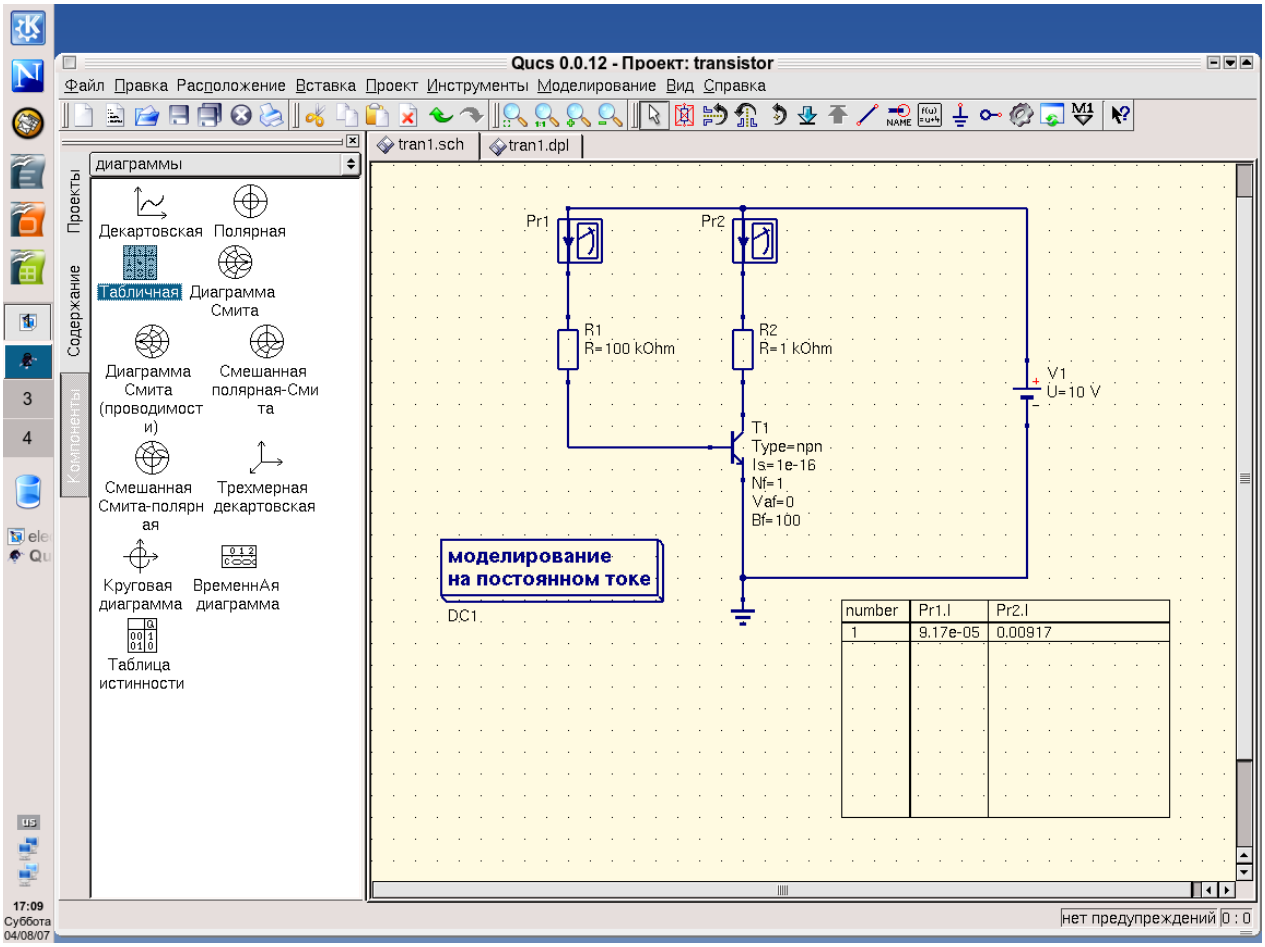


Рис. 1.28. Наблюдение токов базы и коллектора транзистора

Такая схема включения транзистора гораздо чаще встречается на практике. Программа, как вы заметили, которую я использовал – это другая программа, и она называется Qucs. Есть несколько причин, по которым я сменил программу.

Первая причина банальна, о ней не стоило бы говорить, но она есть. Я не смог найти транзистор в программе PSIM, который позволил бы мне легко проиллюстрировать несколько опытов с транзистором. Выбор оставался небольшой, либо потратить некоторое время на чтение документации, либо сменить программу. Я выбрал второе. И для этого была еще одна причина.

Вторая причина смены программы – если первая предназначена для работы в Windows, то вторая для работы в Linux. Хотя первая работает у меня в Linux, а вторая может работать в Windows, я хочу уделить одинаковое внимание программам для обеих операционных систем, переходя к той, что в данный момент удобнее.

В справочной литературе приводится множество параметров для каждого транзистора. Впервые сталкиваясь с этим, испытываешь затруднение – какой из параметров транзистора самый важный? Как выбрать транзистор? Мое мнение – начинающему радиолюбителю не следует «зацикливаться» на этом. Если вы собираете схему, которая вам приглянулась, постарайтесь использовать те транзисторы, которые указаны в схеме, а если их нет, посмотрите, в описании часто указывают возможную замену. Со временем вы разберетесь в основных параметрах или во всех параметрах, но для этого нужно время. В большинстве случаев предельно допустимые напряжение, ток и мощность, и верхняя граничная частота транзистора будут иметь решающее значение, но иногда, хотя и редко, можно применить

только транзистор, указанный в схеме. На протяжении этой книги вы много раз встретитесь с транзисторами и, надеюсь, у вас появится некоторый опыт, который поможет вам в общении со справочником.

Возвращаясь к свойствам транзистора, отметим, что ток базы, измеряемый прибором Pr1, равен 91.7 мкА, а ток коллектора (Pr2) 9.17 мА. Результаты измерений в программе Qucs можно выводить в табличном виде, и приведенные данные я взял из таблицы. Отношение тока коллектора к току базы равно 100. Это и есть статический коэффициент усиления, который на схеме (рядом с транзистором) обозначен как Bf. Если изменить величину резистора R1, то ток базы изменится, а это приведет к изменению тока коллектора. Попробуйте изменить значение резистора R1 так, чтобы ток базы изменился незначительно, скажем, стал равен 90 мкА. Новое значение тока коллектора можно использовать для получения еще одного коэффициента – отношение разностей токов коллектора к разности токов базы называют *динамическим коэффициентом усиления* транзистора по току, если приращения токов небольшие.

Что мне в данный момент кажется самым важным? Что изменения тока базы вызывают пропорциональное изменение тока коллектора. При этом ток базы много меньше тока коллектора. То есть, транзистор усиливает изменения тока базы, но сохраняет закон, по которому ток базы меняется. Именно, благодаря этим свойствам, транзистор находит широкое применение в электронике в качестве основы разного рода усилителей, преобразователей и генераторов.

Вы можете убедиться, что те три закона электротехники, о которых мы говорили раньше, так же справедливы и сейчас. Ток от плюса источника питания разветвляется на коллекторный и базовый. Ток, подходящий к узлу ветвления, будет равен сумме токов в ветвях схемы, базовой и коллекторной. Падение напряжения на резисторе R2, складываясь с падением напряжения на транзисторе (между эмиттером и коллектором), даст величину, равную напряжению источника питания. Зная ток базы транзистора, проходящий через резистор R1, мы можем вычислить падение напряжения на этом резисторе и проверить прямым измерением напряжения, что оно почти равно напряжению питания. Если измерить падение напряжения между базой и эмиттером и прибавить его к падению напряжения на R1, то равенство будет точным, хотя это напряжение не очень велико, для кремниевого транзистора оно составит 0.5-0.7 вольт. Мало того, даже небольшие изменения этого напряжения будут вызывать значительное изменение тока базы, а, соответственно, пропорциональные ему изменения тока коллектора. Последнее, в свою очередь, приведет к существенным изменениям напряжения эмиттер-коллектор транзистора. То есть, можно говорить не только об усилении транзистором тока, но, в этой схеме, об усилении транзистором напряжения.

Биполярный транзистор имеет три вывода: базу, коллектор, эмиттер. В зависимости от того, какой из выводов используется в качестве общего, различают схемы включения транзистора с общей базой, с общим эмиттером и с общим коллектором. Все три схемы включения обеспечивают усиление, но при одних способах включения осуществляется усиление по току, тогда как усиления по напряжению нет, в других случаях есть усиление и по напряжению, и по току. Чаще всего применяется включение транзистора с общим эмиттером, как это изображено на рисунке 1.28. Сигнал при таком включении подается на базу-эмиттер, а снимается с выводов эмиттер-коллектор. Общий вывод у входного и выходного сигнала – эмиттер, поэтому и схему включения называют схемой с общим эмиттером.

С помощью резистора R1 (рисунок 1.28) устанавливается рабочая точка транзистора на постоянном токе. Чаще всего этот резистор выбирают таким, чтобы напряжение эмиттер-

коллектор было равно половине напряжения питания. Зачем это делается? Многие полезные сигналы, о которых мы поговорим позже, симметричны относительно горизонтальной оси, как синусоидальный сигнал. Если, усиливая сигнал, мы нарушим эту симметрию, то получим другой сигнал, а это уже не усиление, и чаще всего нам этого не надо, но для получения максимального симметричного сигнала на выходе, начальное состояние выхода (на постоянном токе) тоже желательно иметь симметричным, то есть, напряжение должно быть равно половине напряжения питания.

Я уже говорил, что величина сопротивления зависит от температуры. Включив сопротивление в цепь постоянного тока и измеряя ток, проходящий через сопротивление, мы можем судить о температуре вокруг сопротивления. Изменения тока дадут нам информацию об изменении температуры. Разные материалы имеют разную чувствительность к температуре. Особенно сильно реагируют на температуру полупроводниковые материалы. Это их и полезное в одних случаях, и вредное при других обстоятельствах свойство. Резисторы, которые изготавливают для получения информации о температуре, так и называю терморезисторами.

Какое же вредное влияние оказывает температура на полупроводники?

Если в схеме на рисунке 1.28, которую мы тщательно наладили, получив напряжение эмиттер-коллектор точнехонько равным половине напряжения питания, подвергнуть транзистор воздействию температуры, на макетной плате его можно обдувать феном, то начальная рабочая точка (напряжение на выходе) сместится. Теперь напряжение эмиттер-коллектор, сколько мы потратили труда!, уже не равно половине питающего напряжения. Фен, которым мы разогревали транзистор, дает представление о влиянии внешней температуры на рабочую точку транзистора. Но не надо забывать, что на любом сопротивлении, через которое протекает ток, мы говорили об этом, появляется падение напряжения и рассеивается мощность. Транзистор ведет себя также. На нем тоже рассеивается мощность, и для него тоже существует такой параметр, как допустимая мощность рассеивания. А образующаяся в процессе работы мощность, выделяемая в виде тепла, разогревает транзистор, смещая его рабочую точку. Для стабилизации рабочей точки транзистора применяют специальные схемные решения, о которых мы поговорим, когда будем говорить об усилителях. Эксперименты, например, по измерению влияния температуры можно проводить в программе Qucs. Если заглянуть в свойства транзистора, то можно обнаружить такой параметр, как температура. Изменяя этот параметр, добавив в схему на рисунке 1.28 вольтметр параллельно транзистору, можно увидеть изменение напряжения при изменении температуры.

Вообще, для любителей очень важно делать только то, что интересно. Не нравится вам разрабатывать собственные схемы, нравится повторять готовые, занимайтесь ими, но не забывайте, что далеко не всегда повторение готовой схемы сразу приводит к ожидаемым результатам. Важно хорошо понимать назначение и работу всех элементов схемы, а нет лучшего средства для этого, чем эксперименты с этими элементами. Часть из них лучше провести с паяльником в руках. Без этого не обойтись. Но подобные эксперименты, порой, требуют хорошего оснащения любительской лаборатории. Далеко не все могут позволить себе покупку всех необходимых приборов. Часть можно изготовить самостоятельно, но их настройка тоже требует наличия приборов, и образуется замкнутый круг, из которого трудно выбраться. В этом смысле компьютерные программы, подобные PSIM и Qucs оказываются очень полезны. Проводя ряд экспериментов за компьютером, можно найти то место, где опыты можно перенести на макетную плату, а работа будет обеспечена теми приборами, что есть в распоряжении любителя. Со временем парк приборов пополнится, а круг интересов и возможностей значительно расширится. Но и в этом случае не стоит пренебрегать теми возможностями и теми удобствами, что дают компьютер и программы.

Глава 2. Программы и схемы

Пришло время рассказать о том, как пользоваться программами, о которых речь шла выше. Полный рассказ займет слишком много места, но многое в рассказе можно опустить, если вы работали хотя бы с одной программой на компьютере. Многого о программах я не знаю сам. О некоторых возможностях программ, я думаю, не подозревают и их создатели. Поэтому задача этой главы дать представление о том, как начать делать что-то полезное, используя эти программы. Каждая из программ САПР (EDA) может иметь свою специализацию и лучше выполнять те задачи, для которых предназначалась. Так программа PSIM (www.powersimtech.com) предназначена к анализу силовых электронных устройств, устройств управления двигателями и симуляции динамических систем. В этом ее специфика, это она делает лучше, чем другие программы, но, видимо, есть то, что она не делает или делает не лучшим образом. Начать работу с ней можно почти сразу, а по мере накопления опыта можно решать все более и более сложные задачи. Программа Qucs, имя которой дает аббревиатура английского названия – почти идеальный симулятор цепей – хорошо работает в широком диапазоне задач, но не захватывает работу с микроконтроллерами. Программа KTechlab позволяет проверить аналоговые и цифровые схемы и создать программу для микроконтроллера. Однако в ней трудно исследовать детали многих процессов, легко поддающихся исследованию в других программах. Вообще, выбор удобной для себя программы, это еще одна область любительских опытов, интересных, с одной стороны, без привязки к собственно электронике, а, с другой стороны, обогащающих знаниями по предмету. В отличие от формального тестирования программ, которое больше ориентировано на проверку руководства к программе, на соответствие технического задания реализации проекта, исследование возможностей программы по реализации собственных интересов предполагает выбор конкретных схем, выбор конкретных вопросов, на которые хотелось бы получить ответы, продумывания такой модификации схемы, которая наиболее ярко подчеркивала бы суть задачи, а это, «как ни крути», прямой путь к пониманию существа работы схемы со всеми сопутствующими деталями этой работы. Полезное и нужное в практике, при правильном подходе оно доставляет и удовольствие, которого мы всегда ждем от предмета своего увлечения.

PSIM в Linux

Я не люблю пересказывать только назначение разделов меню программы или диалогов настроек, не потому, что это не интересно или бесполезно, нет, но для этого есть руководство пользователя, которое написано либо автором программы, либо в тесном контакте с автором, лучше других знающим программу. Поэтому, рассказывая о программах, я постараюсь продолжить рассказ об электронике. Тем более, что о программе Qucs я рассказывал в книге «Наглядная электроника», сама программа имеет встроенное руководство по быстрому началу работы с ней (на русском языке) в разделе *Справка* основного меню, а на моем сайте есть конспект книги об этой программе. А PSIM, если учитывать то, что это коммерческая программа, при покупке будет снабжена всеми необходимыми примерами и описаниями. Демонстрационная версия, которую я использую, обязательно должна иметь ограничения, выявление которых отдельная задача, требующая приложения сил. В простых случаях эти ограничения могут остаться не замеченными, в более сложных затруднить понимание происходящего. В этом отношении я отдаю предпочтение покупке программы, если могу себе это позволить, либо использование некоммерческих программ.

Эту главу я начинаю с рассказа о программе PSIM, но ставлю задачу рассказать об электрических схемах. Схема, это то, с чем любителю придется сталкиваться очень часто.

Сущность любой электрической схемы проста – в наиболее компактной форме передать как можно больше информации о работе электрического устройства. Любая схема, как правило, некоторое графическое представление процесса или устройства. Математические формулы, например, в этом смысле можно отнести к схемам, где каждому элементу математической схемы дается свое графическое представление, а процесс описывается с помощью графических изображений операций над элементами схемы. Как и электрическую схему, математическую можно описать обычными словами. Но и электрическая схема, и математическое выражение, описанные обычными словами, сложнее воспринимаются, чем графика, и допускают вольную трактовку, построенную на восприятии текста. И хотя к любой электрической схеме желательно иметь описание работы устройства, сделать такое описание, ссылаясь на схему, гораздо проще, чем без нее.

Как и программы, электрические схемы имеют некоторую специализацию. Радио схема может сильно отличаться от схемы электропроводки в доме. Кроме специализации графические различия возникают в силу отсутствия единых стандартов во всем мире, в силу того, что стандарты меняются со временем, да и в силу того, что стандарты часто носят рекомендательный характер, не обязательный к применению. Пример разного графического изображения на электрических схемах таких широко распространенных элементов, как резистор и конденсатор, можно обнаружить в любой программе САПР (EDA), достаточно взглянуть на резистор программы PSIM, изображенный в виде ломаной линии, и резистор программы Qucs – прямоугольник, или полистать отечественные схемы за разные годы, где иногда первое обозначение относилось к проволочным резисторам, а второе, положим, к объемным. Однако эти различия несколько скрадываются буквенным обозначением «R» и пояснениями в тексте, описывающем работу схемы. Могут быть отличия в рисунке конденсатора, одна из пластин которого может изображаться в виде дуги, но и здесь буквенное обозначение остается «C». В разные годы электролитические конденсаторы (конденсаторы большой емкости, выполненные с применением электролитов) в нашей стране изображались так, что одна из пластин, положительного вывода, а электролитические конденсаторы полярные, то есть, им не все равно, к какому из полюсов батарейки они подключены, была контурной, потом обе пластины стали изображать сплошными, а у положительной ставить плюс. А уж микросхемы могут иметь и еще больше вариантов графического представления и буквенного обозначения. Но, несмотря на это, схема, даже с отличиями в графике, дает возможность быстро понять очень многое, достаточно небольшого опыта в чтении этих схем.

Напомню те элементы любой схемы, которые упоминались выше: источник питания (батарейка), сопротивление, конденсатор, индуктивность, диод, транзистор, проводник. Я затрудняюсь определить, какое количество схем, сотни, тысячи или сотни тысяч, можно изобразить, используя только эти элементы. Что делают эти элементы в схемах?

Есть много полезных схем, построенных только на пассивных элементах: сопротивление, конденсатор, индуктивность. Это могут быть разного рода делители напряжения постоянного и переменного тока, это могут быть разные фильтры, а если обратиться к электротехнике, то лампы накаливания (своеобразные сопротивления) по сегодняшний день освещают наши дома.

Делители напряжения чаще применяются в схемах измерения и используются при регулировании напряжения. Самым простым представителем делителя напряжения является композиция из двух последовательно включенных резисторов. Как они работают рассмотрим в программе PSIM, работу с которой начнем тоже с самого начала.

Для пользователей операционной системы Windows установка программы сводится к двойному щелчку мышкой по файлу setup.exe с последующими подтверждениями согласия с

лицензией, с местом расположения программы и с созданием ярлычков запуска программы на рабочем столе и в меню. Пользователи Linux предварительно должны установить, если это не сделано, программу Wine. Это эмулятор работы Windows программ, создание которой, если я не ошибаюсь, ставило своей задачей поддержать работу компьютерных игр, написанных для Windows, в среде Linux. Поэтому далеко не все программы операционной системы Windows работают с эмулятором. Но те, что работают, работают достаточно хорошо, чтобы не делать особой разницы между ними и программами, созданными для работы с Linux.

На моем компьютере установлены две ОС Linux, в одной программа работает, а во второй нет даже программы Wine. Чтобы подробнее описать работу с PSIM, я хочу перейти в этом месте к дистрибутиву Ubuntu, установить (его пока нет) эмулятор Wine, установить программу PSIM, и описать с ее помощью работу делителя напряжения. Перезагружаюсь!..

Вот я и в Ubuntu, дистрибутиве, который в силу моего любопытства в данный момент находится в тестовом виде, то есть, все программы могут работать с ошибками. Но мне было интересно взглянуть на будущий дистрибутив, и я установил тестовую версию. Надеюсь, это не помешает выполнить задуманное.

В дистрибутиве Ubuntu для установки программ можно воспользоваться двумя средствами, одно из которых – это штатная программа установки и удаления программ, расположенная в разделе *Приложения* основного меню.

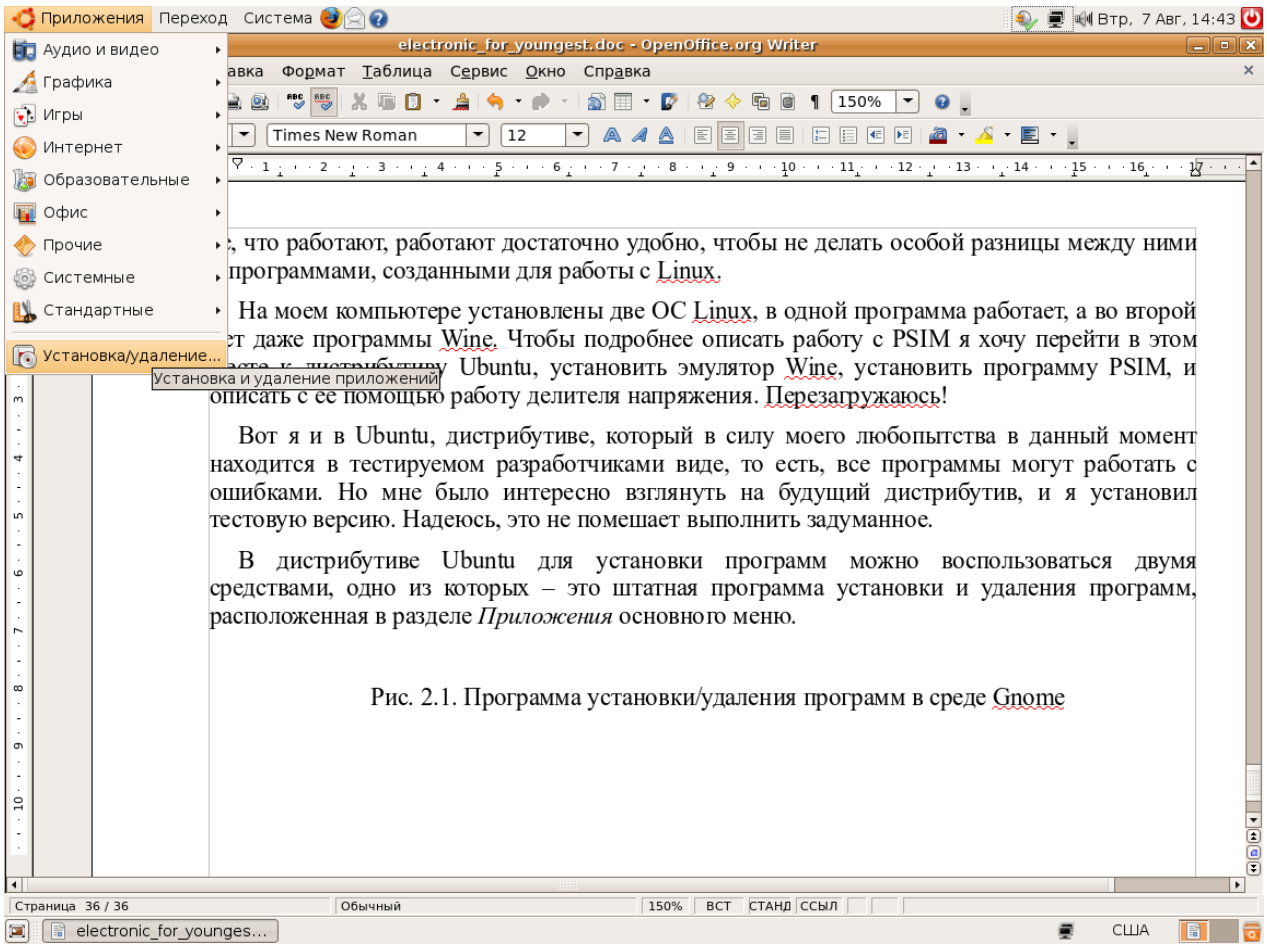


Рис. 2.1. Программа установки/удаления программ в среде Gnome

Попробую воспользоваться этой программой для установки Wine. После первого запуска и

попытки найти приложение пришлось расширить поиск, изменив параметр поиска в окне *Показать*, что в верхней части программы установки приложений. Если бы поиск не увенчался успехом, я мог бы использовать вторую возможность – менеджер пакетов Synaptic. Но пока этого не требуется, все нашлось, и достаточно отметить Wine в окне найденных программ, чтобы эмулятор был установлен. Конечно, я использую подключение к Интернету, иначе это было бы сложнее сделать, поскольку дистрибутив Ubuntu имеет минимальный размер, что подразумевает минимальное количество приложений. Есть все необходимое, но ничего сверх того.

Рис. 2.2. Установка приложения Wine в дистрибутиве Ubuntu

Как можно видеть в окне диалога установки приложений, в левой его части, дистрибутив позволяет обратиться к десятку разделов, каждый из которых выводит в правой части множество доступных для установки программ. Если нужная программа не находится, а ее можно поискать вводом имени программы в окно *Поиск*, то можно использовать менеджер пакетов. Установка приложения Wine заняла несколько минут. Гораздо больше времени у меня уходит на поиски архивированной версии PSIM (psim7.1.1_demo.zip) на моем компьютере. Остается скопировать пакет на рабочий стол и распаковать его, с чем вполне справляется штатная программа работы с архиваторами Linux. Далее установка может происходить с некоторыми отличиями в разных случаях. В Fedora 7 достаточно было, если я ничего не путаю, дважды щелкнуть мышкой по полученному после распаковки файлу Setup.exe. Но это, видимо, связано с тем, что я уже проделал запуск конфигурации Wine, о котором сейчас забыл. После запуска конфигурации Wine Configuration, расположившейся в разделе *Система-Параметры* основного меню оболочки Gnome, в диалоговом окне которой я выбираю Windows XP в качестве прототипа, в моей домашней папке появляется скрытый файл .Wine (файл или папка с предшествующей точкой). Чтобы его увидеть в домашней папке, в менеджере файлов Nautilus следует установить опцию *Показывать скрытые файлы* в разделе *Вид* основного меню.

Я несколько раз щелкаю по файлу установки, безрезультатно. Можно еще попробовать перезагрузить систему, что мне совсем не хочется делать, но для чистоты эксперимента я это сделаю.

Увы, перезагрузка системы не помогла, пойдем другим путем. Откроем скрытую папку .Wine в домашнем каталоге, и поместим программу Setup.exe в раздел drive_c (он

автоматически создан программой Wine), процедура не обязательная, но так удобнее. Теперь открываем основное меню оболочки Gnome в разделе *Приложения-Стандартные* (в разных дистрибутивах это может находится в разных разделах, но меню можно изменить вручную), находим Wine File и запускаем эту программу, которая есть не что иное, как менеджер файлов Windows.

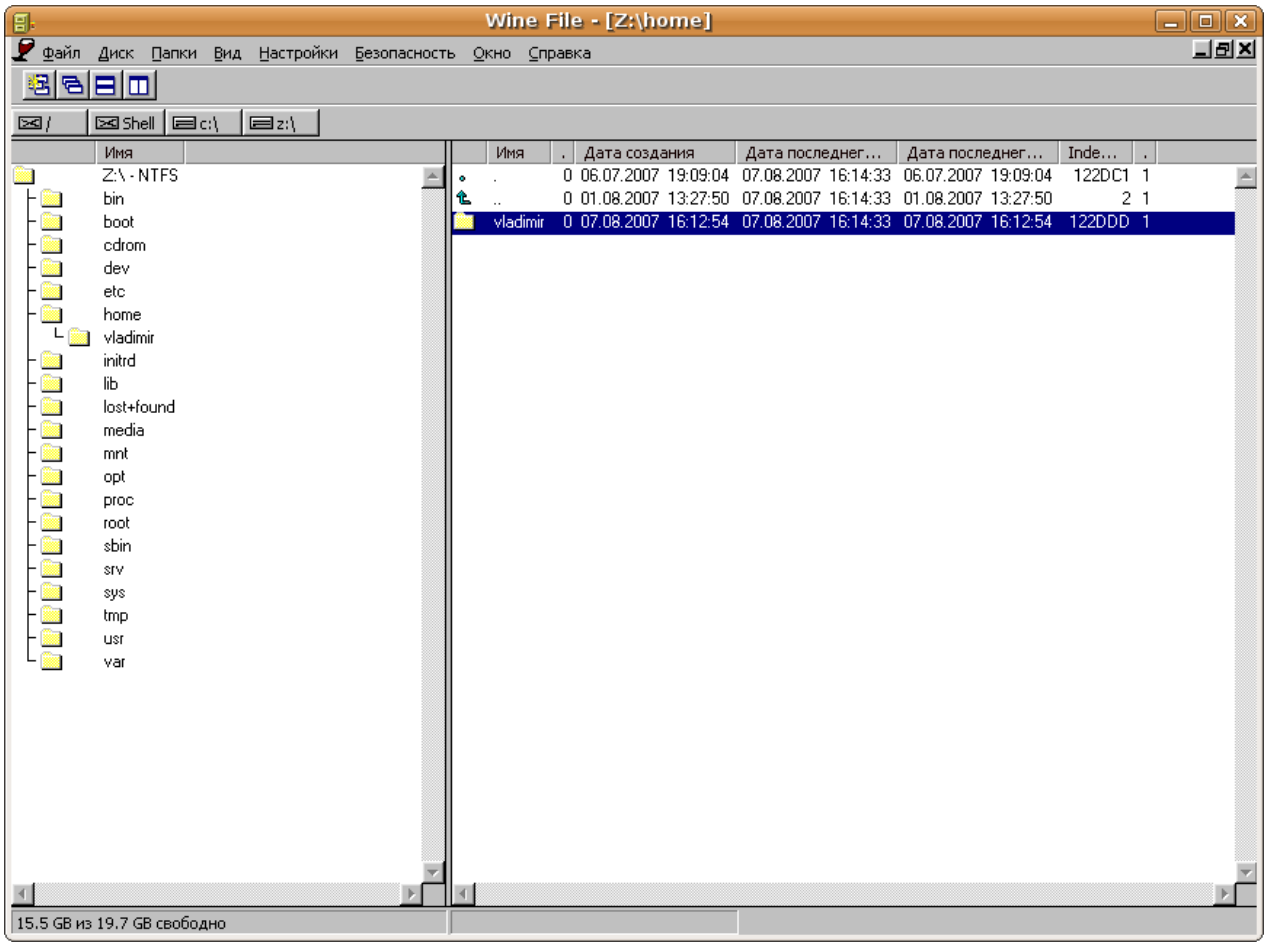


Рис. 2.3. Менеджер файлов Windows в оболочке Gnome операционной системы Linux

Щелкая привычным (и в Windows, и в Linix) образом мышкой по диску C:, находим файл Setup.exe, который теперь запускается на установку двойным щелчком по нему. Я так давно не пользовался Windows, что не помню, вид менеджера файлов на рисунке 2.3 в этой операционной системе соответствует какой ее версии. Возможно, Windows 95, или более ранней? Но это не суть важно – папки открываются как обычно, переход на уровень вверх с помощью стрелки над папками или с помощью «дерева папок» в левом окне позволяет найти все необходимое.

Структура диска «C:» тоже привычна пользователям Windows. Папки открываются двойным щелчком мышки, так же запускаются программы из менеджера файлов.

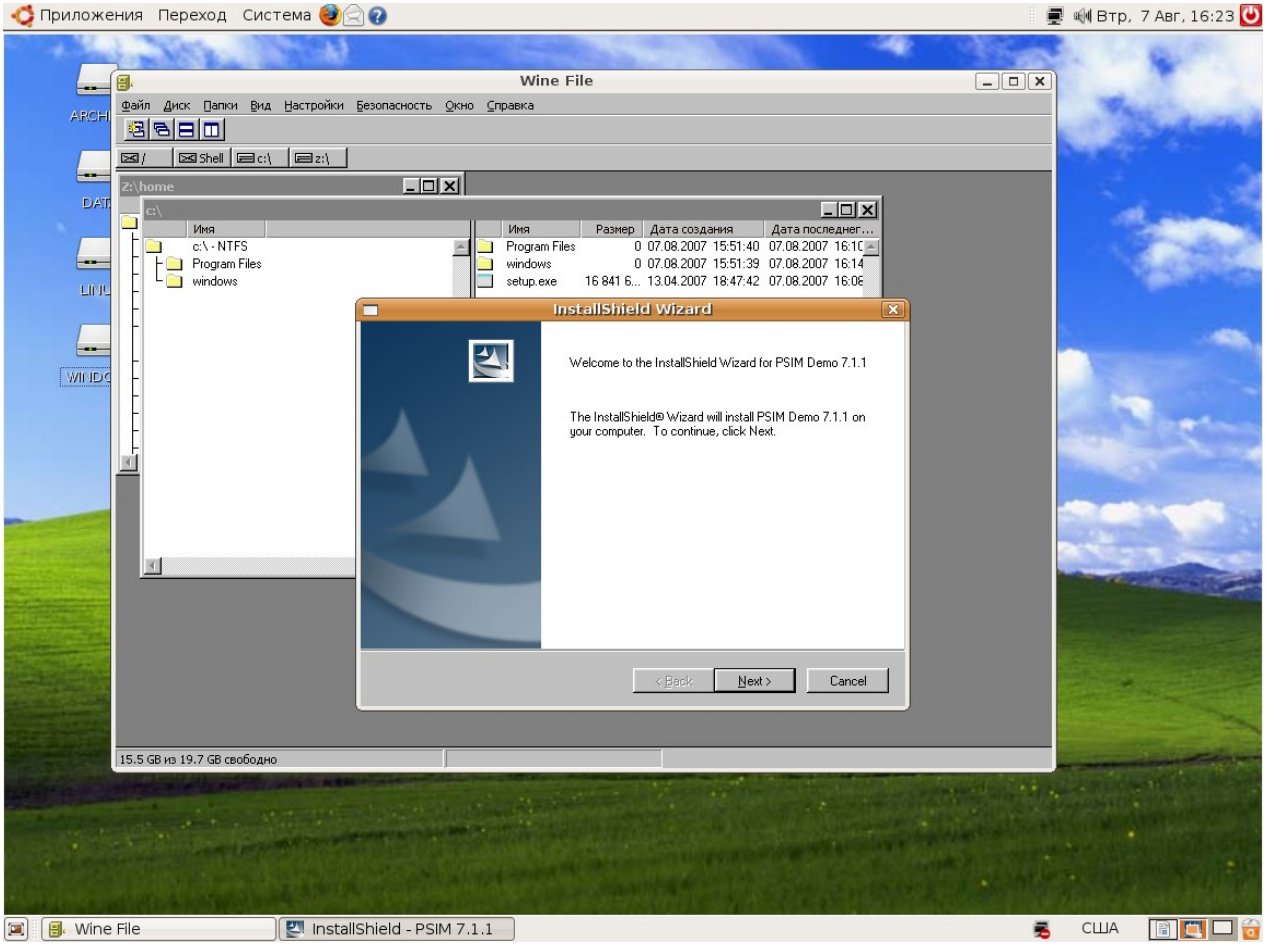


Рис. 2.4. Установка программы Windows в среде Linux

Запустив установку, остается только щелкать по клавишам **Next>**, **Yes** и **OK**, что я делаю настолько быстро, что умудряюсь прозевать сообщение о не найденной при установке программе. Можно попытаться переустановить программу, можно почитать документацию, но попробую разобраться с этим позже.

После установки программы в подразделе *Прочие* раздела *Приложения* основного меню Gnome появляется раздел запуска программы PSIM. Запуск не отличается чем-то особенным от запуска других программ, да и вид рабочего окна программы я не стал бы определять, как особенный. Без чтения документации самое простое и очевидное начало работы – это либо с помощью основного меню в разделе *File* выбрать *New*, либо сделать то же с помощью инструментальной панели меню, нажав крайнюю слева кнопку с иконкой чистого листа.

С появлением рабочей области в нижней части окна появляется второе инструментальное меню, относящееся к компонентам программы. Их можно использовать, нажимая соответствующие клавиши с помощью левой клавиши мышки, затем перемещая мышкой курсор на рабочее поле схемы, где повторный щелчок мышки оставляет нужный компонент. Возможно многократное размещение этого компонента повторным нажатием левой клавиши мышки. Если повторное размещение компонента не нужно, можно нажать клавишу **Esc** на клавиатуре. Нажатием правой клавиши мышки, при перемещении курсора с «привязанным» к нему компонентом по рабочему полю, можно повернуть компонент – каждое нажатие поворачивает его на 90 градусов.

Гораздо больше схемных элементов, чем на инструментальной панели, можно найти, если воспользоваться разделом *Elements* основного меню. Выпадающее меню содержит много

разделов, имеющих собственные подменю, и все необходимые компоненты для работы с силовой электроникой.

Я всегда считал, и остаюсь при своем мнении, что создавая собственную схему (или рисуя схему), следует располагать элементы схемы таким образом, чтобы схема была удобна в работе именно вам. Это позже можно задуматься о более плотном размещении всего в рабочем поле чертежа, а начинать работу лучше с максимальным для себя комфортом. Конечно, если конечной целью вашей деятельности будет переход к профессиональной работе, то следует сразу приобретать профессиональные привычки, о которых лучше всего узнать у профессионалов в той области, которая вас привлекает. Вместе с тем я уверен, то, как рисовать схемы, вы освоите гораздо быстрее, чем поймете работу самих устройств, а, значит, не будет большой беды, если для понимания электроники вы будете рисовать такие схемы, с которыми вам легче работать.

Итак, я говорил о делителе напряжения, который мы изобразим в только что установленной программе PSIM.

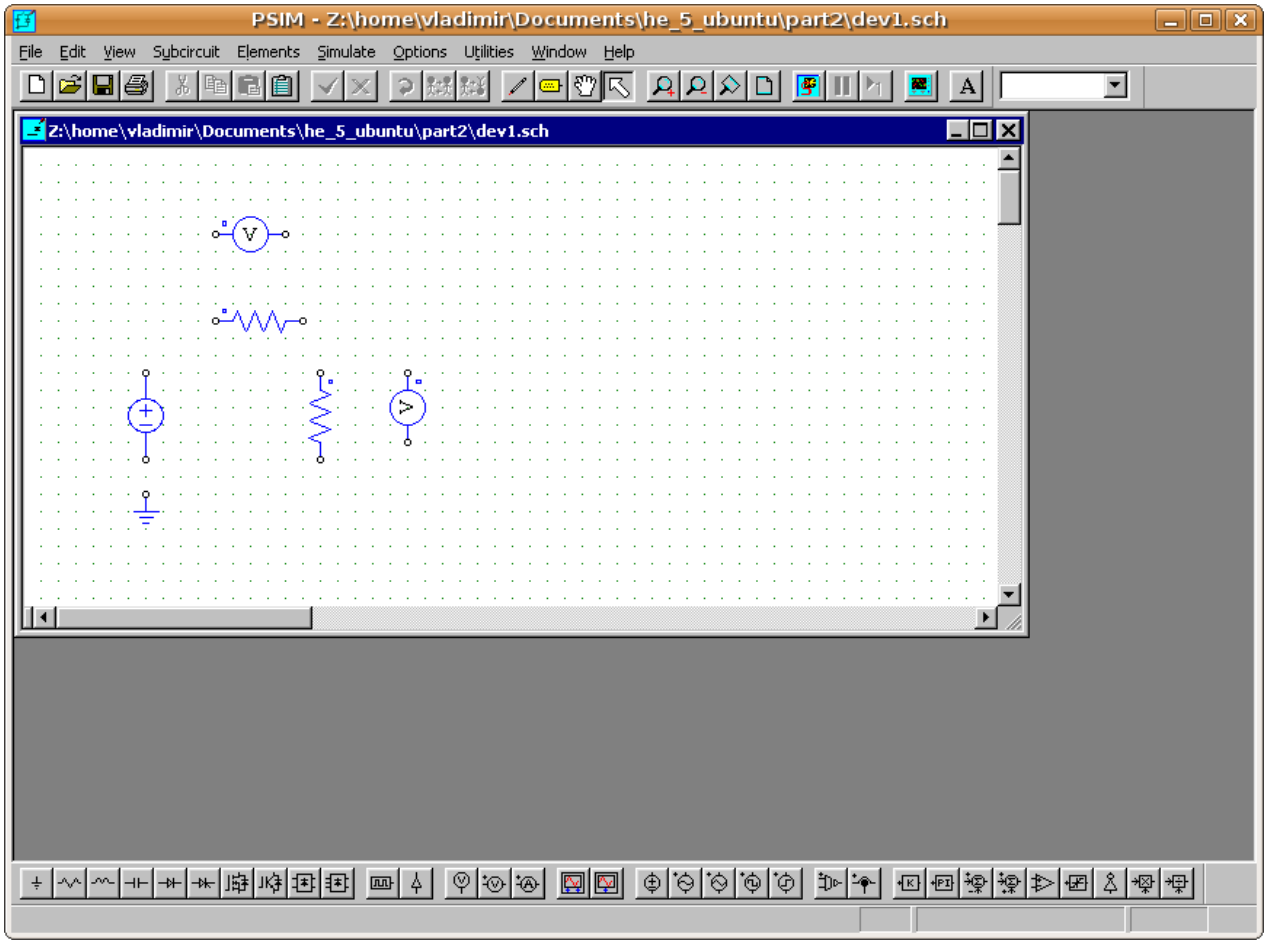


Рис. 2.5. Делитель напряжения, размещение элементов схемы

В программе PSIM после размещения элементов они остаются не обозначенными, пока вы не установите, что должно быть обозначено на схеме. Но, прежде чем ввести обозначения, можно соединить элементы схемы. Для этого следует воспользоваться подразделом *Wire* раздела *Edit* основного меню. А можно использовать верхнее инструментальное меню, где нажать клавишу, которая будет нажата на рисунке ниже, иконка на ней похожа на карандаш, а соединение проводится с помощью мышки. Нажав на левую клавишу мышки в месте,

обозначенном кружочком, у одного конца элемента, следует провести линию к кружочку у конца другого элемента, где левую клавишу мышки отпустить. Останется линия соединения двух элементов схемы, соответствующая проводнику в реальной схеме.

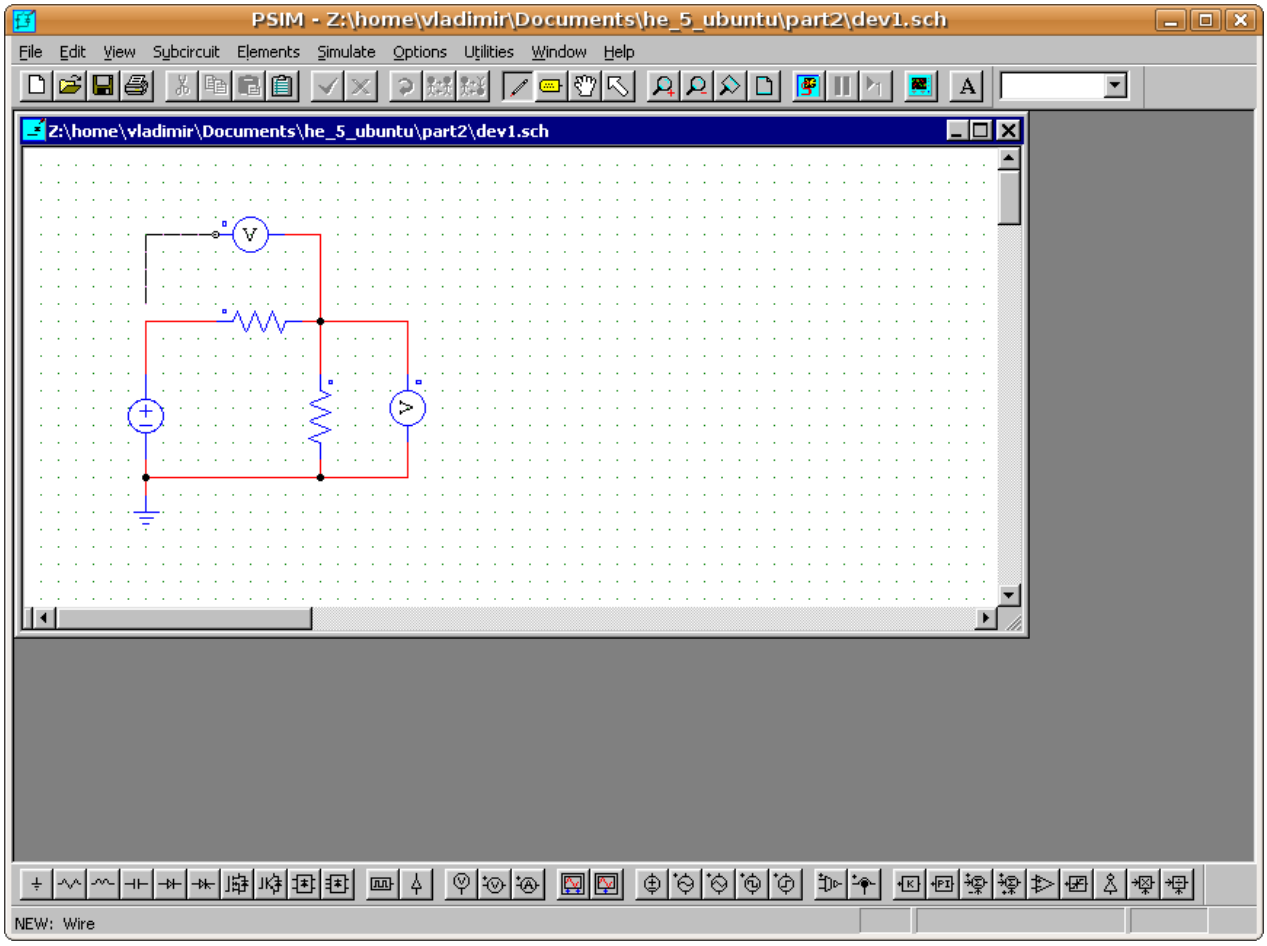


Рис. 2.6. Проведение соединений в программе PSIM

Теперь можно задать величину, скажем, напряжения источника питания и сопротивления резисторов. Для этого вначале нажмем на клавишу **Esc** на клавиатуре или на клавишу инструментального меню правее клавиши соединений с иконкой в виде стрелки. Курсор приобретает обычный вид, а мы переходим к режиму выбора в редакторе программы. Теперь, если щелкнуть правой клавишей мышки, когда курсор находится над выбранным нами компонентом, то появится выпадающее меню, в котором есть раздел *Attributes*. В этом месте тех, кто работает в Linux, я хочу предупредить – у меня, когда курсор смещается чуть ниже на следующий раздел выпадающего меню, программа зависает, некоторое время остается зависшей, а затем пропадает. Я стараюсь почаще сохранять сделанную работу, а при задании параметров элементов аккуратнее управлять курсором, чтобы не попадать на «заминированный» участок меню. Вина ли в этом зависании программы Wine, или причина иная, не знаю. Я не собираюсь долго работать с программой, которая предназначена для другой операционной системы, а если задержусь с расставанием, то потерплю некоторые неудобства. Итак, выпадающее меню.

В выпадающем меню следует щелкнуть по разделу *Attributes*, чтобы попасть в диалоговое окно задания этих атрибутов элемента электрической схемы.

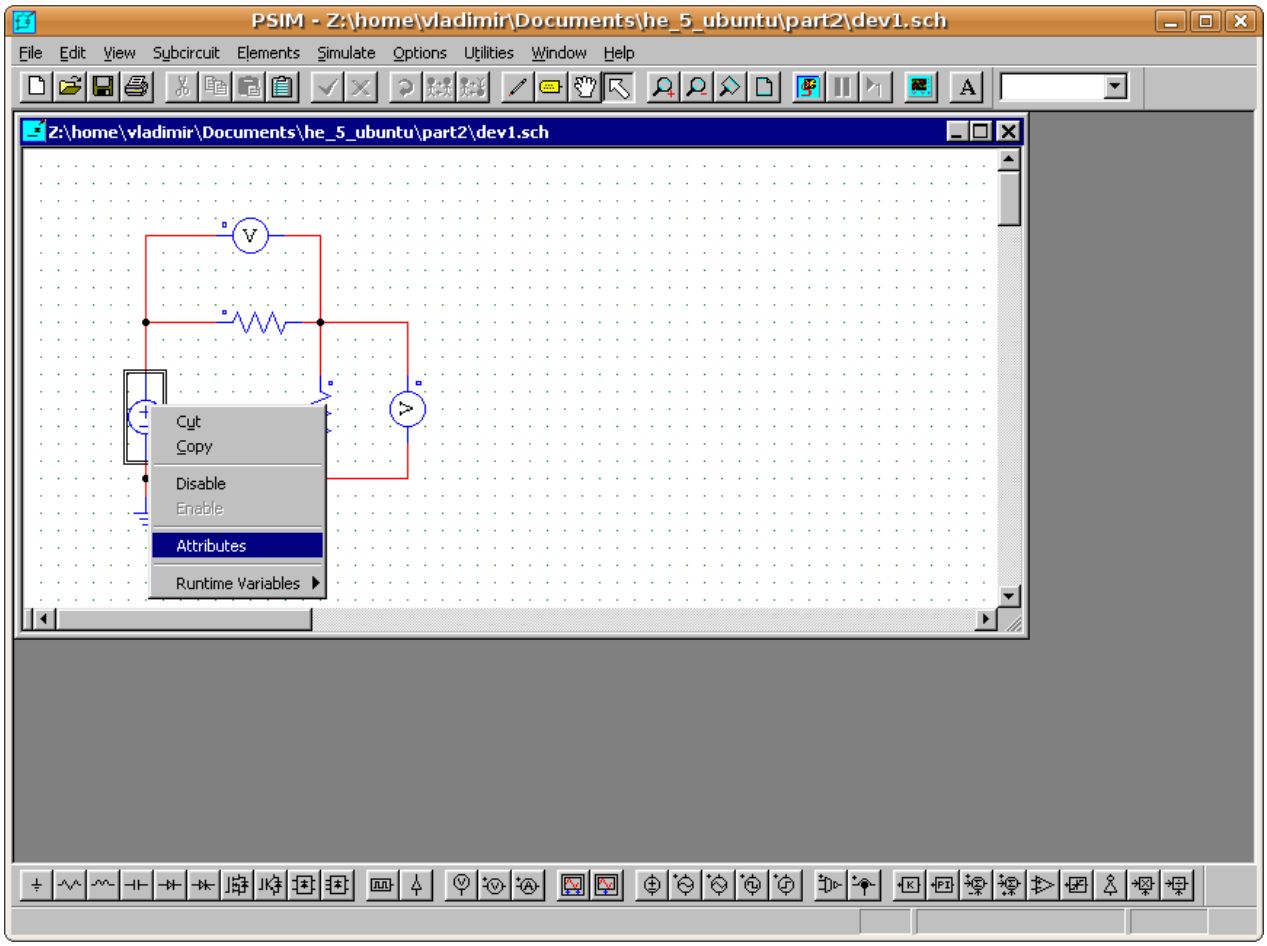


Рис. 2.7. Выпадающее меню PSIM для задания значений элементов схемы

Может быть, удобнее даже использовать для этого не выпадающее меню, а соответствующий подраздел раздела *Edit* основного меню, там ошибиться и «подвесить» программу труднее. Это как вы найдете удобней для вас, но использовав любой из вариантов, вы открываете диалоговое окно задания параметров.

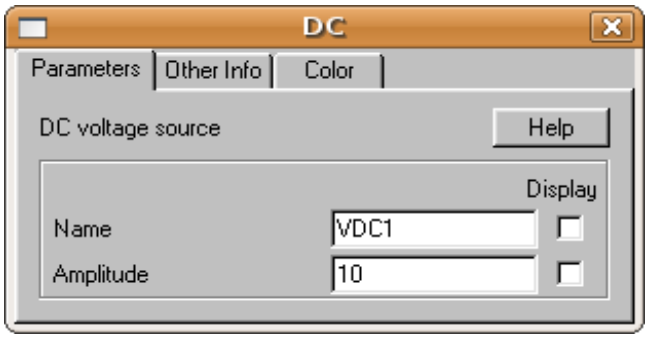


Рис. 2.8. Диалоговое окно задания атрибутов источника питания

В окнах Name и Amplitude для источника питания можно задать, соответственно, имя источника питания, а в окошках под надписью Display установить галочки для тех параметров, которые должны отображаться на схеме. Они появляются сразу. Задав все необходимое, можно закрыть диалоговое окно обычным образом.

Завершив все соединения, обозначив все элементы схемы и задав их величину, можно

начать симуляцию либо с помощью инструментального меню, либо через основное меню (*Simulate-Run Simulation*). Процесс симуляции отображается в нижней части основного окна на панели состояния в виде обычной шкалы прогресса, как при загрузке файлов или сохранении файлов, а по завершении у меня в левом верхнем углу рабочего окна появляется почти полностью свернутый обозреватель результатов (*SimView)*. Его окно можно «растащить» с помощью мышки, как любое окно, и просмотреть результаты симуляции в обозревателе. Для этого следует открыть файл данных симуляции с расширением .smv, который создается в той же папке, где сохранена схема с расширением .sch. Открывается файл, как и любой файл, в основном меню в разделе *File* обозревателя, либо клавишей с иконкой открытой папки инструментального меню. Открывается файл не сразу, вначале вы попадаете в диалоговое окно выбора файла, где перемещаетесь обычным образом, как при работе с любой программой при открывании файла, а после выбора файла вы попадете в меню выбора функции, которую следует отобразить.

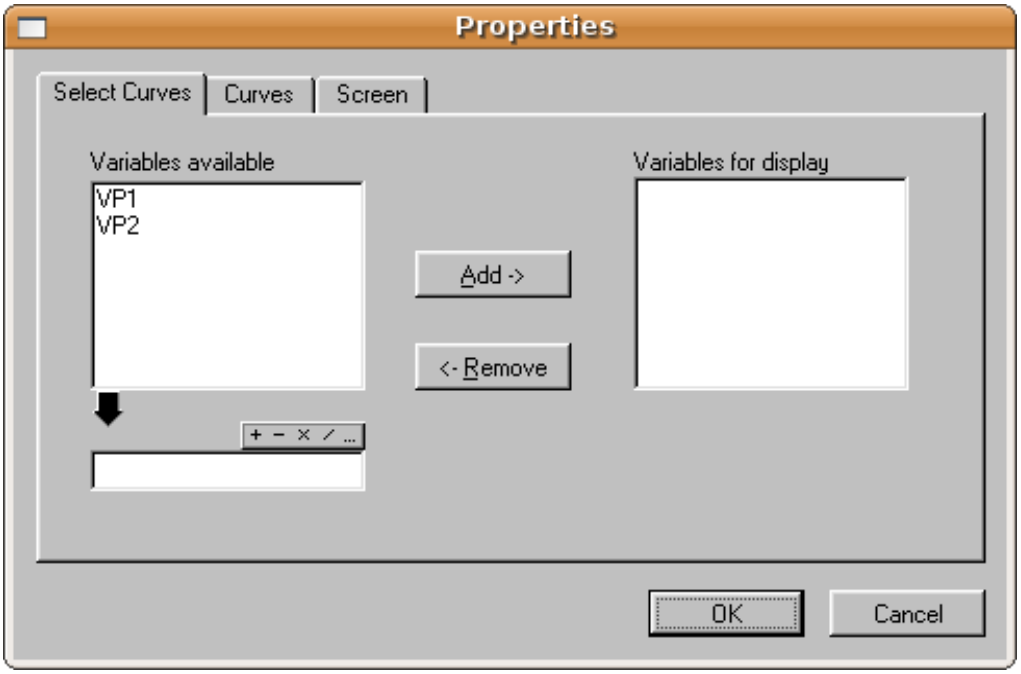


Рис. 2.9. Диалоговое окно выбора функции, отображающей результаты симуляции

В данном случае есть два измерительных прибора – вольтметры VP1 и VP2. Щелкнув по первому из них, а затем по клавишам **Add** -> и **OK**, можно получить результаты измерений первым вольтметром. А повторив всю процедуру для другого вольтметра, получить результаты измерения вторым вольтметром. Оба графика располагаются в окне обозревателя удобным образом, если перетащить их или воспользоваться средствами раздела *Window* основного меню, позволяющими расположить окна каскадом или «черепицей», возможность присутствующая, практически, в любом редакторе.

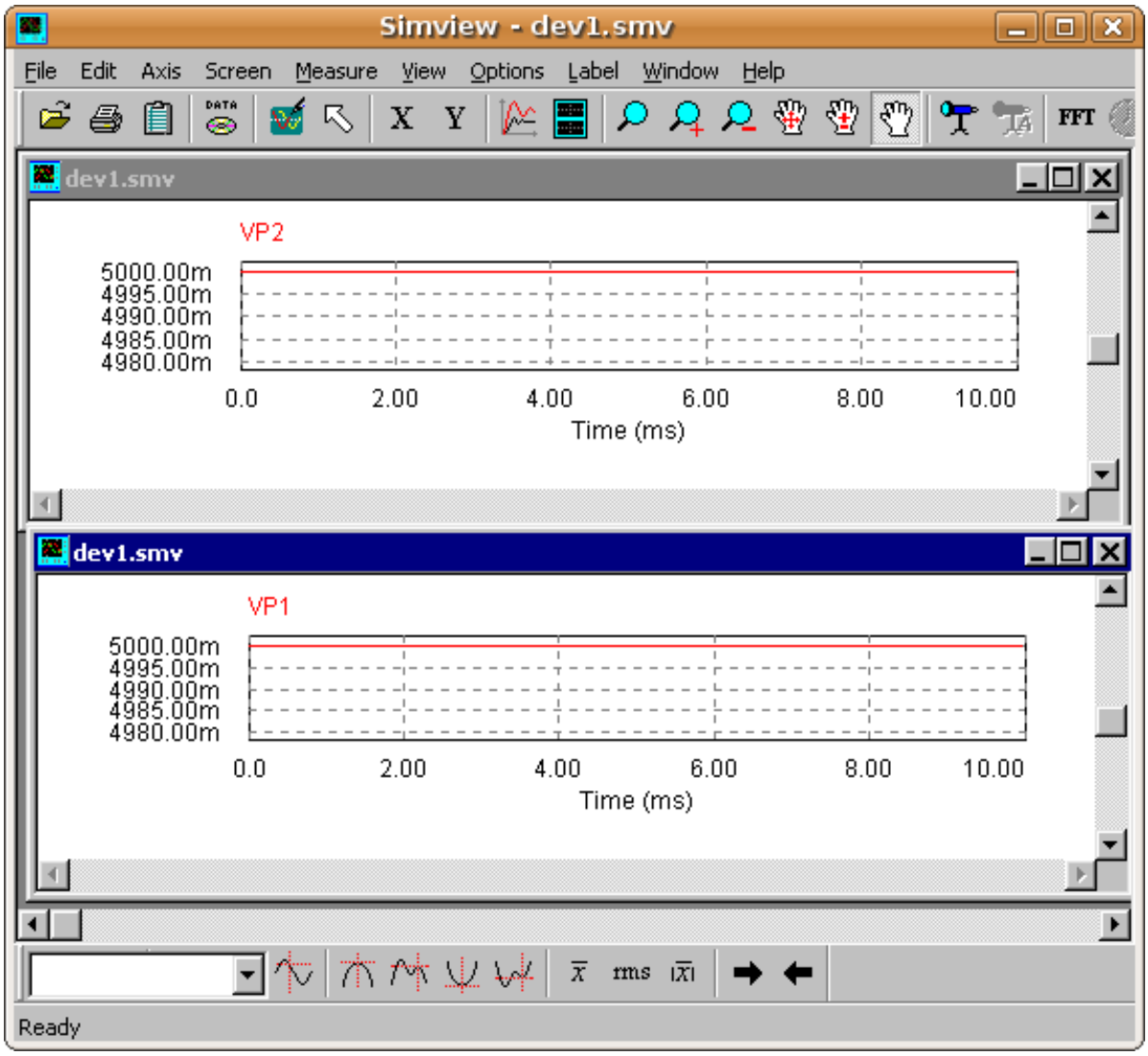


Рис. 2.10. Результаты измерений в окне обозревателя

Как и следовало ожидать, а мы уже знаем законы Ома и Кирхгофа, ЭДС источника поровну распределяется в виде падения напряжения на двух резисторах равной величины. То есть, 10 вольт источника питания *делятся* на два напряжения по 5 вольт.

Если резисторы имеют разное сопротивление, то можно, прибегнув к расчетам, определить эти два напряжения. В первую очередь определим общее сопротивление цепи, сложив сопротивление резисторов R1 и R2. Разделим ЭДС на это сопротивление, получив ток, протекающий в цепи, а затем умножим этот ток на величину сопротивления R1, получив напряжение на нем, затем на величину R2 для получения второго напряжения.

Есть и другая возможность увидеть эти результаты – не использовать обозреватель, а воспользоваться основным меню программы, где в разделе *Simulate* есть подраздел *Runtime Graphs*, открывающий подменю всех полученных функции при симуляции. В этом случае окно программы выглядит так:

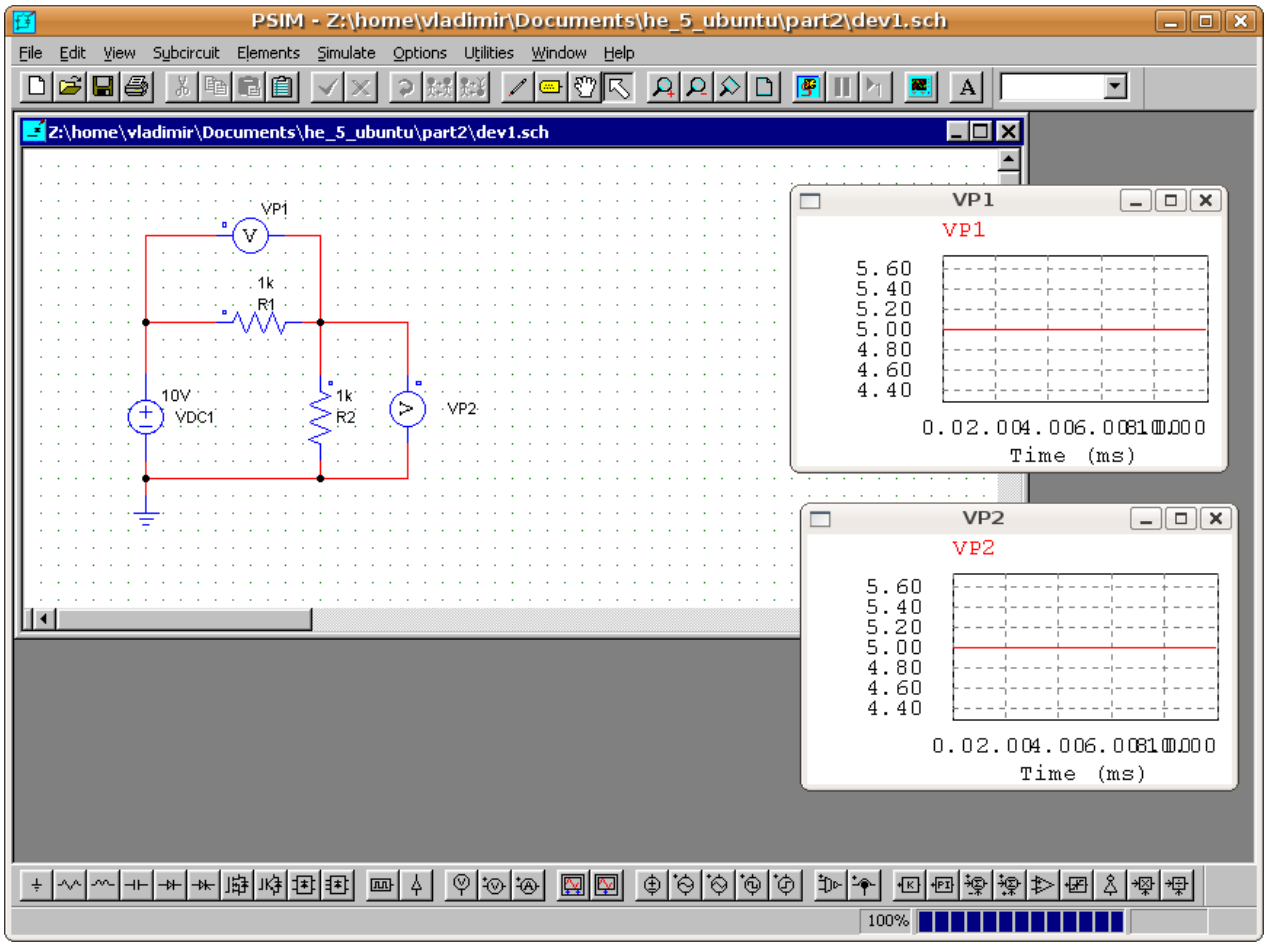


Рис. 2.11. Вывод результатов измерений в основном окне программы

В измерительных схемах делитель напряжения применяют для расширения пределов измерений. Самый простой вид решение этой задачи будет иметь при делении напряжения на 10. Положим для определенности, что вольтметр имеет предел измерения 1 В, и что мы хотим иметь три предела: 1 В, 10 В и 100 В, а сопротивление вольтметра 100 кОм.

На схеме ниже я вместо переключения пределов нарисую три вольтметра. Источник питания возьмем 100 В, а делитель напряжения построим из резисторов R1-R3. Когда измеряемое напряжение минимально, то есть, 1 В, вольтметр (VP1) включен параллельно делителю напряжения. Чтобы схема измерения не слишком ухудшала параметры вольтметра, желательно иметь результирующее сопротивление, определяемое параллельным включением сопротивления вольтметра и делителя, как можно больше. Но на следующем пределе измерения, 10 В, вольтметр (VP2) оказывается включен параллельно сопротивлениям R2-R3 и последовательно с резистором R1. Чтобы в этом случае не проводить сложных вычислений, возьмем суммарное сопротивление резисторов R2 и R3 на порядок меньше сопротивления вольтметра, тогда влияние сопротивления вольтметра на делитель напряжения будет приемлемым. Таким образом, выберем суммарное сопротивление резисторов R2 и R3 равным 10 кОм.

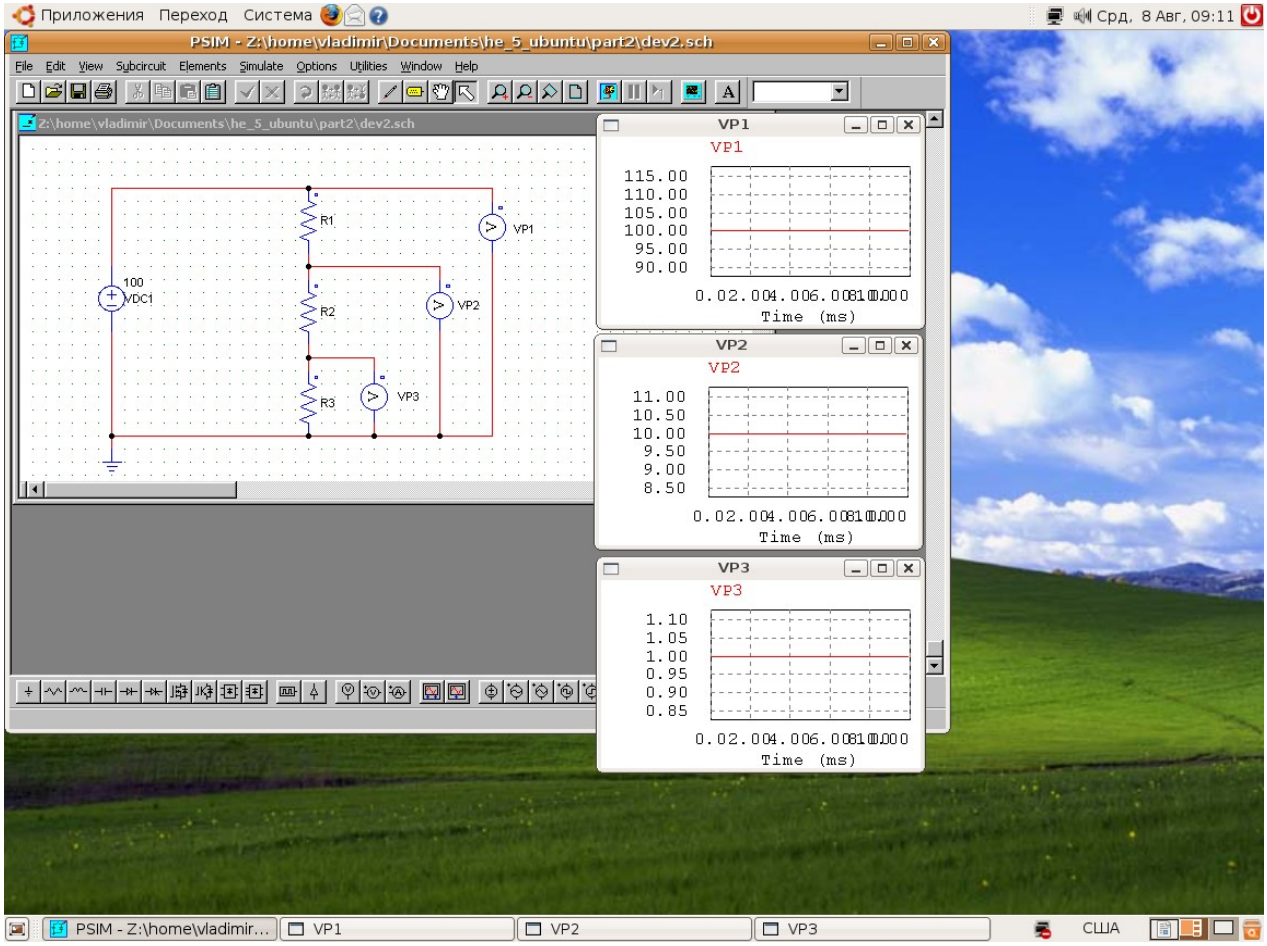


Рис. 2.12. Делитель напряжения для трех пределов измерения

Определить все значения сопротивлений делителя, если не учитывать влияние сопротивления вольтметра, можно так: суммарное сопротивление резисторов R2 и R3, как мы решили, 10 кОм, а напряжение на них 10 В, в этом случае через них протекает ток 1 мА. При токе в 1 мА на сопротивлении R3 должно быть падение напряжения 1 В, что дает сопротивление 1 кОм. Значит сопротивление R2=10 кОм – 1 кОм, то есть, 9 кОм. А сопротивление R1 при общем сопротивлении делителя 100 кОм определится разностью R1=100 кОм -10 кОм = 90 кОм. Именно такие значения сопротивления я использовал в схеме на рисунке 2.12. Методику расчета делителя напряжения для расширения пределов измерений вольтметра я советую вам посмотреть в учебнике или книге, посвященной измерениям, а сейчас отметим одно – *напряжение, снимаемое с нижнего плеча делителя так относится к полному напряжению на делителе, как сопротивление этого нижнего плеча делителя к полному сопротивлению делителя*.

На принципе деления напряжения основана работа регуляторов напряжения. Так регулятор громкости усилителя, называемый еще потенциометром, конструктивно представляет собой подковообразное сопротивление, по поверхности которого движется контакт, связанный с ручкой регулировки. Входное напряжение подается на все сопротивление, а выходное снимается с части этого сопротивления, образуя делитель напряжения. Схемы делителя, представленные выше, примеры очень простой электрической цепи, но применяемой очень часто. А цель, которую я преследовал, немного рассказать о том, как установить и использовать программу PSIM в среде Linux. Даже демонстрационная версия этой программы после установки имеет раздел документации, в котором подробно

описаны все компоненты электрических схем, с которыми может работать программа, и рассказано, как с ней работать.

Qucs в Linux

Теперь я хочу немного рассказать о другой программе, работающей и в Linux и в Windows, о программе Qucs.

Программа Qucs, как многие современные среды разработки, поддерживает такое понятие как проект. Для начала работы можно использовать и просто создание файла со схемой, используя раздел основного меню *Файл*, но лучше создать новый проект, выбрав раздел *Проект* и подраздел *Новый проект...* Чем это лучше?

Кроме схемы, а проект может иметь и не одну схему, можно позволить себе хранить вместе со схемой план работы, рабочий журнал и документацию, описывающую схему. Эти дополнительные файлы, как любые бумажные работы, могут вызвать уныние не только у любителя, не всегда их любят и профессионалы, но поверьте, что пройдет немного времени, и, возвращаясь к прежней схеме, и обнаружив эти дополнительные файлы, вы порадуетесь своей предусмотрительности. Без рабочего журнала и подробного описания схемы по прошествии времени трудно вспомнить не только то, как ты пришел к нужному решению, но, порой, и каково это решение. Разбираться с тем, как работает твоя собственная схема, еще грустнее, чем разбираться с чужой схемой. Эти рекомендации – не плод теории, но собственной практики, прошедшей во многом под флагом разгильдяйства. Оправдания можно найти всегда, однако они слабое утешение, когда, вглядываясь в каракули, сделанные на клочке бумаги, ты пытаешься понять, о чем же ты думал, когда увековечил все в таком таинственном виде...

Итак, запускаем программу... я опять вернулся в свой основной дистрибутив Fedora 7, где после очередных экспериментов с операционной системой вход в графический менеджер Gnome сломался, а тяга к новизне перед этим занесла меня в другой менеджер окон KDE (или графическую оболочку системы), и в результате этих событий я использую KDE. Напомню, что любой дистрибутив Linux может работать с несколькими (можно и почти одновременно) графическими оболочками: Gnome, KDE, Xface и т.д. Их, вероятно, правильнее называть оконными менеджерами, поскольку меняется не только вид окон, но и панелей, меню, обозревателей и т.д. Некоторые из этих менеджеров очень похожи, такими их и задумывали, на графическую оболочку Windows, другие совсем на нее не похожи и внешне, и поведением. Если вас, как и меня, временами тянет сменить обстановку, то, используя разные графические менеджеры Linux, вы сможете совершать путешествия в новые неизведанные страны вашего компьютера, хотя каждый путешественник знает, что любая страна – это некоторое пространство, заполненное природой вперемежку с городами и людьми. С компьютером тоже самое.

Итак, запускаем программу Qucs и создаем новый проект, что начинается с выбора имени для нового проекта. Назовем его «amplifier (усилитель)», поскольку активные элементы схемы, в первую очередь транзисторы, чаще всего используются для построения разного рода усилителей и генераторов, для разных целей, с разными свойствами, но объединяемых одним свойством, связанным с усилением сигналов. Мне хотелось бы показать, хотя это и не совсем так, что транзистор в усилителе можно, и подчас это удобно, в первом приближении рассматривать, как управляемый делитель напряжения.

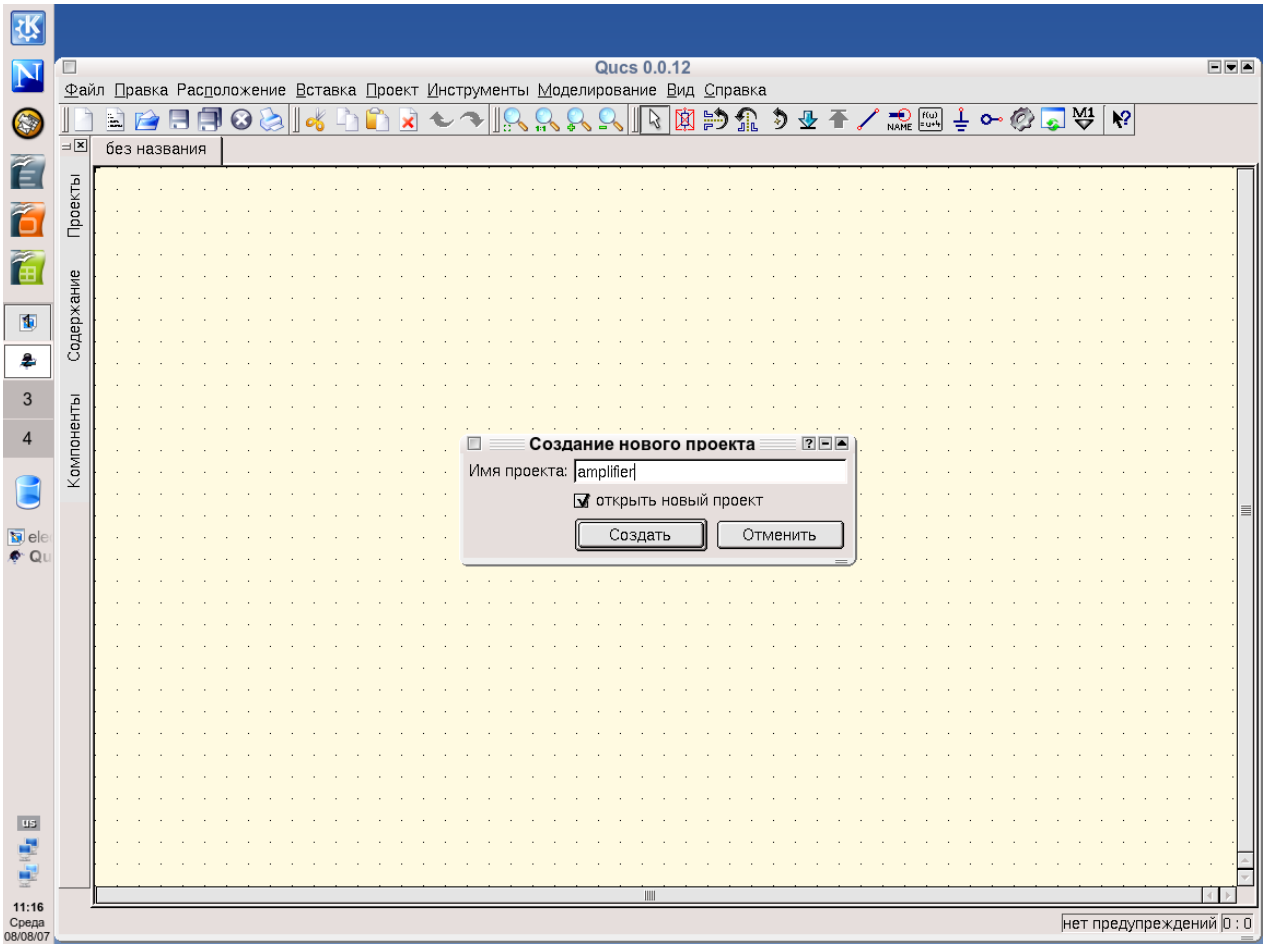


Рис. 2.13. Создание нового проекта в программе Qucs

Нажимаем на клавишу **Создать**, и получаем новенький проект. Рабочее поле для рисования схемы открывается сразу, и чтобы не забыть сделать это позже, я сразу сохраню файл, назвав его ampl1.sch, воспользовавшись разделом основного меню *Файл* и его подразделом *Сохранить как...*, где в открывающемся диалоговом окне есть возможность выбрать место расположения файлов, либо в созданной программой скрытой папке с именем программы, либо там, где мне удобнее сохранять проект. Диалоговое окно выглядит и ведет себя как при сохранении обычного текстового файла. Если проект сохраняется в скрытой папке программы, то файл можно добавить в проект с помощью раздела *Проект* основного меню, где есть подраздел добавления файлов в проект. Если вы хотите сохранить проект и его файлы в другом месте, то можно перенести созданный программой проект в другое место, разместив в папке проекта нужные файлы. В любом случае файл будет хранится в папке, содержащей все необходимые файлы. Если открыть проект, имеющий файлы, то в окне менеджера проекта (слева, закладка *Содержание*) будет дерево файлов проекта, классифицированных по их назначению. Я прежде не использовал эту программу для создания проектов и мне интересно, как я смогу добавить файлы плана работ и рабочего журнала. Для их создания можно попытаться использовать два варианта – простой, но мощный, редактор Gedit и встроенный редактор текста. Какой из вариантов будет работать, я еще не знаю...

Собственно, получается так, что удобнее создать файл в программе Qucs. Заходим в раздел *Файл-Новый текст,* создаем файл, который сохраняем с расширением * (выбранным в окошке вида файла), что не мешает ему сохраниться с расширением .vhdl, затем

переименовываем его, удалив расширение. Для этого в дереве проекта выделяем этот файл (предварительно закрыв его), щелкаем правой клавишей мышки, в выпадающем меню выбираем *Переименовать*, удаляем все, кроме имени файла. При следующем открывании проекта (а не файла) в дереве проекта все созданные файлы переместятся из раздела «VHDL» в раздел «Другие». Так я создаю файлы plan, work_book, doc. Их можно открыть встроенным в программу редактором и работать с ними, можно открыть вне программы с помощью редактора Gedit и не менее успешно работать с ними.

А я открываю пока пустой файл схемы, чтобы нарисовать транзисторный усилитель, где для удобства пояснений использую транзистор типа p-n-p. Добавить компоненты можно выбрав (слева) закладку *Компоненты*. В верхней части этого окна есть раскрывающееся окно выбора компонент: дискретные компоненты, источники, измерители и т.д. На рисунке открыт раздел «дискретные компоненты», правее которого две стрелки, открывающие меню.

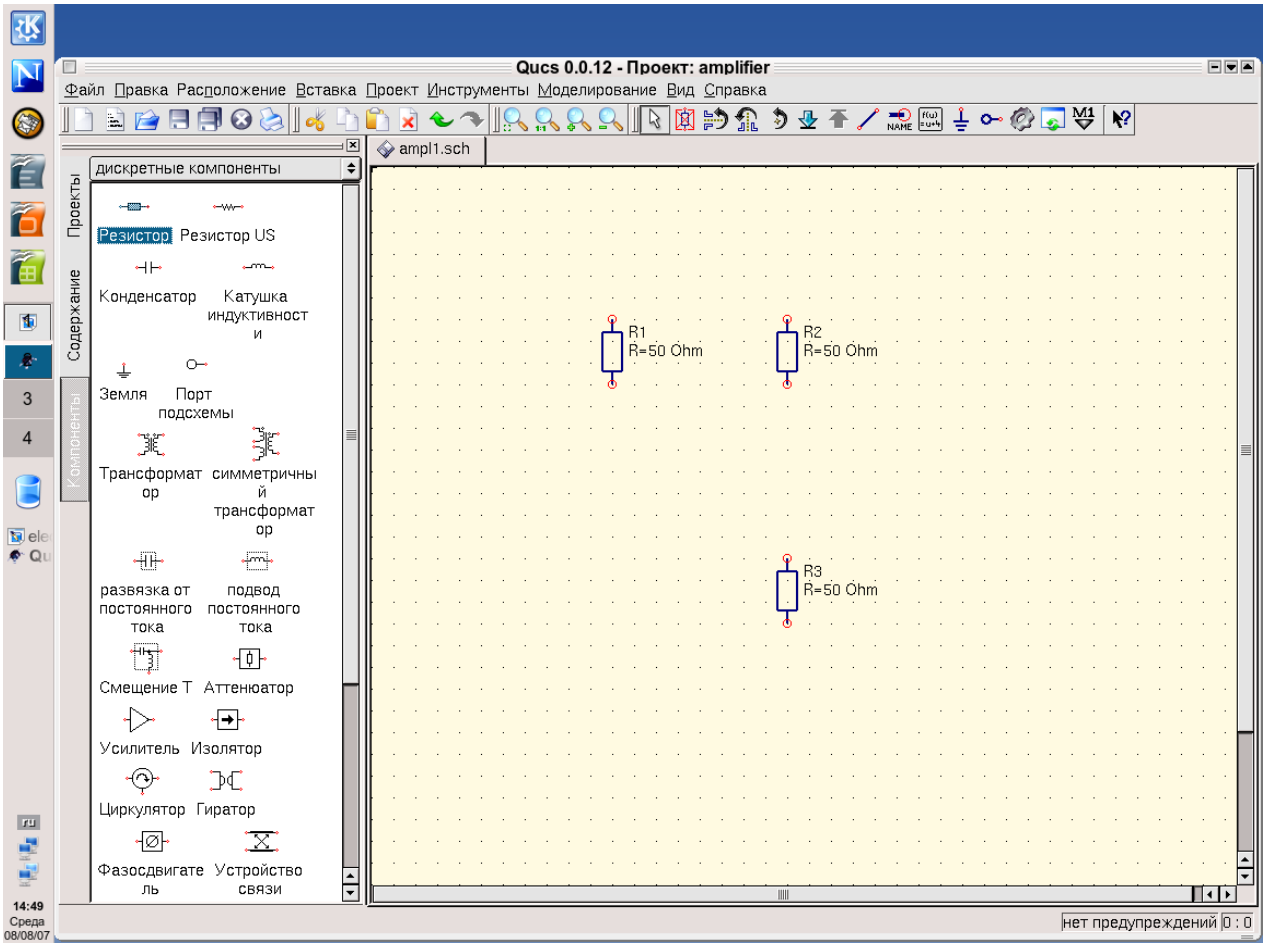


Рис. 2.14. Выбор компонентов в программе Qucs

Щелкнув по элементу схемы слева, его можно перенести на рабочее поле чертежа. Как и в программе PSIM щелчок правой клавишей мышки при переносе поворачивает элемент, и можно последовательно добавить нужное количество этих элементов или нажатием клавиши **Esc** на клавиатуре отказаться от этого. Также щелчком правой клавиши мышки по элементу схемы можно открыть выпадающее меню с пунктом *Изменить свойства*, вызывающим диалоговое окно изменения свойств. В нем можно изменить все параметры элементов, не следует забывать только использовать латинскую раскладку клавиатуры и нажимать клавишу **Применить**.

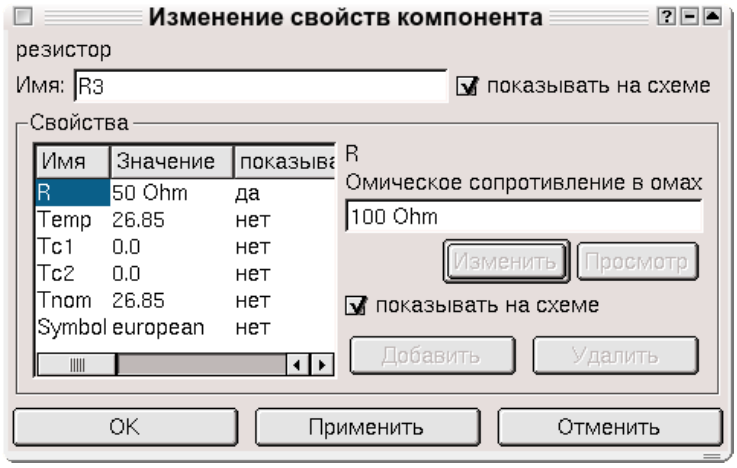


Рис. 2.15. Диалоговое окно свойств резистора

Соединения элементов схемы можно провести с помощью раздела основного меню *Вставка-Проводник*, или нажав на клавишу инструментального меню с изображением проводника. При проведении соединений левую клавишу мышки удерживать не надо, достаточно щелкнуть по отправной точке и щелкнуть по точке «прибытия».

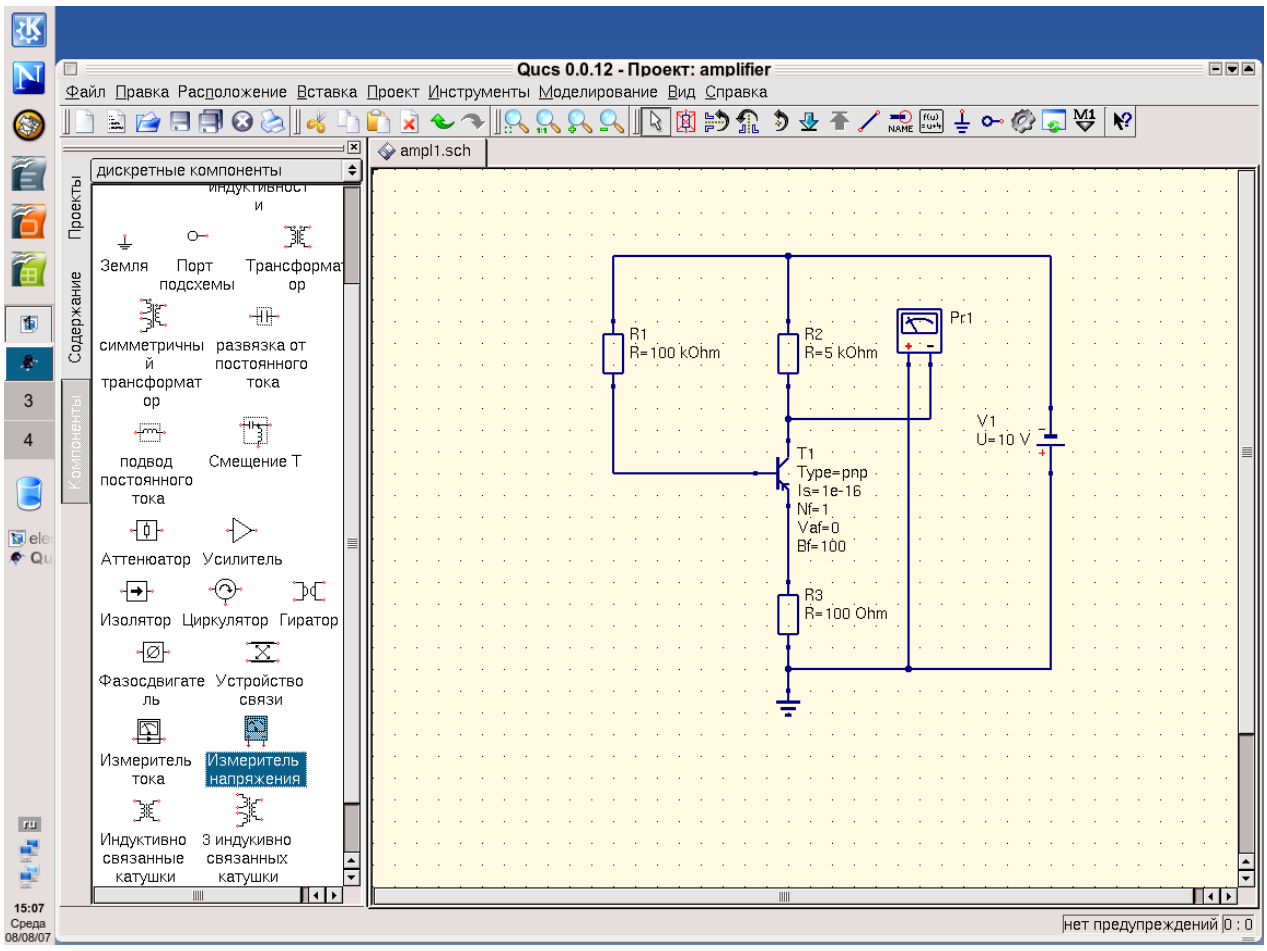


Рис. 2.16. Схема включения транзистора типа p-n-p

Почему я выбрал транзистор именно типа p-n-p? Легче объяснить, что ток эмиттера в транзисторе разветвляется на ток базы и ток коллектора, поскольку за техническое

направление тока, напомню, принято направление от плюса к минусу. Токи, проходя по резисторам R1 и R2, вновь сливаются в один ток, возвращающийся в источник питания. Здесь все в соответствии с законом Кирхгофа. А если вспомнить, что ток базы и ток коллектора транзистора связаны соотношением Iк=Bст*Iб, то есть, ток коллектора равен произведению тока базы на статический коэффициент усиления по току, имеющий значение порядка несколько десятков единиц, то становится ясно, что ток коллектора почти равен току эмиттера, поскольку ток базы невелик.

Каким образом, если вспомнить о подаче напряжения на переход база-эмиттер, о котором я говорил, этот ток базы получается в схеме на рисунке 2.16? Переход база-эмиттер транзистора ведет себя подобно диоду в прямом включении, то есть, пропускает ток от источника ЭДС V1 через резистор R1, при этом в грубом приближении мы можем его определить по закону Ома, разделив 10 В на 100 кОм. Более точный расчет должен включать падение напряжения на переходе база-эмиттер (порядка 0.5-0.7 В для кремниевого транзистора) и падение напряжения на резисторе R3, которыми мы пока пренебрегаем. Ток базы 0.1 мА при коэффициенте усиления транзистора равном 100 превратится в 10 мА тока коллектора. А этот ток, протекая по резистору R2, должен вызывать на нем падение напряжения равное 50 В (5000 Ом * 0.01 А), что, похоже, никак не может иметь место.

Добавив к схеме *Моделирование на постоянном токе* из состава компонент (левое окно, меню выбора компонент, раздел *Виды моделирования*), запустим это самое моделирование. После моделирования открывается новое рабочее поле, на которое можно разместить график или таблицу, которые выбираются в левом окне, открывающемся на разделе *Диаграммы*. Перенесем табличную форму отображения результатов моделирования (из *Диаграмм*) в новую рабочую область, а в открывшемся диалоговом окне дважды щелкнем по Pr1.V под надписью *Набор данных* и нажмем клавиши **Применить** и **ОК**. В полученной таблице измеритель напряжения покажет напряжение эмиттер-коллектор транзистора равное 0.291 В. На резисторе R2, конечно, не 50 В, но, практически, все напряжение источника питания. Мне отчего-то кажется, что следует уменьшить ток базы, увеличив резистор R1, что приведет к уменьшению тока коллектора. Хотелось бы получить падение напряжения эмиттер-коллектор транзистора (напомню, что эмиттер рисуется со стрелочкой) близкое к половине напряжения питания. Попробуем увеличить резистор R1 в десять раз, то есть, сделаем его равным 1000 кОм (1 МОм). Действительно, теперь моделирование показывает, что напряжение эмиттер-коллектор, измеряемое нашим *измерителем напряжения*, равно 5.43 В.

Если в этот момент забыть, что имеешь дело с транзистором, в коллекторную цепь которого включен резистор, если представить себе, что к резистору R2 подключен тоже резистор, назовем его Rтран, то мы получим своеобразный делитель напряжения. Базовый ток определяет величину резистора Rтран, и, если базовый ток будет переменным, изменяющимся по некоторому закону, то величина этого резистора будет меняться по тому же закону, а, следовательно, делитель напряжения будет делить напряжение источника питания, повторяя закон изменения базового тока. Существенные изменения базового тока происходят при небольших изменениях напряжения эмиттер-база, что обусловлено свойствами p-n перехода в прямом включении и конструкцией транзистора. А в десятки раз большее изменение коллекторного тока позволяет получить большие изменения напряжения эмиттер-коллектор. Возможно, модель транзистора, как управляемого резистора, легче воспринимать, тем более, что есть схемы, где транзистор используется именно в качестве управляемого резистора.

Подкрепим эти соображения экспериментом в программе Qucs. Добавим к схеме, изображенной выше, источник переменного (синусоидального) напряжения на вход и посмотрим осциллограмму на выходе (коллектор транзистора и общий провод).

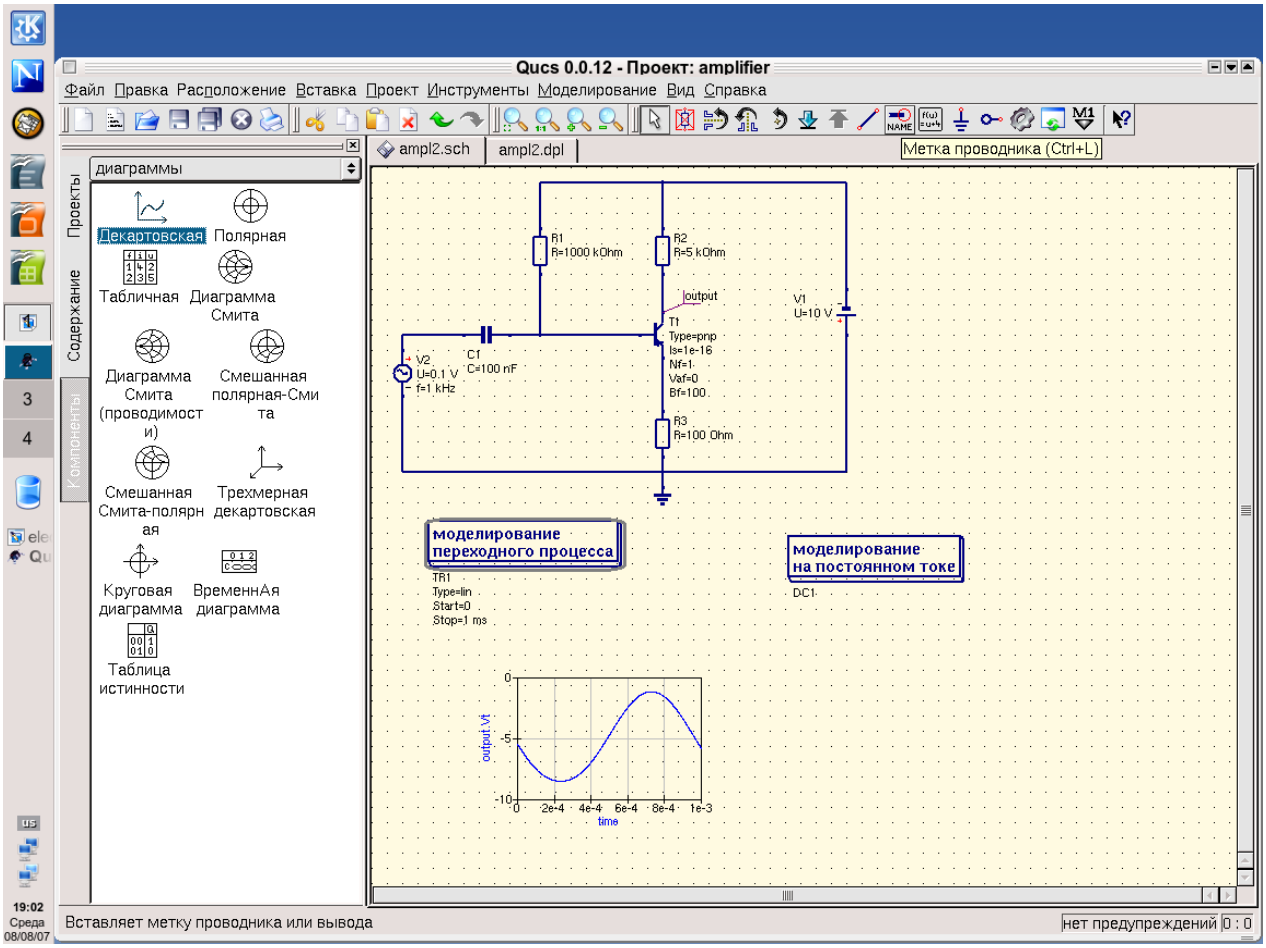


Рис. 2.17. Усиление переменного напряжения транзистором

Чтобы добавить источник переменного напряжения, на вкладке компонент в окне слева следует найти раздел *источники*, выбрать *источник напряжения переменного тока*, перенести его в рабочую область редактора схем и изменить его свойства, используя выпадающее меню и правую клавишу мышки: зададим напряжение 0.1 В и частоту 1 кГц. Кроме того, следует добавить моделирование переходных процессов из раздела *виды моделирования*, и изменить свойства моделирования, увеличив количество шагов до 1000. Затем следует на коллектор транзистора добавить метку (с помощью клавиши инструментального меню с надписью *name* я обозначил метку как *output*. Надпись под нужной клавишей инструментальной панели видна на рисунке выше. После моделирования следует добавить *Декартовскую* диаграмму, где выбрать (двойным щелчком мышки) метку *output*.

Как видно из диаграммы, выходное напряжение синусоидальное, то есть, закон изменения напряжения сохраняется, а его амплитуда (размах напряжения от середины до максимума) близка к 5 вольтам. Входной сигнал имеет амплитуду 0.1 вольта. Отношение выходного напряжения ко входному — берутся либо действующие значения, либо пиковые — это отношение есть не что иное, как коэффициент усиления по напряжению.

Усилитель, как любой компонент электрической схемы, характеризуется своими параметрами. Наиболее часто нас будет интересовать его входное сопротивление, коэффициент усиления, полоса рабочих частот, выходное сопротивление и то, насколько усилитель склонен искажать входной сигнал. Почему нас интересует входное сопротивление усилителя – его можно рассчитать по току эмиттера или другими способами, но мы

попробуем получить его иначе, экспериментально — так почему оно нас может интересовать? Усилители обычно работают либо с датчиками (источниками полезного сигнала), либо с другими каскадами усиления, и если входное сопротивление усилителя мало, меньше внутреннего сопротивления источника, то это может повлиять на напряжение источника, подобно тому, как внутреннее сопротивление вольтметра влияет на результаты измерения. Часто входное сопротивление усилителя стараются сделать больше сопротивления источника сигнала.

Как определить входное сопротивление усилителя экспериментально? Если мы последовательно со входом схемы включим резистор, то он со входным сопротивлением усилителя образует делитель напряжения. Зная величину этого добавленного резистора можно определить входное сопротивление усилительного каскада. Еще проще это проделать, добавляя такое сопротивление, при котором напряжение на выходе уменьшится в два раза. В этом случае входное сопротивление будет равно добавленному.

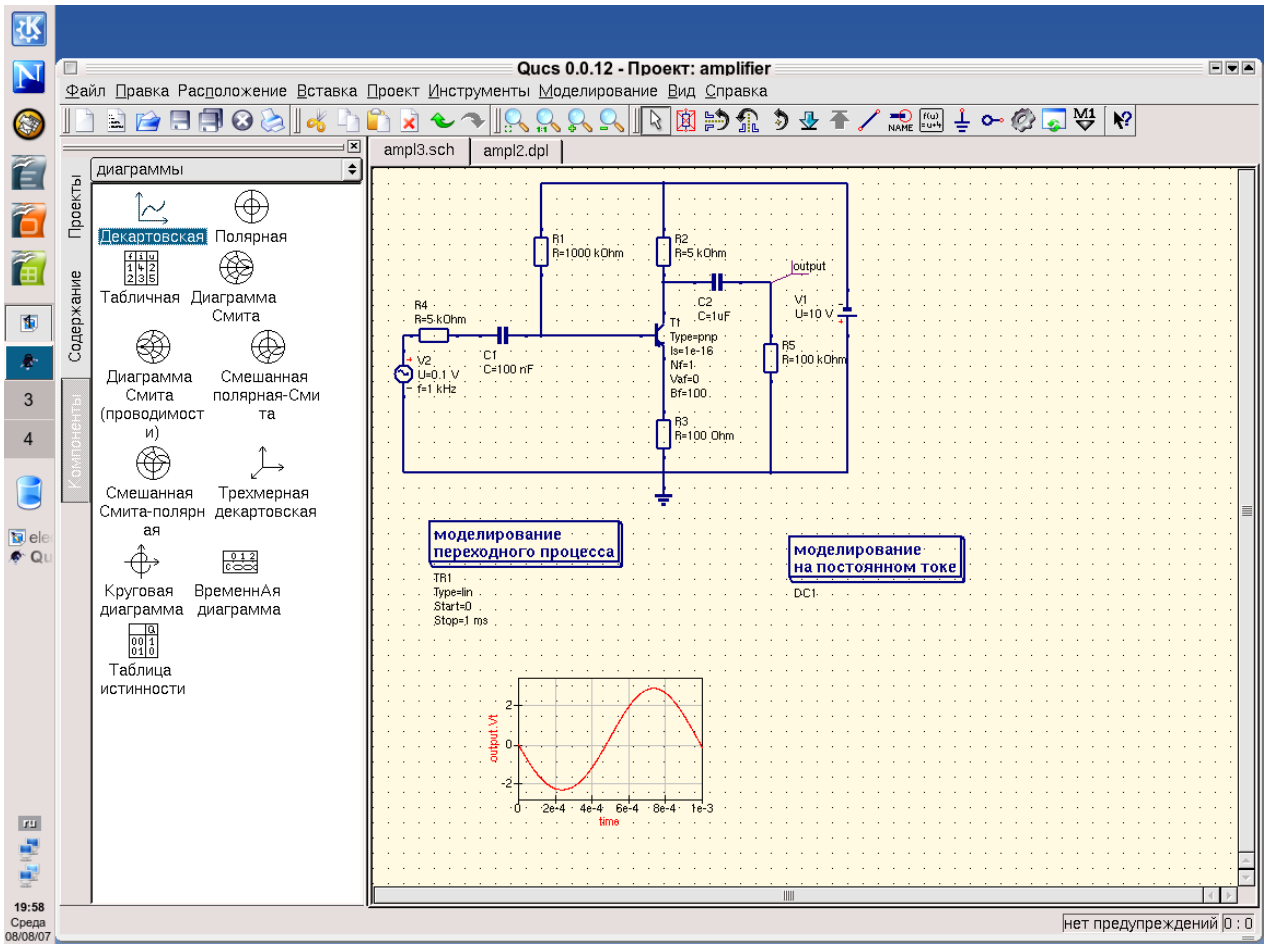


Рис. 2.18. Определение входного сопротивления усилителя

Чтобы было удобнее рассматривать графики, я добавил на выход усилителя конденсатор, отделяющий постоянное напряжение, и еще один резистор, который может представлять сопротивление измерительного прибора.

На практике при таком методе определения входного сопротивления лучше использовать подходящее переменное сопротивление (на входе) и вольтметр переменного напряжения (на выходе). Измеряя напряжение на выходе усилителя при сопротивлении потенциометра равном нулю, поворачивают ручку потенциометра до положения, в котором это выходное

напряжение становится равным половине исходного. Теперь потенциометр можно отключить от схемы измерения, и замерить величину сопротивления мультиметром.

Каково назначение элементов на схеме рисунка 2.17? Конденсатор C1, препятствуя прохождению постоянного тока, устраняет влияние этого тока на режим работы транзистора, который, в свою очередь, задается током базы транзистора, а величина этого тока резистором R1. Чаще всего его выбирают таким, чтобы напряжение на коллекторе транзистора было равно половине напряжения питания. Вторая половина напряжения питания оказывается приложена к резистору R2. Этот резистор – сопротивление нагрузки усилительного каскада на транзисторе. В реальных схемах нагрузкой может быть реле постоянного тока, громкоговоритель или наушник карманного приемника, или просто резистор, к которому через конденсатор подключается следующий каскад усиления. Конденсатор в этом случае часто называют разделительным или переходным. Комбинируя два типа транзисторов в многокаскадных усилителях можно обойтись без этих конденсаторов, что особенно важно, когда усилитель должен усиливать не только переменное напряжение, но и постоянное.

Очень интересно назначение резистора R3. Его может не быть в реальной схеме. Этот резистор не является обязательной частью схемы, но, благодаря ему, входное сопротивление каскада близко к 5 кОм, и без него это сопротивление было бы значительно меньше. Резистор R3 вводит в каскад обратную связь. Каким образом это происходит?

Рассмотрим распределение напряжения питания на элементах входной цепи. Напряжение питания равно сумме падений напряжений: на резисторе R1, переходе база-эмиттер, резисторе R3. Если какой-то фактор вызывает непредусмотренное изменение тока эмиттера транзистора, это может быть температура или питающее напряжение, изменяется напряжение на резисторе R3, а, следовательно, остальные напряжения. Для определенности положим, что ток эмиттера под воздействием температуры увеличился, увеличилось напряжение на резисторе R3, тогда уменьшится напряжение на переходе база-эмиттер, что приведет к уменьшению базового тока, уменьшению тока коллектора и уменьшению тока эмиттера, как суммы базового и коллекторного тока. Таким образом, при непредусмотренных изменениях режима работы транзистора резистор R3, как бы стремится вернуть режим работы к задуманному. Резистор R3 оказывает стабилизирующее действие на работу каскада. Этот резистор является общим элементом для входной и выходной цепей усилителя, осуществляя *обратную связь*. То есть, такую связь, когда часть выходного сигнала попадает на вход, оказывая влияние на работу схемы. В данном случае эта обратная связь получается отрицательной – часть выходного сигнала вычитается из входного. Если бы часть выходного сигнала складывалась со входным, то получилась бы положительная обратная связь. О влиянии отрицательной обратной связи на входное сопротивление вы можете судить, повторяя эксперимент с измерением входного сопротивления усилителя, как на рисунке 2.18, но меняя величину резистора обратной связи R3.

Отрицательная обратная связь по постоянному току стабилизирует работу транзистора, стараясь поддерживать заданный режим работы. Еще большего эффекта стабилизации можно добиться, если применить во входной цепи транзистора делитель напряжения, добавив еще один резистор между базой транзистора и общим проводом. Если выбрать величину этого резистора меньше входного сопротивления каскада, то падение напряжения на этом резисторе можно рассматривать как источник питания базовой цепи транзистора. В этом случае величина резистора R1 существенно уменьшится для обеспечения прежнего тока базы. Но теперь стабилизирующее действие резистора обратной связи R3 становится еще очевиднее. Напряжение источника питания (напряжения на дополнительном резисторе, R4 на схеме ниже) разделяется между напряжением база-эмиттер и напряжением на резисторе R3. Если напряжение на нем непредусмотренным образом изменится из-за изменения тока эмиттера, то изменение напряжение на переходе база-эмиттер транзистора, меняя ток базы,

вернет режим работы к исходному (или постарается сделать это).

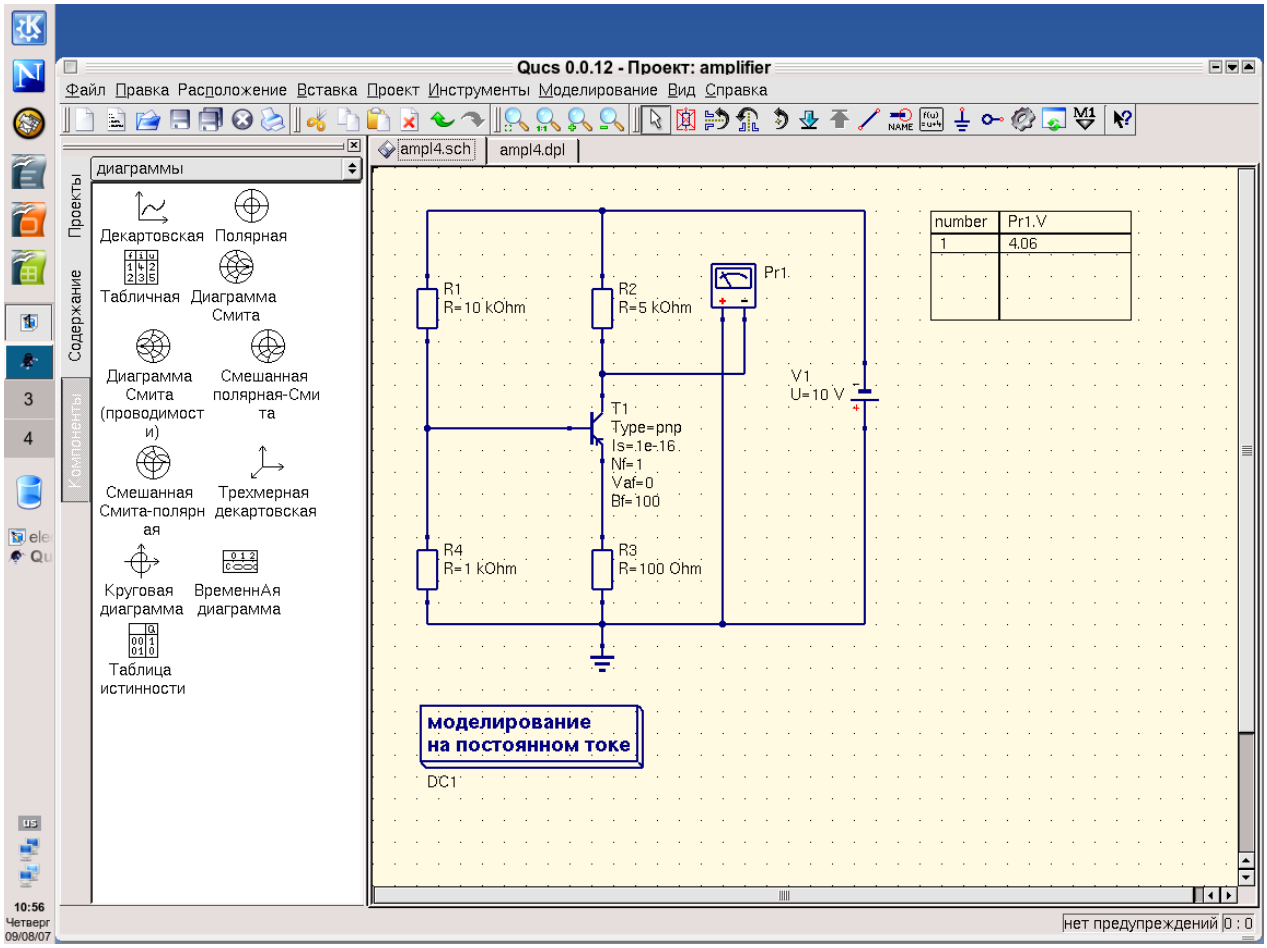


Рис. 2.19. Схема стабилизации рабочего режима транзистора

Как видите, даже такая простая схема, как делитель напряжения, находит широкое применение на практике. Очень советую, даже если у вас есть некоторый опыт работы со схемами, проделать ряд экспериментов с простым усилителем на одном транзисторе, меняя значения резисторов. Если вы будете проделывать эти эксперименты на макетной плате, то, чтобы не «сжечь» компоненты, постарайтесь хотя бы «на вскидку» подсчитать токи, напряжения, рассеиваемую мощность. Если эти эксперименты вы будете проделывать в программе Qucs, то советую почитать Qucs Workbook, русскоязычная версия которой есть на моем сайте (мой конспект оригинала, http://vgololobov.narod.ru).

Схем усилителей на транзисторах очень много. И лет двадцать-тридцать назад было очень важно уметь рассчитать, построить или выбрать подходящую схему на транзисторах. За последние годы подход к этому несколько изменился. В первую очередь благодаря появлению большого количества типов специализированных микросхем и операционных усилителей.

Операционный усилитель

На плате, что передо мной, на микросхеме трудно прочитать надпись, но, похоже, это 140УД708, операционный усилитель. Что такое операционный усилитель?

Когда цифровая техника только начинала свое развитие, для сложных математических расчетов использовали аналоговые компьютеры. Такие операции, например, как

дифференцирование и интегрирование, с успехом выполнялись этими компьютерами, электронной основой которых были специальные усилители с большим коэффициентом усиления, большим входным и малым выходным сопротивлением, способные усиливать и постоянный, и переменный ток. Эти усилители, предназначенные к выполнению операций, назывались операционными. С появлением транзисторов эти усилители стали делать на транзисторах, позже их стали конструктивно оформлять в виде микросхем. Все более и более совершенствовались и транзисторы, и схемотехника, и технологии изготовления микросхем, что приводило к росту количества микросхем операционных усилителей и снижению их стоимости. В какой-то момент стало понятно, что усилитель, операционный он или нет, это усилитель. Операционные усилители стали все шире применять там, где требуется усиление сигналов или их преобразование. Постепенно микросхемы операционных усилителей стали вытеснять транзисторы (хотя сами буквально «напичканы» транзисторами) из их привычных областей применения. Сегодня, пожалуй, операционный усилитель в этом смысле можно рассматривать также, как раньше рассматривали транзистор – базовая ячейка активного элемента схемы. Сегодня много типов микросхем, в корпусе которых два или четыре операционных усилителя, каждый из которых имеет очень большой коэффициент усиления по напряжению, и там, где раньше для усиления сигнала требовалось собрать схему на десятке транзисторов, сегодня достаточно применить один операционный усилитель (а их в микросхеме четыре!).

Конечно, и операционный усилитель не панацея. В перечне выпускаемых микросхем есть и разного рода функциональные узлы, и те же усилители, как высоких частот, так и низкочастотные. Их существование обусловлено особенностями задач, ими решаемых, конкретным назначением. Все больше появляется микросхем, выполняющих комплекс задач, и, по существу, являющихся полным электронным устройством – посмотрите на электронные часы! Но и в таких микросхемах вы найдете те же резисторы, конденсаторы, индуктивности и транзисторы.

С появлением операционных усилителей и специализированных микросхем существенно изменился вид электрических схем. Сейчас многие микросхемы изображаются в виде прямоугольников, из которых в разные стороны уходят выводы, обозначенные одной-двумя буквами или знаками. Так происходит с аналоговыми устройствами, так происходит с цифровыми устройствами, и с устройствами автоматики, и с измерительными приборами. Если раньше, взглянув на схему устройства, можно было сразу сказать, что перед тобой – усилитель низкой частоты, или радиоприемник, или тестер, – то сегодня это сделать гораздо труднее. То же можно сказать и о работе со схемами. Если раньше на макетной плате размещалось несколько транзисторов, некоторое количество резисторов и конденсаторов, и можно было легко измерить с помощью тестера все токи и напряжения, а, подав сигнал от генератора, посмотреть на экране осциллографа выходной сигнал и «все понять», то сегодня приходится долго думать, как разместить на макетной плате микросхему с большим количеством выводов, где набрать достаточное количество генераторов, чтобы обеспечить микросхему всеми необходимыми для ее проверки сигналами, и чем посмотреть отклик на эти сигналы на нескольких выводах микросхемы. В этом смысле все большее значение приобретает моделирование работы электрической схемы на компьютере, особенно при разработке микросхем, где элементы зачастую представляют собой некие геометрические объекты, создаваемые из разных материалов. Не следует только забывать, что компьютер, моделирующий работу электрической схемы, позволяющий лучше разобраться в работе схемы, позволяющий больше узнать о схеме, все-таки остается теоретической конструкцией, и только собрав «живое» устройство и проверив его работу, вы можете с уверенностью сказать, работает ли оно, и правильно ли оно работает.

Однако ветер технологических перемен унес меня далеко в сторону. Гораздо дальше, чем

этого требует рассказ, который я начал с таких простых предметов обсуждения, как ток и сопротивление. Возвращаясь к рассказу, замечу, что, согласитесь, разделив материалы на проводники и изоляторы по их сопротивлению прохождению электрического тока, мы достаточно естественно перешли к полупроводникам, занимающим промежуточное положение между проводниками и изоляторами. Обозначив же наличие двух типов проводимости у полупроводников (n-типа и p-типа), вполне обосновано перешли к композиции из этих двух материалов, из которых построен полупроводниковый диод. Для практических целей можно просто принять во внимание, что на границе раздела полупроводниковых материалов образуется приграничный слой, напоминающий заряженный конденсатор, внутри которого есть электрическое поле, и этого будет достаточно, чтобы понять работу, например, диода в качестве выпрямителя.

Диоды, простейшие конструкции из двух полупроводниковых материалов разного типа проводимости, сами по себе очень интересные элементы электрических схем. При разном изготовлении они приобретают разные свойства, позволяющие применять их и в качестве выпрямителей в силовых устройствах, и в качестве детекторов в радиоприемнике, и в качестве стабилизаторов напряжения, и в качестве конденсаторов переменной емкости для настройки приемника на частоту радиостанции. Приграничный слой, похожий на конденсатор, при изменении питающего напряжения, как бы меняет емкость этого конденсатора – вот вам и конденсатор переменной емкости, а такой диод носит специфическое название *варикап*. Есть диоды, способные играть роль активного элемента генератора – туннельные диоды. Есть диоды, которые светятся при прохождении через них постоянного тока, и их используют в качестве индикаторов, а называют *светодиоды*. Есть диоды, которые реагируют на падающий на них свет, генерируя постоянное напряжение, их называют *фотодиодами*.

Кроме двухслойных полупроводниковых конструкций, есть трехслойные – транзисторы (биполярные типа n-p-n и p-n-p). Кроме биполярных есть полевые или канальные транзисторы. Благодаря особой конструкции эти транзисторы имеют между эмиттером (теперь он называется исток) и коллектором (стоком) канал, по которому движутся носители тока, а ширина этого канала (величина тока) определяется полем, образованным с помощью базы (которая стала затвором полевого транзистора). Базовый ток такого транзистора настолько мал, что его можно считать отсутствующим. А такой транзистор по своему поведению больше напоминает радиолампу, чем биполярный транзистор.

Кроме трехслойных полупроводниковых конструкций существуют не менее интересные и полезные многослойные, например, тиристор (или триак).

Словом, такое, по отношению к сопротивлению протеканию электрического тока, поведение, которое можно было бы назвать «ни рыба, ни мясо», определяемое свойствами материала, открыло невообразимое количество форм очень полезных применений в электронике. А упоминаю я об этом, унесенный ветром своей непоследовательности, только за тем, чтобы еще раз подчеркнуть – за многими таинственными терминами, которые любитель встретит на своем пути, как, например, гиратор или импеданс, лежат такие простые сущности, как ток и сопротивление. Возьмем *импеданс*. Когда мы говорили о конденсаторе и индуктивности, мы рассматривали их сопротивление переменному току. И обозначили его как реактивное, зависящее от частоты изменения переменного тока. Но, что очевидно, кроме реактивного (индуктивного) сопротивления у катушки, имеющей много витков обычного провода, есть и активное сопротивление, определяемое сопротивлением провода, который использовался при намотке катушки. Сочетание этих двух видов сопротивлений позволяет говорить о полном сопротивлении или импедансе.

Вот я, вроде, и вернулся, вернулся к программе Qucs и усилителям. Я хочу проделать

несколько экспериментов с операционными усилителями. Но прежде чем на вкладке *Компоненты* программы Qucs я займусь поиском операционного усилителя, я немного скажу о графическом изображении этого элемента электрической схемы. Достаточно часто его изображают в виде равнобедренного треугольника, основание которого направлено влево и имеет два входа, отмеченных знаками + и -, а вправо уходит выход усилителя. Это самое простое графическое изображение ОУ (OPAM, операционного усилителя), к которому иногда сверху и снизу добавляют выводы питания +Uпит и -Uпит. Вы можете встретить изображение операционного усилителя в виде прямоугольника, к левой части которого подходят входы, а к правой выход и питающие напряжения. Есть, я думаю, и другие вариации графического изображения операционного усилителя. Но в любом графическом виде этот усилитель будет иметь два входа. Один из них называют прямым, а другой инверсным входами операционного усилителя. Об этих двух входах.

В программе Qucs нарисуем, я предпочитаю использовать транзисторы типа n-p-n, простую схему на двух транзисторах.

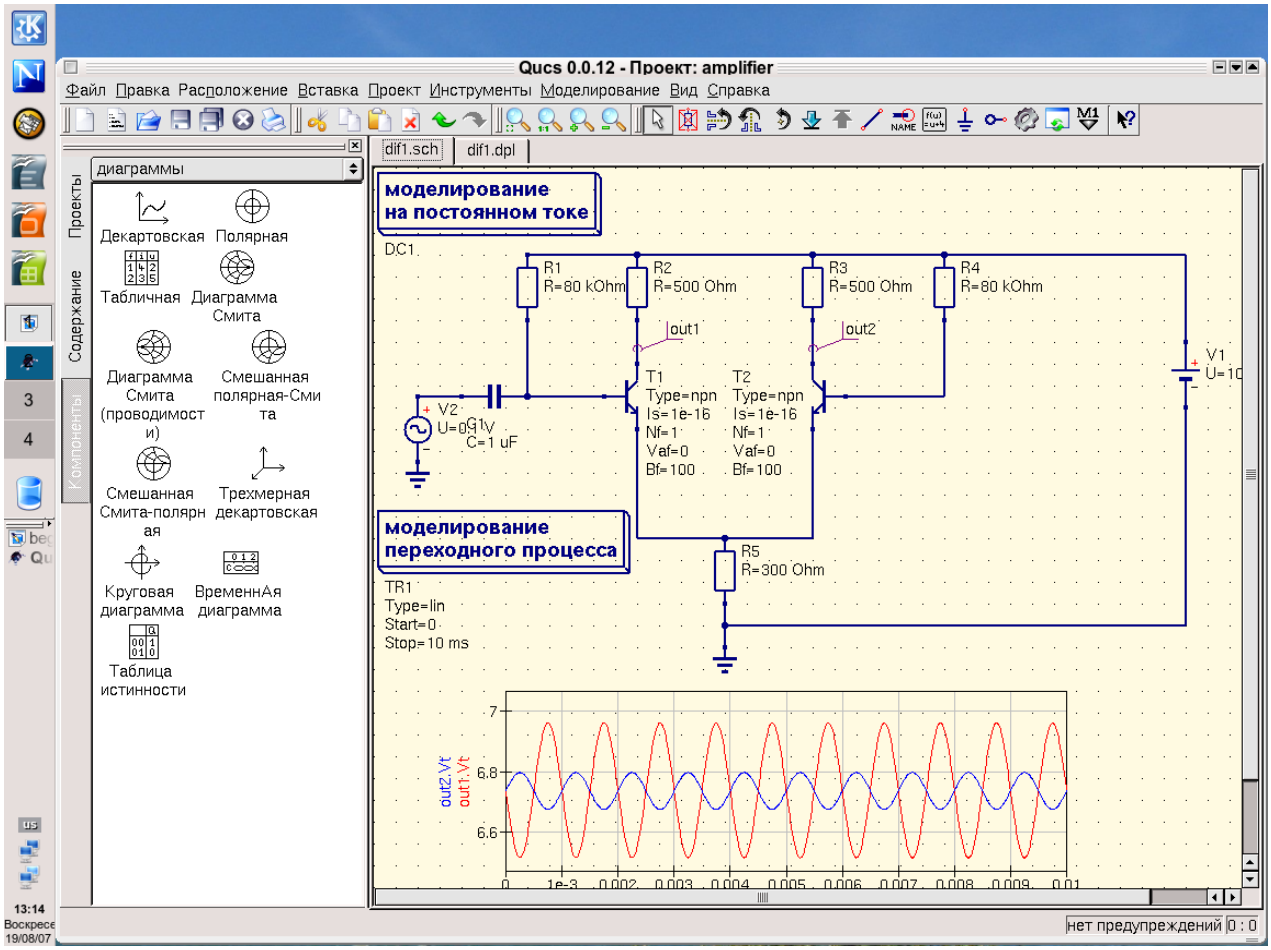


Рис. 2.20. Дифференциальный вход операционного усилителя

Я не сделал ничего особенного, только схему на рисунке 2.17, которую использовал для рассказа об усилении транзистора, повторил еще раз, развернув ее. Все также резистор R1 (и R4) задает ток базы транзистора T1 (соответственно, T2), резистор R2 (и R3) служит нагрузкой, а резистор R5 создает цепь обратной связи (и связи двух транзисторов). Но теперь я могу подключить источник переменного напряжения или к базе транзистора T1, или к базе T2. Подключив его к базе T1, я могу снять напряжение или с резистора R2 (с земли и коллектора транзистора) или с R3. На рисунке 2.20 изображена осциллограмма двух

напряжений: на коллекторе T1, относительно земли, и на коллекторе транзистора T2. Если помните, когда я говорил о напряжении и токе через реактивное сопротивление, то сказал, что они находятся не в одной фазе изменения состояния. Так вот, напряжения на коллекторах двух транзисторов схемы (out1 и out2) находятся в противоположных фазах, или в противофазе. Но одно из них будет в фазе с изменениями напряжения источника. Выбрав в качестве выхода схемы коллектор одного из транзисторов, я получу два входа, один из которых будет в фазе с выходом (прямой вход), другой в противофазе (инверсный вход). И еще. Такая схема позволяет снять выходное напряжение не с коллектора одного из транзисторов и общего провода, а с точек, отмеченных на схеме, как out1 и out2. Включите между ними измеритель напряжения. Посмотрите осциллограмму этого напряжения. А затем подключите ко второму входу еще один источник переменного напряжения, задав ему такие же точно параметры, как у первого и еще раз посмотрите осциллограмму. Пожалуй, последний вариант схемы я приведу.

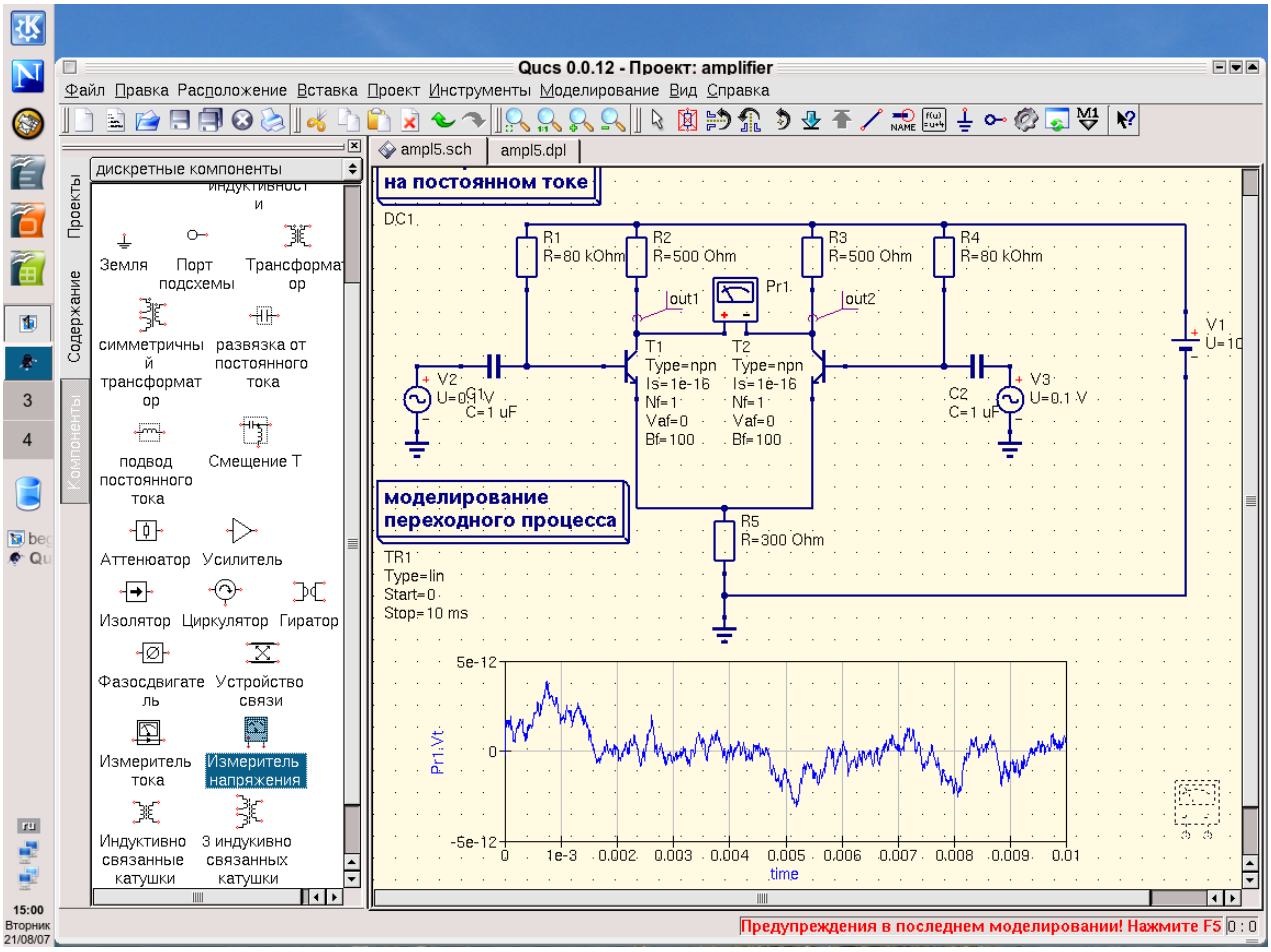


Рис. 2.21. Синфазные сигналы на входах ОУ

Если с одним источником сигнала на входе операционного усилителя (представим, что это схема операционного усилителя) измеритель напряжения, включенный между точками *out1* и *out2*, показывал вам некоторое напряжение, то два синфазно работающих источника напряжения дают на выходе только небольшой шум (амплитуда на графике меньше, чем 5 в минус 12 степени!). Так проявляется очень полезное свойство операционного усилителя, имеющего обычно входной каскад, включенный по схеме дифференциального усилителя. Последнее означает, что на выходе будет усиленный сигнал только тогда, когда на два входа подаются не синфазные (не находящиеся в одной фазе) сигналы. Синфазные сигналы не

усиливаются, а взаимно подавляются. В схеме на рисунке 2.21 я использовал именно синфазные источники переменного напряжения V2 и V3, то есть, находящиеся всегда в одной фазе изменения напряжения. И настроил я их так, что их частота одинакова, их напряжение одинаково. Так в чем же польза этого эксперимента?

Построение симметричного входного каскада операционного усилителя, имеющего два входа, которые реагируют на разностное (по входам) переменное напряжение, но подавляют синфазные (по входам) сигналы, полезно тем, что часто источник полезного сигнала (источник переменного напряжения) включают двумя проводами, на которые наводятся посторонние напряжения. Их так и называют «наводками». Эти наводки наводятся одинаково на два провода, соединяющие источник и усилитель. Если мы включаем полезный источник, как на рисунке 2.20, то наводки складываются и усиливаются усилителем – ему все равно, что усиливать. Но если мы включим два провода на два входа ОУ, то получим, что источники вредного напряжения наводок включены, как на рисунке 2.21, то есть, они синфазны, взаимно компенсируются и не усиливаются усилителем. Я упоминал о существовании диода, чувствительного к падающему на него свету, фотодиоде. Обычно он работает как очень слабенький источник напряжения. Его напряжение нужно усилить, а наводки могут полностью подавить полезный сигнал от него. Но, если включить его на дифференциальный вход ОУ, то можно не беспокоиться о наводках. Усилится полезный сигнал, а не наводки.

Боюсь, я не очень внятно все объяснил с синфазностью и противофазностью. Позже, когда мы лучше познакомимся с усилителями, я попробую сделать это иначе.

Операционные усилители любят, когда их цепь питания состоит из двух источников, включенных последовательно, а за общий схемный провод (или землю) принимается точка соединения двух источников питания. Такое устройство питающего напряжения позволяет исключить из схемы переходные конденсаторы между каскадами, позволяет реализовать усилитель постоянного тока, что важно для работы с очень медленно меняющимися напряжениями. Там же, где эти преимущества не имеют особого значения, применяют питание от одного источника. Но и в этом случае для переносных устройств иногда применяют схемы преобразования одного источника питания в два. Многие удобные операционные усилители плохо, или вообще не работают, с питающим напряжением, скажем, +4.5 и -4.5 вольт, им нужно более высокое напряжение питания. В этом случае батарейка используется для питания, в первую очередь, преобразователя напряжения, от которого уже запитываются операционные усилители. И схема, которая ближе к реальной, выглядит несколько иначе. Я перерисую предыдущую схему.

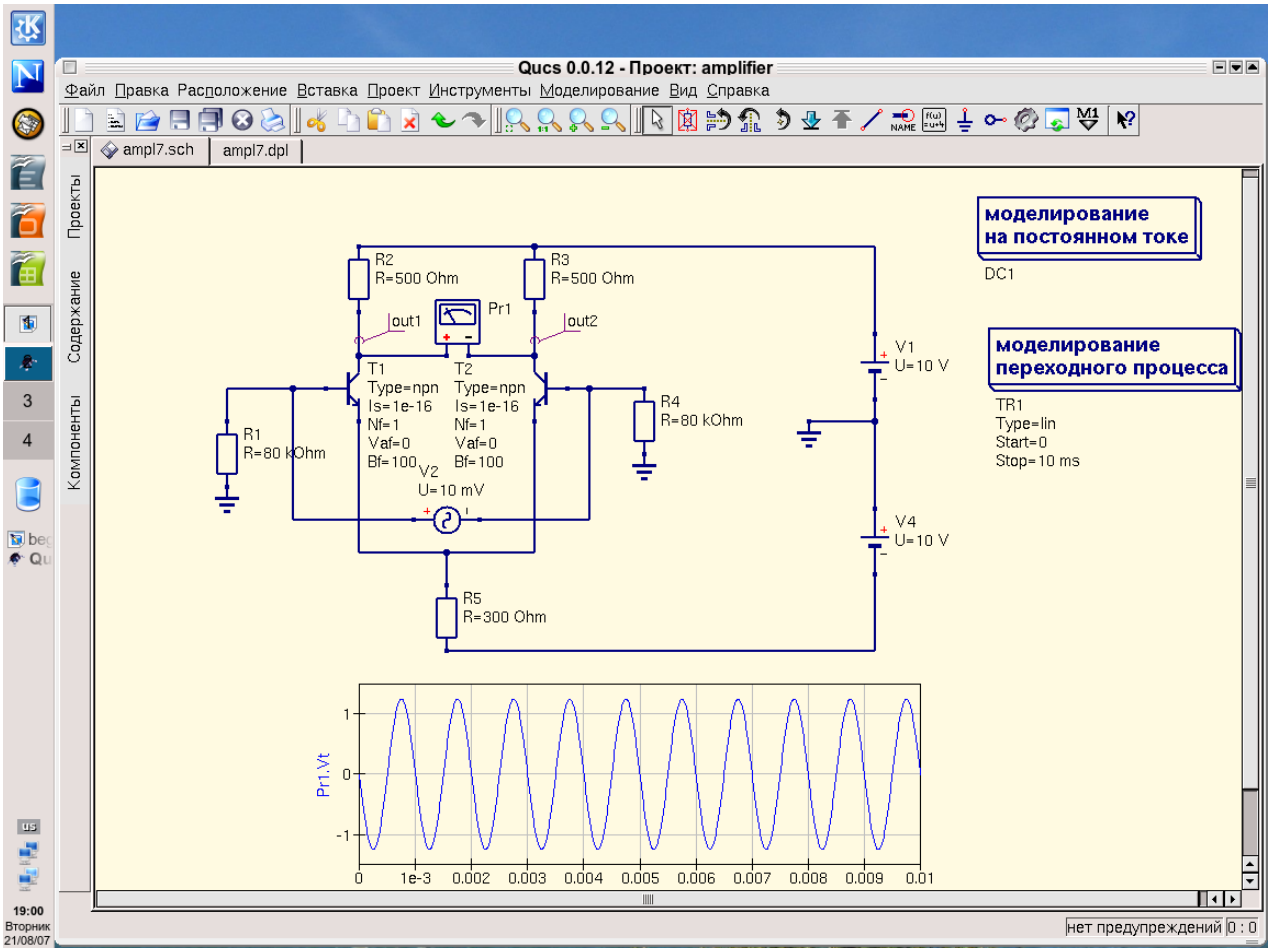


Рис. 2.22. Питание входного каскада ОУ от двух источников

Хотя реальный входной каскад современного операционного усилителя устроен гораздо сложнее, его сущность можно свести к такой схеме. На ней я включил источник переменного напряжения – это может быть, например, микрофон – на два входа, измеритель напряжения между двумя транзисторами, и, если вы обратили внимание, что напряжение источника 0.01 В (10 мВ), а напряжение, показываемое измерителем (амплитуда), более 1 В, то можно оценить коэффициент усиления этого каскада по напряжению. И, что тоже приятно, не понадобился разделительный конденсатор на входе. А получившийся усилитель может усиливать и постоянное напряжение, и переменное. Тоже очень полезное свойство.

Я немного рассказал об операционном усилителе, который увидел на плате устройства, лежащей передо мной. Можно рассказать гораздо больше полезного, что я сделаю в следующей главе, а пока отметьте, мы использовали только те понятия, о которых уже говорили – ток, сопротивление, транзистор.

Цифровая микросхема

Последний из типов компонент, которые я обнаруживаю в устройстве, разобранном мной для знакомства с электронными компонентами, это цифровая микросхема. Надпись так стерта, что я даже не хочу пытаться ее прочесть. Откуда же я знаю, что это цифровая микросхема? Сознаюсь, что посмотрел схему устройства, и посмотрел спецификацию – перечень элементов к схеме. Нужно же как-то продолжить рассказ.

Почему цифровые микросхемы так называют?

Не знаю. Может быть правильнее их было называть числовые, поскольку их создавали для операций с числами: сложение, вычитание, умножение, деление... Но о числах и о том, как микросхемы справляются с операциями над числами несколько позже, а прежде я хочу вернуться к схеме на рисунке 2.20 и пояснить, почему захотел нарисовать ее именно так. Я нарисую ее еще раз, чуть-чуть видоизменив.

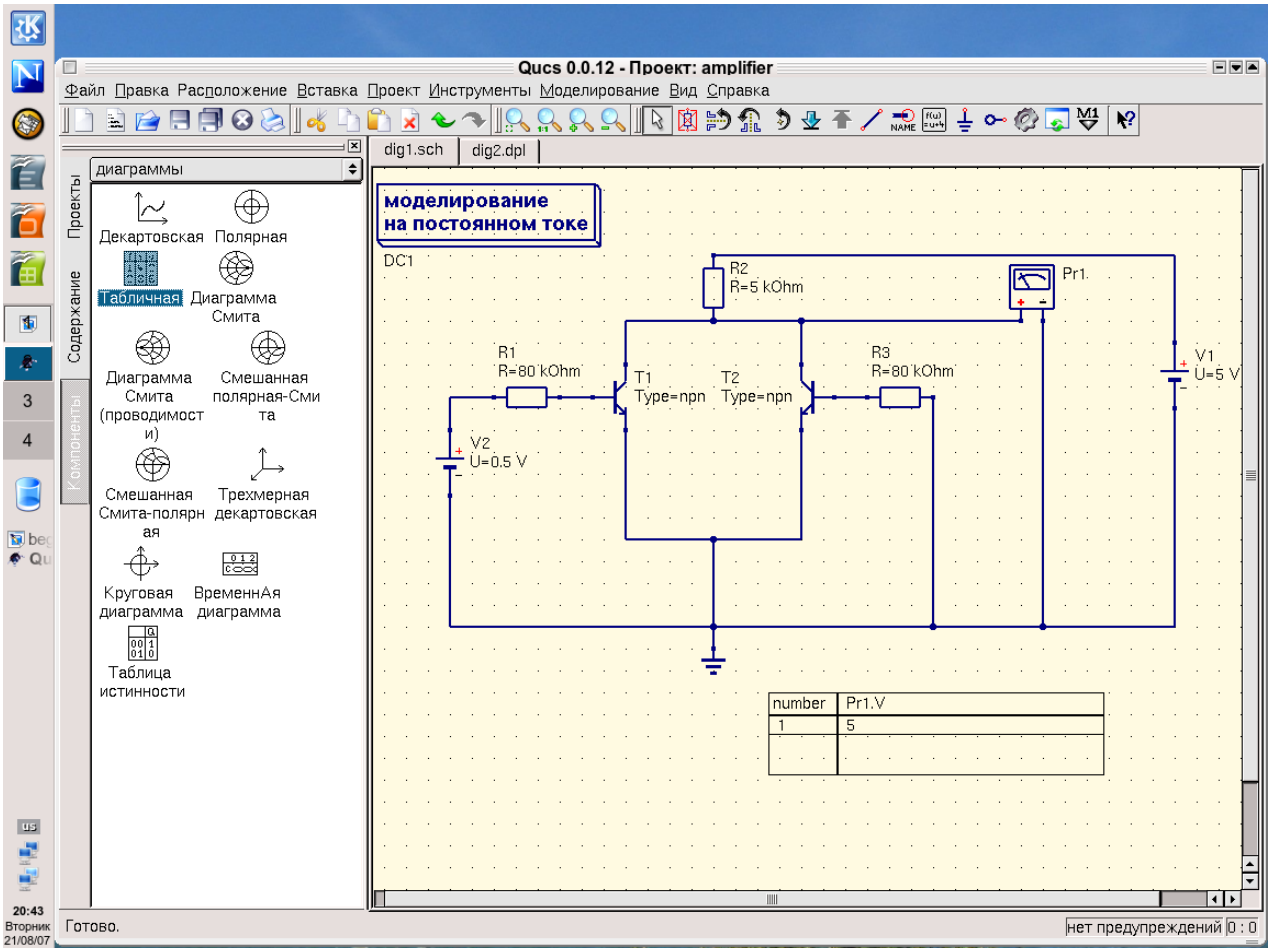


Рис. 2.23. Схема с двумя входами и общим резистором нагрузки

Эта схема похожа на входной каскад операционного усилителя, но представляет собой упрощенную схему цифровой микросхемы (резистивно-транзисторной логики, РТЛ). При том включении, что есть на рисунке, измеритель показывает напряжение 5 В равное напряжению источника питания. Если любой из резисторов R1 или R3 (или оба) подключить к источнику постоянного тока с напряжением 2.5 В, то измеритель напряжения покажет...

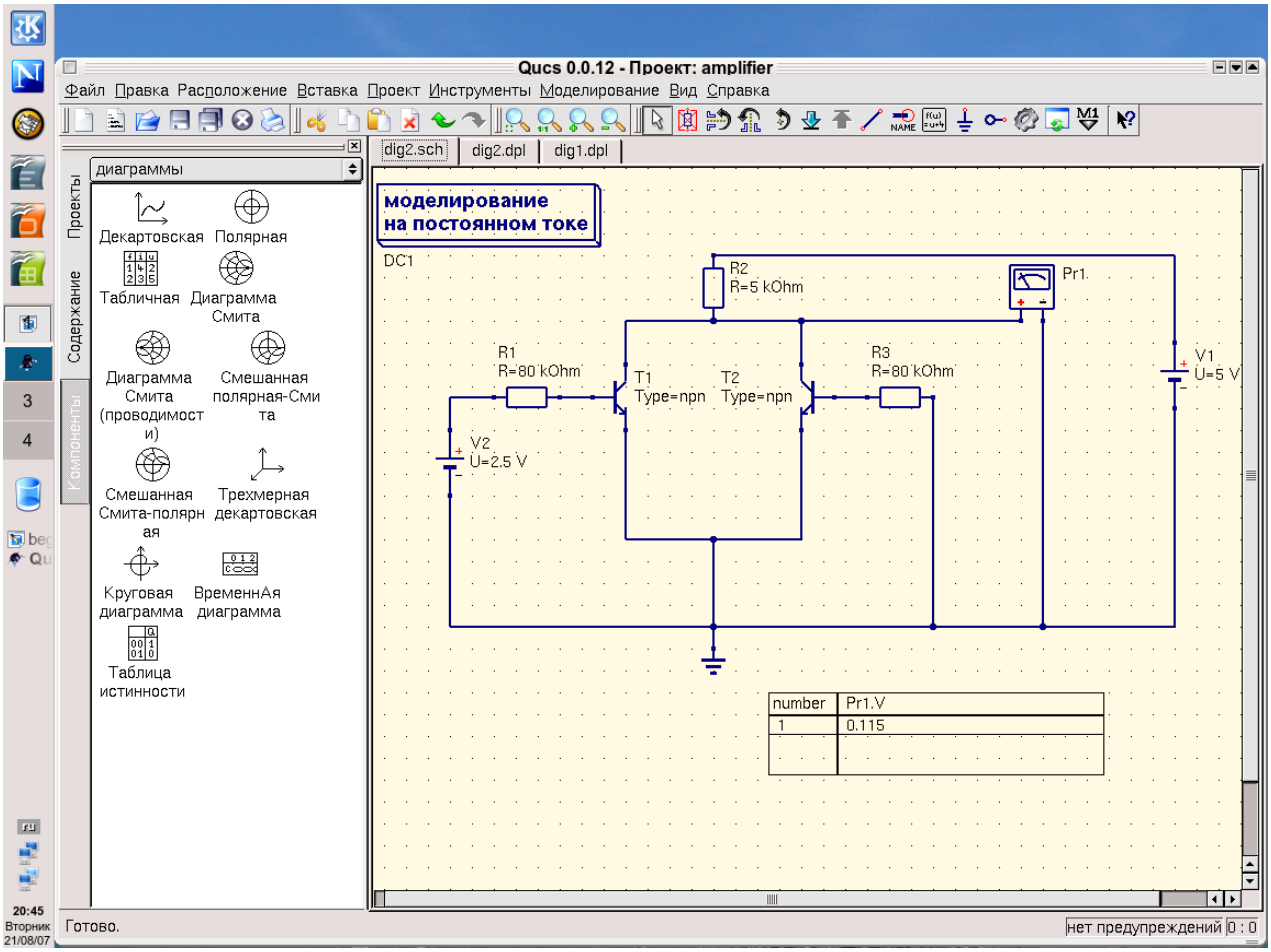


Рис. 2.24. Работа схемы при входном напряжении 2.5 В

... покажет 0.115 В. Напряжения, которые я выбрал: 5 В для питания, 0.5 В в первом случае и 2.5 В во втором – это характерные напряжения для цифровых микросхем, как о них говорят, ТТЛ. Что это значит? Это значит, что микросхема транзисторно-транзисторной логики. Надеюсь, понятно... но я, кажется, умудрился, пока правил текст, удалить большой кусок текста, без которого, скорее непонятно, чем понятно. Восстановим этот удаленный текст.

Числа записываются с помощью цифр. В привычной нам системе счисления этих цифр 10, от 0 до 9, а система счисления называется десятичной. Но так было не всегда. Бывали и другие системы счисления, от которых нам досталось деление времени на 12 частей на циферблате часов. С появлением цифровой техники, особенно процессорной, стали употреблять шестнадцатеричную систему счисления, имеющую 16 цифр, от 0 до 9 плюс A, B, C, D, E, F. Последние латинские буквы дополняют недостающие цифры. Иногда употребляют восьмеричную систему счисления с цифрами от 0 до 7. Но самая главная для цифровой техники остается двоичная система счисления с цифрами 0 и 1. Всего две цифры. Любое число можно представить в любой системе счисления. Им, числам, все равно, как их записывают. Они так же складываются, вычитаются, умножаются и делятся в любой системе счисления, а в итоге получаются опять числа в той же системе счисления. Почему двоичная система счисления удобна в цифровой технике? Потому что можно сопоставить цифре 0 напряжение низкого уровня, например, 0.5 В. А цифре 1 сопоставить напряжение высокого уровня, например, 2.5 В. Эти два напряжения, как это следует из наших последних экспериментов, хорошо распознаются электрическими схемами. Мы всегда по выходному напряжению можем

судить о том, какая цифра 0 или 1 на входе. Как выглядит число, возьмем что-нибудь попроще, например, 5 в двоичной системе счисления? Как 101. С помощью двух цифр можно записать любое число, только оно получается очень «длинным». Есть правила перевода чисел из одной системы счисления в другую. Но сам я, сознаюсь, пользуюсь калькулятором операционной системы Linux, который может сделать это быстро и точно.

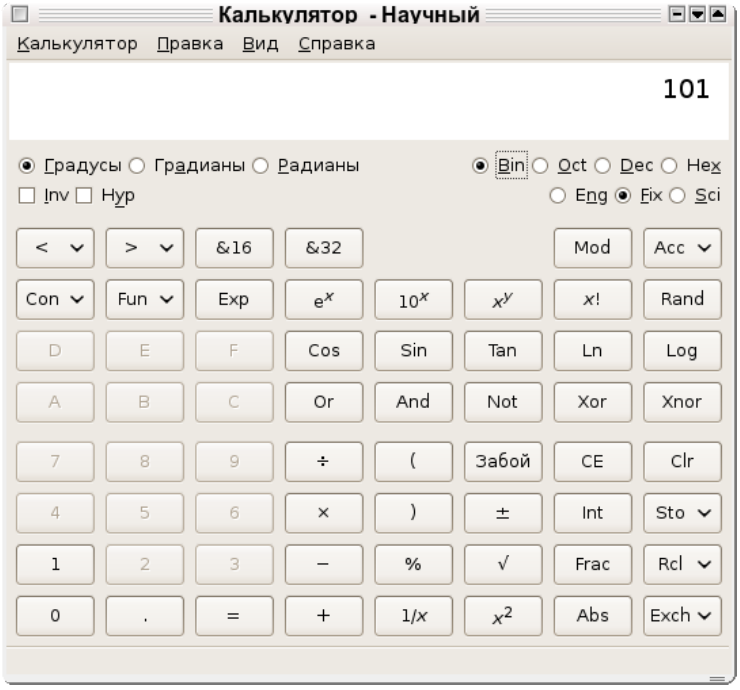


Рис. 2.25. Калькулятор в Linux с заданным «научным» видом

Переключая режим работы калькулятора в Bin, Oct, Dec и Hex можно получить любое число в любой системе счисления. Конечно, если вам больше нравится, вы можете в соответствующем разделе математики или в книге по вычислительной технике посмотреть формулы перехода от десятичной системы счисления к любой другой, но можно использовать калькулятор, для того его и придумывали. А я вспоминаю, как огорчался мой добрый знакомый, когда после появления первых микропроцессоров они стали «сыпаться как из рога изобилия». Работая с первыми процессорами специалисты часто заучивали наизусть все их команды в ассемблерном виде. Я сам запоминал машинные коды коды команд в восьмеричном виде. Но все это потеряло смысл с появлением новых процессоров, которые имели новые обозначения своих команд, а старые процессоры быстро сходили со сцены. Но это так, к слову. Важно запомнить, что суть числа не меняется от того, как мы его записываем, какие значки используем для записи цифр, и сколько этих цифр используем для записи числа. Моих 15 рублей (число денег в кармане) мне хватит на одну поездку, в какой системе счисления я их ни буду пересчитывать. Но вернемся к двоичной системе счисления, которую так любят цифровые микросхемы, и которую так не любят люди – тяжело запоминать длинные последовательности нулей и единиц. Из-за нелюбви к этим длинным последовательностям они и стали использовать восьмеричную и, особенно, шестнадцатеричную системы счисления для записи всего, что творится в компьютерных мозгах. Схемы, которые хорошо различают два напряжения, могут быть построены разным образом, их разным образом в разное время и создавали. С развитием технологий создания микросхем наибольшее распространение получили схемы,

использующие биполярные, а затем и полевые транзисторы. И технологии продолжают совершенствоваться. Каждая из технологий использует свое питающее напряжение и значения напряжений для цифр 0 и 1. Параллельно с развитием идей расчетов с применением двоичной системы счисления развивались идеи бинарной логики, для операций которой требуются два состояния «истинно» и «ложно». При этом сами операции бинарной (или Булевой) логики во многом схожи с арифметическими операциями. Если сопоставить нулю, скажем, состояние «ложно», а единице – «истинно», то одни и те же, практически, схемы можно использовать для двух целей. Логические операции удобно записывать с помощью таблиц, которые называют таблицами истинности. В применении к микросхемам эта таблица описывает состояние выходов микросхемы в терминах «истинно-ложно», когда на ее входах сигналы принимают все возможные значения в тех же терминах, которые легко записывать с помощью цифр 0 и 1. Так с самого начала базовым узлам цифровых микросхем стали присваивать такие названия, как транзисторно-транзисторная логика, и обозначать микросхемами И, ИЛИ, НЕ, И-НЕ и т.д. А схема на рисунке 2.24 может быть названа схемой ИЛИ-НЕ.

Первоначально этот текст, отчего-то мне кажется, был написан лучше. Но теперь уже поздно об этом говорить. Проведем несколько экспериментов с цифровыми микросхемами, чтобы получить таблицы истинности для микросхем «И» и «ИЛИ».

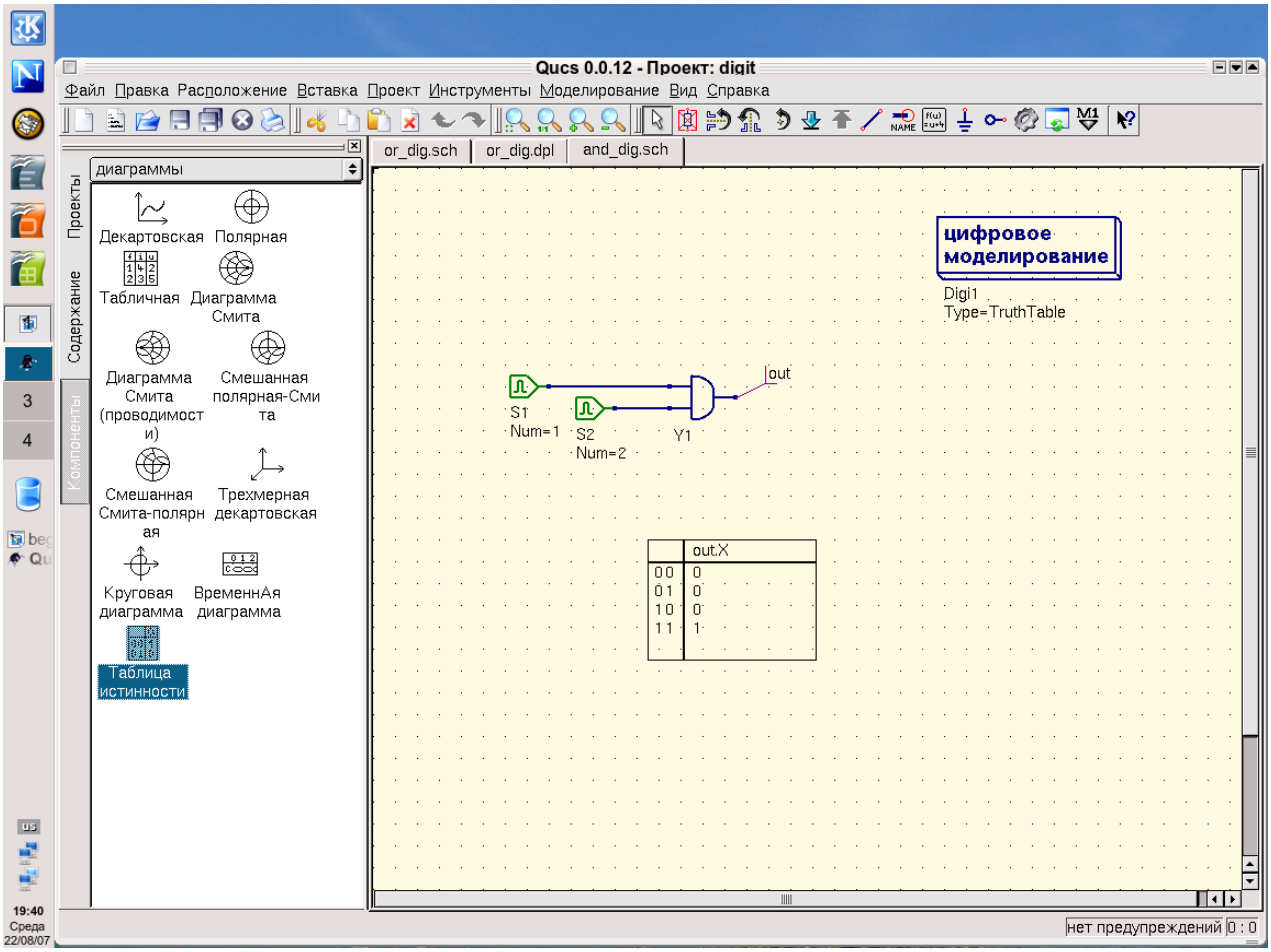


Рис. 2.26. Работа схемы И

К входам двух-входовой схемы «И» подключены источники цифровых сигналов, ее выход

обозначен как «out». А таблица истинности показывает, что логическая единица на выходе будет появляться только тогда, когда оба входа имеют уровень 1.

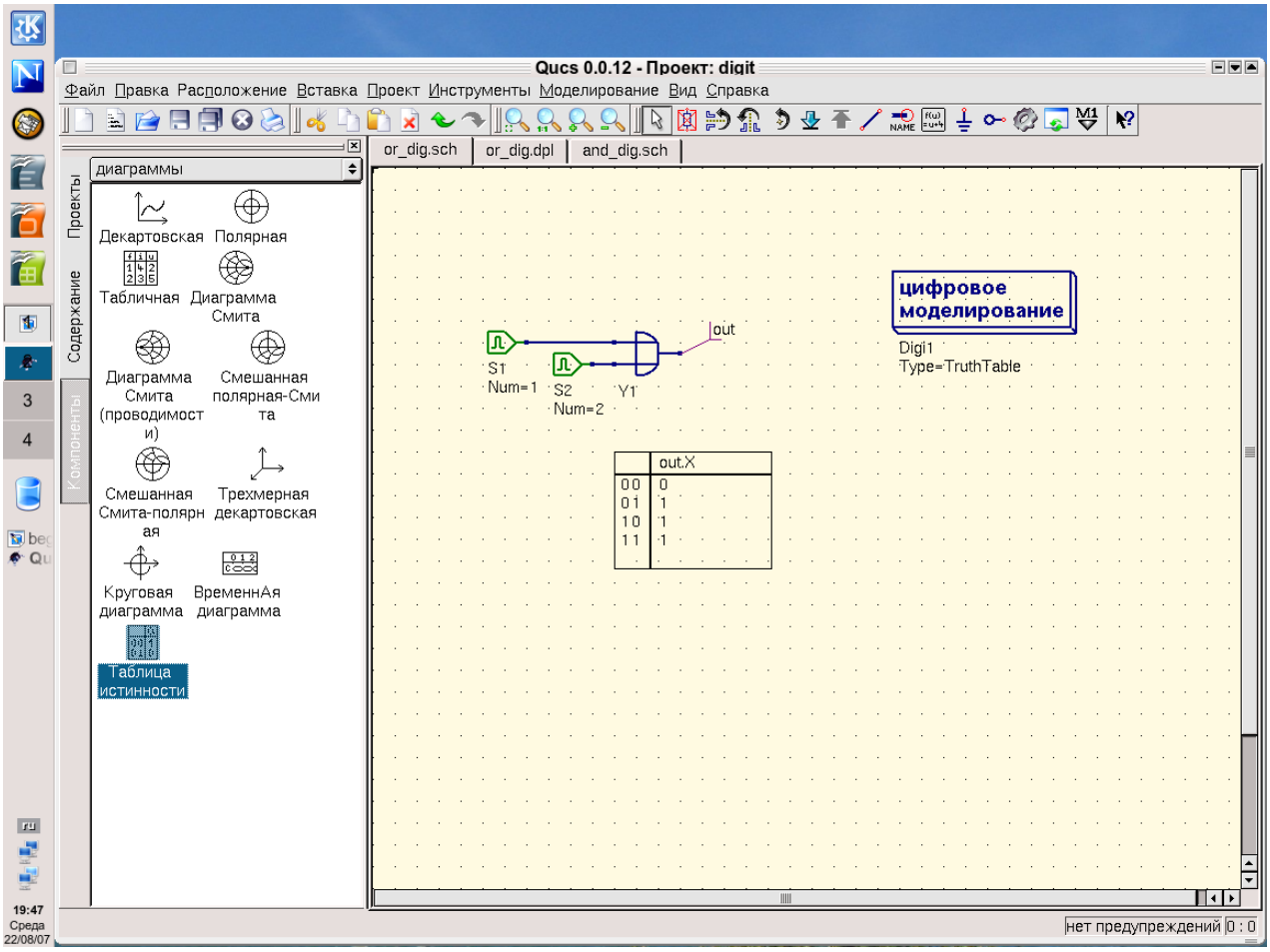


Рис. 2.27. Работа схемы ИЛИ

В случае схемы «ИЛИ» логический 0 на выходе будет присутствовать только тогда, когда на обоих входах логический 0.

Таблицы истинности для логических элементов, таблицы состояния выходов в зависимости от состояния входов, можно построить иначе, чем сделал я, воспользовавшись средствами программы Qucs. Можно подключать ко входам схем источник постоянного напряжения для получения уровня логической «1», и соединять входы с землей для получения уровня логического «0», а на выход включить измеритель напряжения, как я это сделал на рисунке 2.24. Перебирая все возможные состояния входов, мы получим все состояния выходов, которые и занесем в таблицу. Получившаяся таблица и будет таблицей истинности – таблицей состояния выходов. Если выход в состоянии «1», то это соответствует в логике состоянию «истинно», если в «0», то «ложно».

Самое важное, что сейчас можно отметить, пожалуй, то, что, если к базовым элементам «И» и «ИЛИ» добавить инвертор, превращающий логическую «1» в логический «0», то мы получим набор, из которого можно построить все многообразие цифровых микросхем: триггеры, регистры, дешифраторы, мультиплексоры, сумматоры и т.д. Как это происходит, можно рассмотреть на примере того, как можно сложить в двоичной системе счисления две единицы и получить в ответе двойку (10 в двоичной записи). Этот пример я подсмотрел в книге о микро-компьютере.

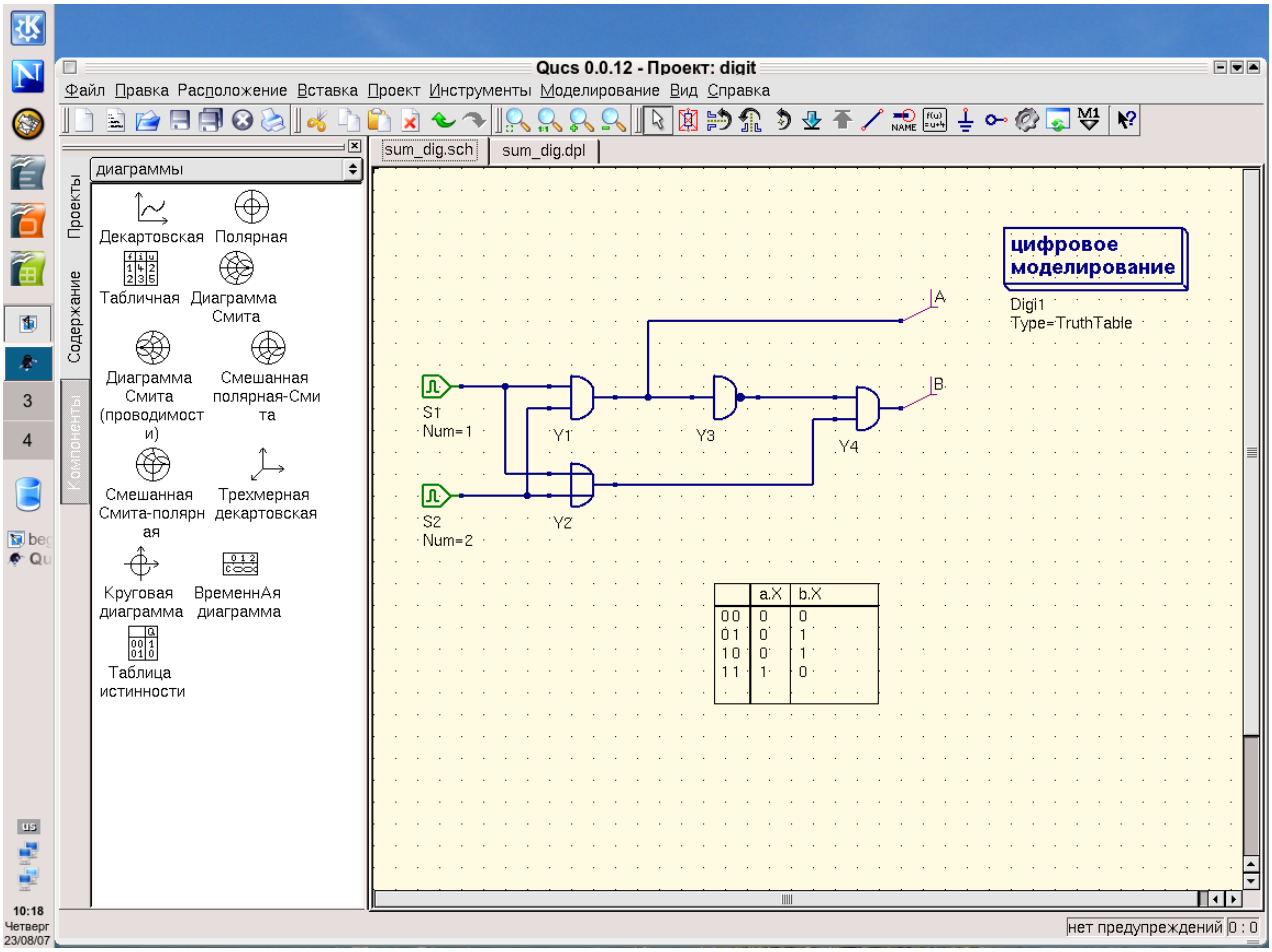


Рис. 2.28. Сложение в двоичном мире цифровых микросхем

Поскольку на выходе схемы должно получиться число, то его величина, а числа мы привыкли записывать позиционно, записывается в двух разрядах. На схеме *b* – младший разряд, *a* – старший. Позиционная запись, напомню, означает, что в зависимости от занимаемой цифрой позиции (разряда) она обозначает разные числа. Например, в десятичной системе счисления 1 – это единица, 10 – десять. Единица, занимающая вторую позицию, указывает уже количество десятков в числе. В полученной таблице в первой колонке показано, какие двоичные числа подлежат сложению. Первая сумма, ноль плюс ноль, дает ноль, а последняя, один плюс один, дает двоичное число два – 10.

На схеме Y1 и Y4 – схемы «И», Y2 – схема «ИЛИ», Y3 – инвертор. Используя эти базовые элементы мы получили устройство для сложения двух двоичных чисел от 0 плюс 0, до 1 плюс 1. Усложнив эту схему, думаю, можно получить схему сложения, имеющую возможность складывать любые числа с необходимым количеством разрядов.

Из-за того, что входные и выходные напряжения цифровых микросхем имеют только два значения, формируются сигналы, которые называют импульсными. Поэтому цифровые микросхемы могут успешно генерировать импульсы (и периодические их последовательности), преобразовывать импульсы – расширять их, укорачивать, формировать заданную длительность импульса, задерживать формирование импульса и т.д. В итоге с помощью цифровых микросхем создается огромнейшее количество схем, выполняющих разные полезные функции. С их помощью строятся цифровые процессоры – мозг любого компьютера. Думаю, правы были те, кто находился у истоков построения цифровой техники, назвав микросхемы цифровыми, а не числовыми, вы согласны?

Глава 3. Путешествие по плате с осциллографом

Г-н Прибор, который лежит в разобранном виде передо мной, не работает, потому я его и разобрал. Если «побродить» по выводам элементов его схемы щупом осциллографа, то, вероятнее всего, ничего интересного не увидишь. Но можно проверить работоспособность элементов, а если найдется что-то явно вышедшее из строя, попробовать заменить неисправный элемент. Так что, есть, мне кажется смысл, «побродить». Однако вначале, кому-то не интересно, но кому-то может показаться полезным, разберемся с тем

Что такое осциллограф?

Осциллограф, правильнее называть его осциллоскопом, но я (привык) буду называть его осциллографом, это изумительнейший прибор, который позволяет воочию наблюдать все, происходящее в схеме. Конечно, если наблюдать постоянное напряжение на резисторе (или любом другом элементе схемы), то удовольствия не больше, чем от просмотра рекламы по телевизору. Но современные осциллографы и в этом случае «на коне», их часто объединяют с мультиметром, и вы можете произвести измерения с необходимой точностью. А вот во всем, что связано с переменным напряжением, равным по наглядности этому прибору я не знаю. Даже самый примитивный осциллограф, о котором я собираюсь рассказать, вызывает у меня чувство гордости за принадлежность к роду человеческому, способному решать свои проблемы с такой великолепной изобретательностью.

Основным элементом традиционного осциллографа является специальным образом устроенная радиолампа. Стеклянный баллон, из которого выкачан воздух, имеет специальное покрытие на плоском торце, имеет, как любая радиолампа, катод, при нагреве излучающий носители тока (в данном случае электроны, которые «разогреты» настолько, что в своем тепловом движении улетают за пределы металла), анод, выполненный так, чтобы не задерживать электроны. Если между катодом и анодом приложить напряжение, подключив их к источнику ЭДС, то между ними появится электрическое поле, разгоняющее электроны, вылетевшие из катода. Разогнавшись, электроны пролетают мимо анода и попадают на плоский торец баллона, вызывая свечение покрывающего его вещества. Такую специальную лампу называют электронно-лучевой осциллографической трубкой. Устройство анода позволяет так сфокусировать поток электронов, что на торце, покрытом люминофором (специальным составом, светящимся под воздействием потока электронов), при включении трубки появится светящаяся точка.

Кроме этих электродов осциллографическая трубка имеет две пары взаимно перпендикулярно расположенных пластин. Если подать напряжение на одну пару (от источника ЭДС), то под воздействием электрического поля, возникающего между пластинами, поток электронов откланяется так, что точка из центра экрана осциллографа (торец с люминофором) переместится к его краю. К какому, зависит от полярности приложенного к пластинам напряжения. Если полярность напряжения поменять, то светящаяся точка переместится к противоположному краю. Если это напряжение подать на вторую пару пластин, то точка будет смещаться к верней и нижней кромке экрана. Первая пара называется горизонтальными отклоняющими пластинами, вторая пара – вертикальными.

Если мы подадим на горизонтальные отклоняющие пластины такое напряжение, чтобы светящаяся точка сместилась к левому краю экрана, затем плавно будем уменьшать это напряжение до нуля, используя делитель напряжения (потенциометр), то точка переместится от левого края экрана в его центр. Если теперь поменять полярность напряжения и плавно увеличивать его, то точка переместится к правому краю экрана.

Предположим, что мы настолько изобретательны, что можем создать источник питания, который в начальный момент времени дает напряжение одной полярности, плавно убывающее до нуля, затем плавно возрастающее в другой полярности, и резко возвращающееся к исходной полярности. И так раз за разом в постоянном ритме. Что мы увидим на экране? Точку, которая появляется у левого края экрана, плавно движется к правому краю, затем резко перемещается к левому краю, и так раз за разом. Если бы экран был бесконечен, то точку не надо было бы возвращать назад, а плавное движение светящейся точки по экрану мы можем, как вы уже сообразили, уподобить плавному равномерному течению времени.

Если на оставшуюся пару (вертикального отклонения) пластин подать исследуемое нами напряжение, то любое переменное напряжение будет наглядно представлено на экране осциллографа: в каждый момент времени светящаяся точка будет занимать такое вертикальное положение, которое соответствует мгновенному значению наблюдаемого напряжения.

Вот так, приблизительно, устроен простейший осциллограф. Он удобен для наблюдения периодических переменных напряжений. Для большего удобства период напряжения, подаваемого на горизонтальные отклоняющие пластины, его еще называют напряжением развертки, делают изменяемым. Для медленных процессов, этот период повторения может быть порядка секунд, для быстрых, как у радиосигналов, миллионных долей секунды. При быстрых повторениях развертки глаз уже не может разглядеть точку, он видит сплошную линию. На экран осциллографа для возможности оценки параметров наблюдаемого напряжения наносят сетку, имеющую деления, а переключатель времени периода развертки помечают значениями в сек/деление (или миллисекунда, микросекунда). Наблюдая, сколько делений занимает период исследуемого сигнала, мы можем оценить его в секундах, можем перевести период в частоту сигнала.

Напряжение, которое нужно подать на пластины вертикального отклонения, чтобы переместить светящуюся точку от нижнего края экрана к верхнему, может составлять сотни вольт. А сигналы, которые мы наблюдаем, могут иметь величину в единицы милливольт (тысячных долей вольта). Следовательно, перед подачей напряжения сигнала на пластины вертикального отклонения его нужно усилить. И, чтобы работать с разными наблюдаемыми напряжениями, усилитель можно дополнить делителем напряжения. Переключатель напряжения вертикального отклонения обычно маркируют в вольт/деление. Теперь мы можем определить и величину переменного напряжения.

Если вы начинающий любитель, и если вам позволяют финансы, обязательно обзаведитесь осциллографом. Сегодня можно купить и самый современный традиционный осциллограф, и сверх портативный, по размерам мультиметра, и совмещающий возможности мультиметра и осциллографа, и приставку к компьютеру, позволяющую использовать экран монитора, как экран осциллографа. Но не огорчайтесь, если не будет возможности обзавестись на первых порах осциллографом. Есть программы, превращающие звуковую карту компьютера, а они все имеют линейный вход, в осциллограф для наблюдения переменных напряжений звуковой частоты. Не с любой звуковой картой они работают, но это можно определить только испробовав программу. Кроме того, можно с успехом использовать такие программы, как PSIM и Qucs, чтобы увидеть все интересное, что есть в электрических схемах. А программа Multisim, которая стоит, однако, не дешевле очень неплохого осциллографа, может познакомить вас с работой реального осциллографа, у которого нажимаются кнопки, крутятся ручки, не хватает только описания работы с таким осциллографом, потому что потребуется время, чтобы разобраться, как с ним работать. Но, честное слово, это время будет потрачено не зря – осциллограф великолепнейшее изобретение, как, впрочем, любые измерительные приборы.

Что такое сигнал?

Договоримся, что любое переменное напряжение, которое мы можем (хотя бы в принципе) наблюдать на экране осциллографа, мы будем называть сигналом, который несет либо ранее заложенную в него информацию, либо информацию о состоянии точки наблюдения. Такая информация может быть использована для анализа состояния схемы в данной точке. Чтобы не быть голословным, я хочу проверить «жива» ли микросхема операционного усилителя на плате г-на Прибора, лежащего передо мной. Мне понадобится осциллограф. С реальной микросхемой я использую старенький С1-94, который у меня есть. Мне понадобится генератор низкой частоты, есть у меня и такой. А схему проверки я изображу в программе Qucs.

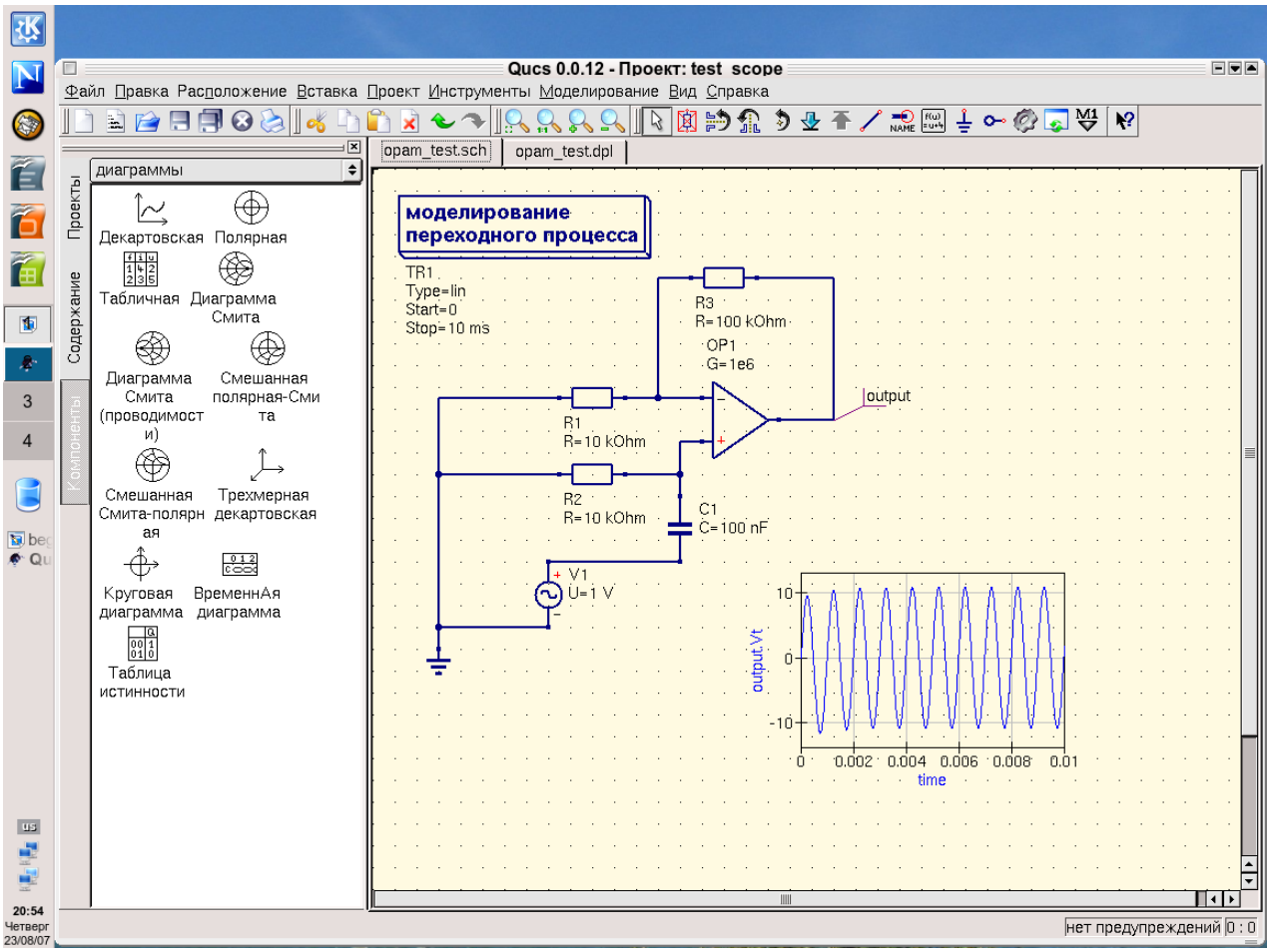


Рис. 3.1. Проверка работоспособности операционного усилителя

Прежде, чем проверять микросхему на плате, я зарядил батарейный отсек прибора, нужен же источник питания, и мультиметром проверил наличие питающих напряжений на выводах микросхемы. Если нет питающего напряжения, то и «живая» микросхема работать не будет. Конденсатор, на схеме С1, я подпаял, а то, что увидел на экране осциллографа ничем не отличается от того, что на рисунке. Генератор, на рисунке он изображен в виде источника переменного напряжения V1, создает синусоидальное напряжение, и это напряжение можно наблюдать на выходе схемы (метка output) с помощью осциллографа. Схема включения операционного усилителя позволяет быстро оценить коэффициент усиления. Для этого можно разделить величину резистора отрицательной обратной связи, на рисунке R3, на величину резистора на инверсном входе, R1. От генератора я подаю напряжение 1 В, на

выходе получаю искомые 10 В. Микросхема, эта микросхема, жива. Здесь, если оценивать усиление, следует вспомнить, что сопротивление конденсатора переменному току – это некоторая величина, образующая со входным сопротивлением усилителя делитель напряжения. Попробуйте заменить конденсатор C1 на другой, имеющий величину, положим 100 пФ (пикофарад), и посмотрите, что получится.

Раз уж я включил генератор и осциллограф, посмотрю, жива ли цифровая микросхема? Для ее проверки я использую тот же прием. Вначале проверю напряжение на выводах питания. Напряжение есть. Теперь, памятуя о таблицах истинности, и что логические элементы микросхемы двух-входовые, проверю, какое напряжение на входах (для схемы «И-НЕ» мне нужен высокий уровень), потом подам сигнал от генератора.

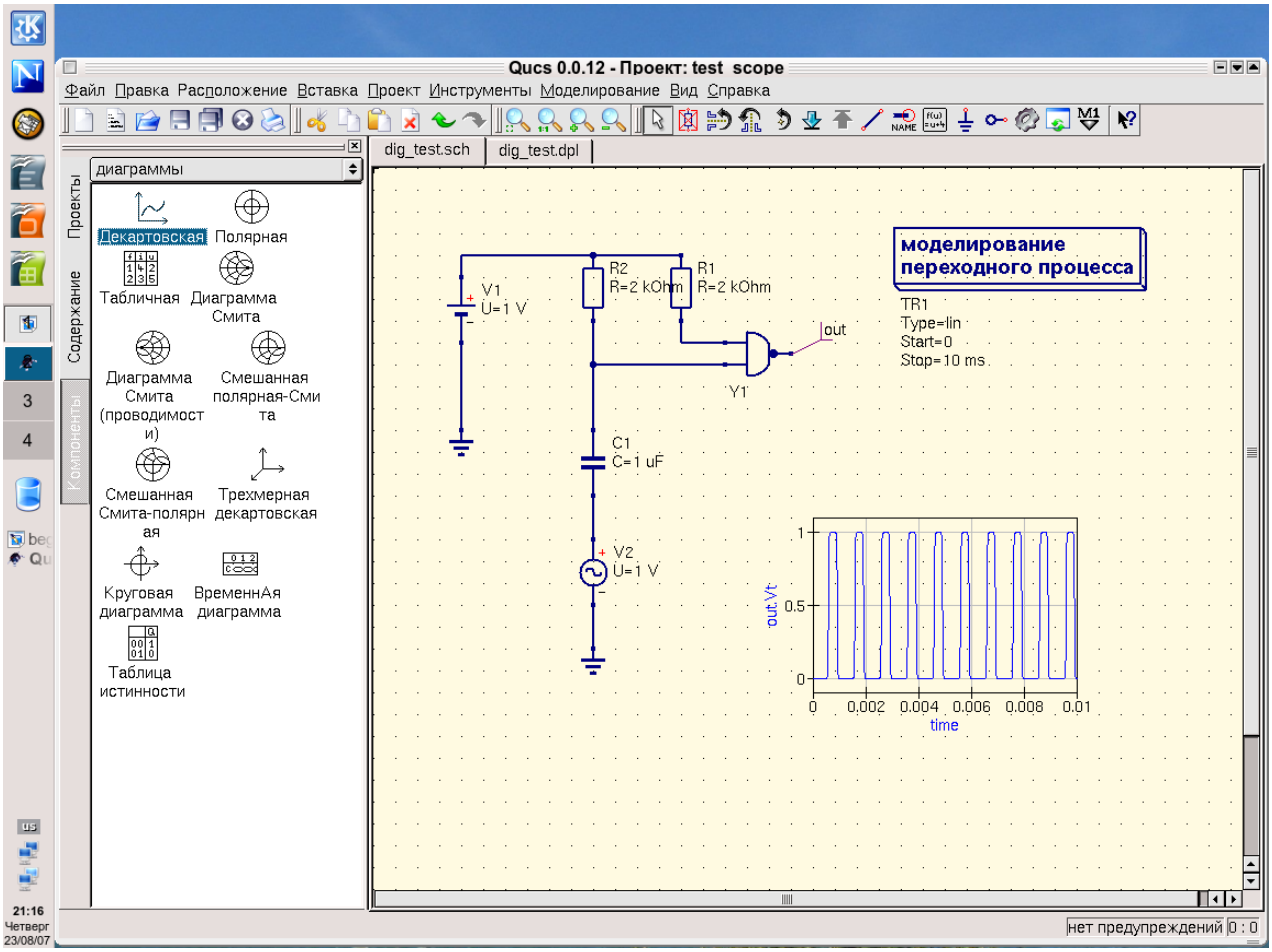


Рис. 3.2. Проверка работоспособности цифровой микросхемы

На рисунке я не стал менять напряжения источников постоянного и переменного напряжения, программа успешно работает с уровнем 1 В, а выходное напряжение реального генератора пришлось увеличить до 2.5 В. И конденсатор C1 пришлось заменить, существенно увеличив его значение. Но на экране осциллографа, подключенного к выходу микросхемы, я увидел ту же, практически, картинку, что показывает программа Qucs. Не все элементы, их у этой цифровой микросхемы четыре, имеют на входах высокие логические уровни, так что, не все я могу проверить с помощью генератора, даже переключив его в режим генерации прямоугольных импульсов. Вы, понимаете. Хотя есть, конечно, приемы, позволяющие проверить и работоспособность входов, которые получают от предыдущих элементов схемы сигнал в виде логического нуля. Есть специальный пробник, который очень коротким импульсом может изменить состояние и такого входа на состояние логической

единицы. Но собирать пробник мне хочется, хотя он совсем не сложен. Мне проще заменить микросхему, благо исправная такая же у меня есть. Но вернемся к сигналам.

Сигналы, которые я наблюдал на экране осциллографа, принесли мне информацию «о здоровье» микросхем. Радиосигнал, полученный приемником в вашем автомобиле, несет вам информацию о событиях в мире, развлекает вас музыкой и анекдотами, сообщает о пробках на дороге. Как выглядит радиосигнал? Здесь осциллограф мне пока не поможет, не разбирать же радиоприемник, да и тот у меня в часах-будильнике. Осциллограф не поможет, а программа Qucs поможет.

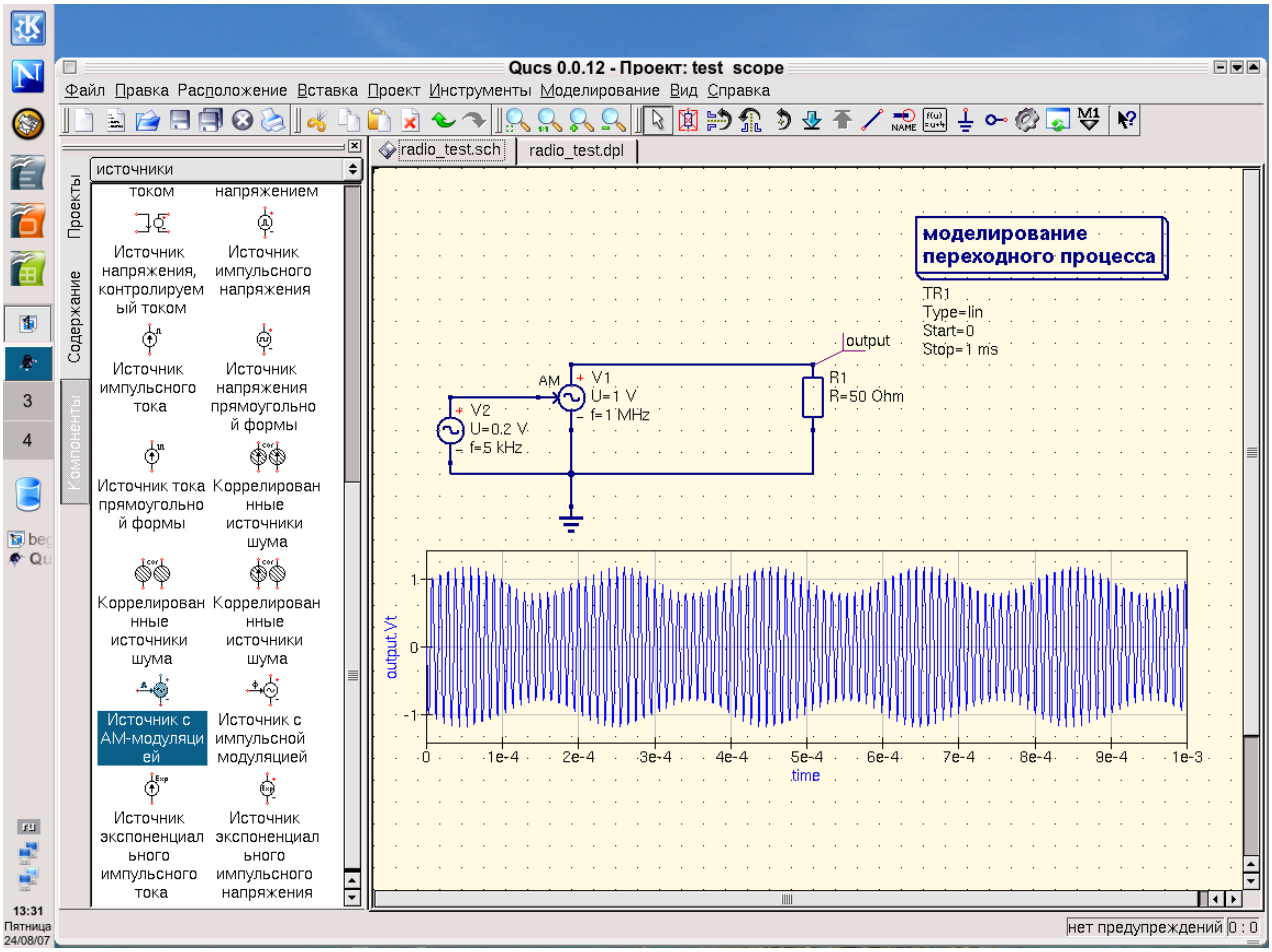


Рис. 3.3. Амплитудно-модулированный радиосигнал

В программе я выбрал на вкладке компонент в разделе источников *Источник с АМ-модуляцией* V1, для которой мне понадобился второй источник *Источник напряжения переменного тока* V2. Первый я настроил на частоту несущей 1 МГц, второй на частоту 5 кГц. А что такое несущая?

Когда я говорил о синусоидальном переменном напряжении, то привел формулу y=A*Sin(x). Функция синуса, как мы знаем, периодическая и изменяется от 0 до 1 по закону синуса. Напряжения, как мы знаем, бывают разной величины. Чтобы отобразить это я добавил к функции синуса множитель A – постоянную величину, некоторое число: 2, 10, любое. Эта постоянная величина изменила амплитуду, то есть, теперь наш синус будет меняться от 0 до A. Когда я рассказывал об осциллографе, то говорил, что горизонтальная развертка, как бы моделирует время, задавая его течение по оси x. Периодическая функция синуса означает, что есть некоторое время, называемое периодом, через которое вид функции

будет повторяться. Чем меньше это время, тем чаще (с большей частотой) повторяется вид функции. На рисунке 3.3 несущая частота (что-то вроде грузчика), определяемая источником V1, меняется часто, с частотой 1 МГц. Если бы источник V2 был источником постоянного напряжения, то мы увидели бы просто синусоиду. Множитель «А» был бы именно постоянной. Но, представим себе, что множитель А – это тоже функция, для определенности тоже синусоидальная, которая меняется значительно медленнее, чем несущая частота. Другими словами, ее период много больше, или ее частота много меньше. Тогда амплитуда несущей в каждый момент времени была бы равна значению этой функции, и амплитуда медленно-медленно менялась бы, как мы видим на рисунке 3.3. Эта переменная амплитуда содержит тот полезный сигнал, который нас интересует, когда мы слушаем радио. А быстро меняющееся напряжение источника V1 несет этот полезный сигнал через реки и поля, города и страны. Потому и называется несущей частотой. На эту частоту настроен ваш приемник. Конечно, речь диктора, рассказывающего о новостях, выглядит иначе, чем напряжение от источника V2, но это уже не столь важно, тем более, что если диктор «присвистнет» от удивления, то сигнал будет почти таким же. И еще. Для формирования информации мы воспользовались воздействием на амплитуду, поэтому такой вид формирования информации (модуляции, воздействия на модуль) называется амплитудной модуляцией. Сегодня чаще, пожалуй, применяют другой вид модуляции – частотную модуляцию.

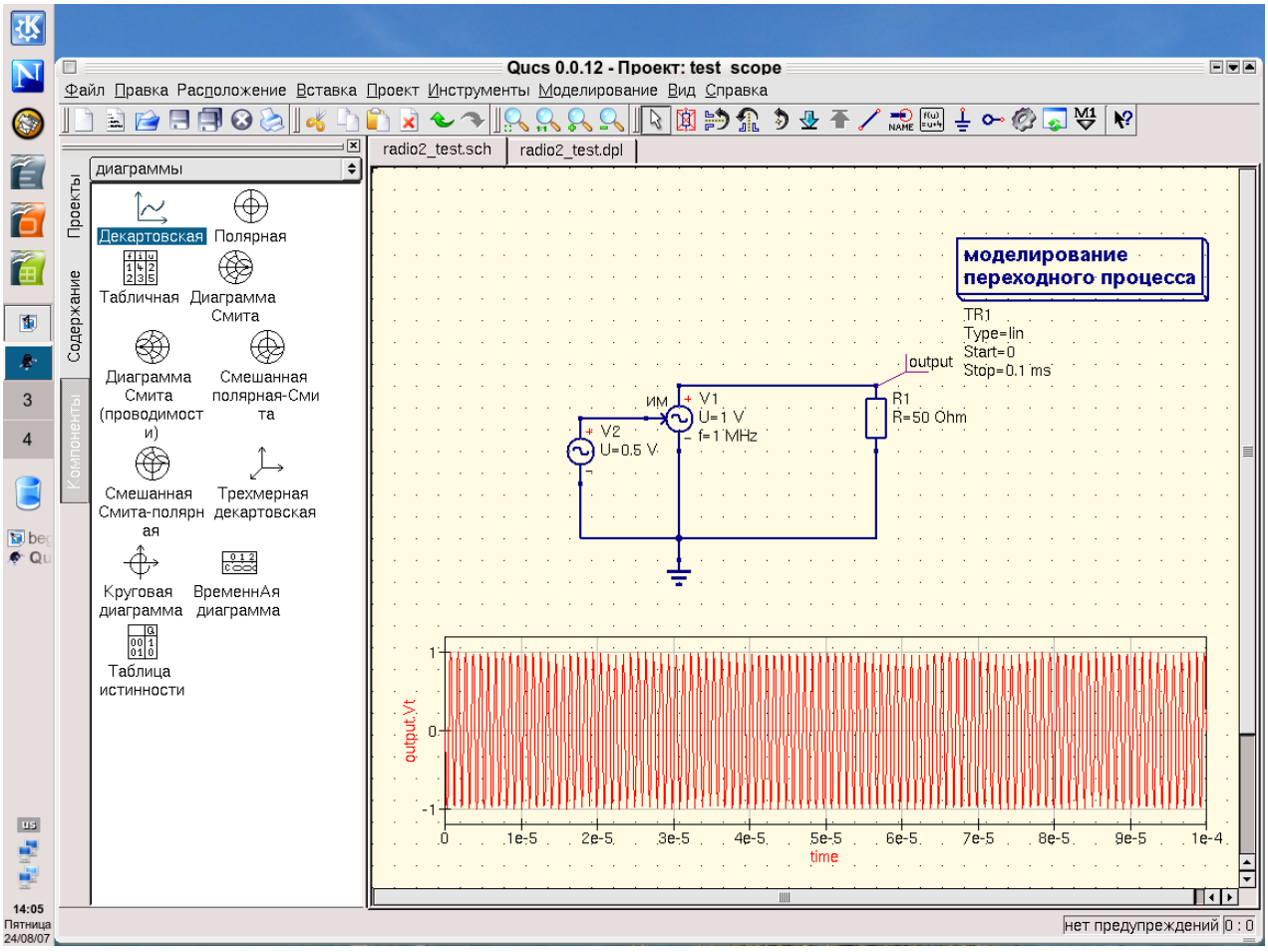


Рис. 3.4. Радиосигнал с частотной модуляцией

При таком способе модуляции частота несущей меняется в темпе информационного сигнала пропорционально его амплитуде. О преимуществах частотной модуляции и других видах мы поговорим, возможно, позже, как и о разных видах сигналов, применяемых в

электронике, а сейчас я хотел бы отметить, что многие генераторы испытательных сигналов производят синусоидальное переменное напряжение. Почему?

Видимо потому, что такой сигнал нельзя разложить на составляющие. Многие другие сигналы: прямоугольные, треугольные, пилообразные (как сигнал генератора развертки у осциллографа) можно представить как совокупность синусоидальных сигналов с разными частотами, амплитудами и фазами. В математике есть методы, разработанные математиком Фурье, которые позволяют разложить любую функцию в ряд Фурье, то есть, в совокупность других сигналов. Гармонический анализ Фурье (разложение на более простые функции) как раз использует функции синуса и косинуса в качестве «исходного материала». Поскольку я заговорил о пилообразном переменном напряжении, и поскольку я не привел ни одного рисунка «настоящего» осциллографа, я чувствую некоторую неловкость, и попробую загладить свою вину. Где-то в архиве, я думаю, у меня сохранилось что-нибудь подходящее.

Рис. 3.5. Вид переменного треугольного напряжения на экране осциллографа

Пилообразного напряжения не нашлось, но нашлось треугольное напряжение, каким его можно было бы увидеть на экране осциллографа фирмы Tektronix, если бы такой осциллограф был в вашем распоряжении. А картинка сделана в демонстрационной версии программы Multisim. Очень хорошая программа, но дорогая. Демонстрационная (вернее пробная) версия работает недолго, требует постоянной загрузки данных из Интернета при каждом включении, да и сама она достаточно «весомая». Но весьма наглядная. Осциллограф, изображенный на рисунке, программное творение, но позволяет нажимать все кнопки, вызывая те же эффекты, что у настоящего, позволяет крутить все ручки с настоящими эффектами, словом ведет себя, как настоящий. Если вам позволяют финансы обзавестись такой программой и таким осциллографом, они будут хорошими помощниками в вашей работе. Но, если этого не произойдет, скажу, знавал я тех, у кого все было, но не было ни умения, ни времени, а может воли, для того чтобы эти красивые вещи применить к своему удовольствию. Знавал я и тех, кто даже без приличного тестера с огромным удовольствием делал интереснейшие вещи на зависть окружающим. Думаю, желание и терпение важнее дорогостоящих приборов, а самодельные приборы не только многому вас научат, пока вы их делаете, но будут работать не хуже покупных. Здесь важно понять, что вы хотите от радиолюбительства, получить электронное устройство, или получить удовольствие от работы. Подумайте, прежде, чем начать что-то покупать.

Что я увидел на плате с помощью осциллографа?

Поскольку схему г-на Прибора, лежащего предо мной, я уже отыскал, попробую по схеме сориентироваться, чтобы такое интересное посмотреть осциллографом. Вообще, схема – дело первейшее. Я очень уважаю схемы, видимо оттого, что приходилось часто работать с устройствами в отсутствии схемы. Такое, может быть, даже полезно, но очень утомительно. Вспомнилась давняя история. Мой непосредственный руководитель попросил посмотреть, если не ошибаюсь, уж очень давно было, называлось это устройство «Элетап», и было, в сущности сегодняшним телефоном, но скорее перевозным, чем переносным, он попросил посмотреть, отчего не работает это устройство? Схема аппарата была, но с «черным ящиком». Вся сущность устройства была в этом черном ящике, а его схемы не было. Я позвонил своему доброму знакомому, но у него тоже не было схемы. Конечно, я разобрал все, уж это я умел тогда делать, попробовал включить аппарат, и убедился, что он работает нормально. Я собрал его, это не всегда, но иногда удается, он работал. Пришло время отдать это чудо техники, но именно в тот момент устройство вновь перестало работать. Тут уж я разозлился по-настоящему. Договорившись с начальником, я решил нарисовать схему электронного блока: с десяток плат, на каждой из которых с десяток цифровых микросхем. Да, рисование помогло определить структуру (функциональную схему) и принципиальную схему аппарата, помогло выявить неисправную микросхему, при небольшом нагреве она переставала работать, а «поостыв» великолепно работала, но этот опыт «рисования» заставил меня очень уважительно относиться к схемам.

Чтобы не возвращаться к рассказу о том, как выглядит осциллограф, хочу привести еще один вид, вид программы для работы с осциллографом-приставкой к компьютеру. Такая приставка имеет свои преимущества, она позволяет записать сигналы, которые можно позже просмотреть и проанализировать. Это особенно важно, если вы имеете дело с короткими и непериодическими импульсами, такие сигналы вы получите при исследовании сигналов управления пультов (инфракрасные пульты) многих бытовых устройств, и телевизоров, и музыкальных центров и т.п., а также при проверке сигналов в линии RS232, и аналогичных им. Экран такого осциллографа соизмерим с экраном монитора, а это всегда соблазнительно. Есть и еще одно преимущество, в цене. Если купить приставку-конструктор, то обойдется это, примерно, в 3000 руб.

Не могу сказать со всей определенностью, поддерживает ли такой конструктор все заложенные в программу функции, но если поддерживает, то кроме осциллографа вы получите, например, анализатор спектра — еще один очень полезный прибор при работе с высококачественными усилителями. Очень трудно налаживать усилитель без возможности проверить вносимые им нелинейные искажения.

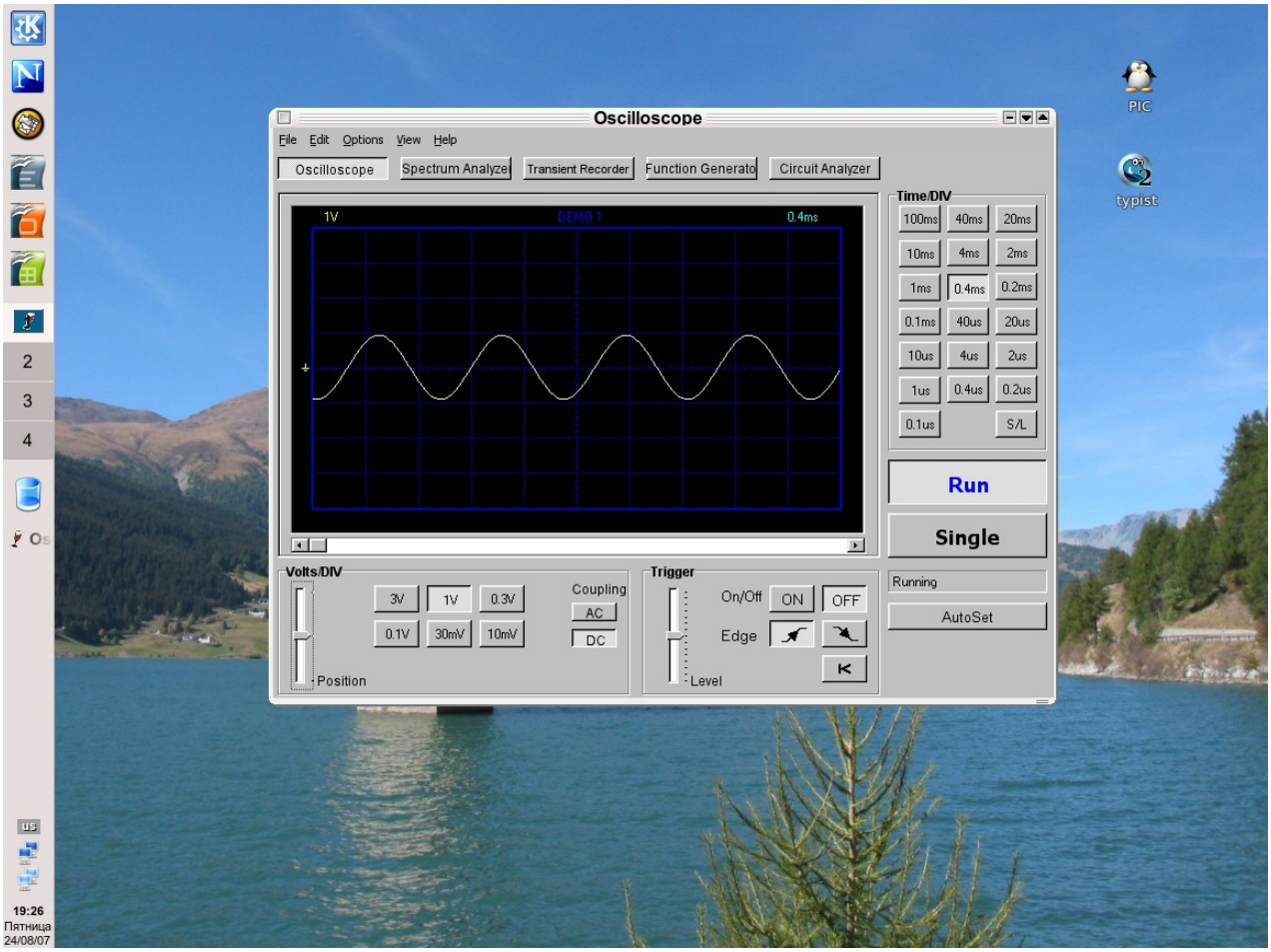


Рис. 3.6. Вид программы для обслуживания осциллографа-приставки

Итак. Схема прибора подсказывает мне, что в нем на цифровой микросхеме собран генератор прямоугольных импульсов. Генераторы на цифровых микросхемах в цифровых устройствах реализуют достаточно часто, их используют как тактовые или синхрогенераторы. Есть, по меньшей мере, несколько разновидностей типовых генераторов, и множество схем генераторов со специальными параметрами. Причина, по которой логические элементы цифровых микросхем позволяют использовать их в качестве активных элементов генераторов, в том, что они остаются, хотя и специализированными, но усилителями. Если вход микросхемы ТТЛ с помощью делителя напряжения привести к запрещенному для логических элементов уровню в 1.2-1.5 В, то логический элемент превращается в усилитель напряжения. Если такой усилитель охватить положительной обратной связью, то возникает самовозбуждение усилителя, превращающее его в генератор. Генератор в приборе вырабатывает импульсы, которые через усилитель мощности приходят на измерительную цепь, и эти же импульсы управляют, с помощью ключей на полевых транзисторах, процессом измерения. В первую очередь есть смысл посмотреть, а работает ли генератор? Схема генератора выглядит так.

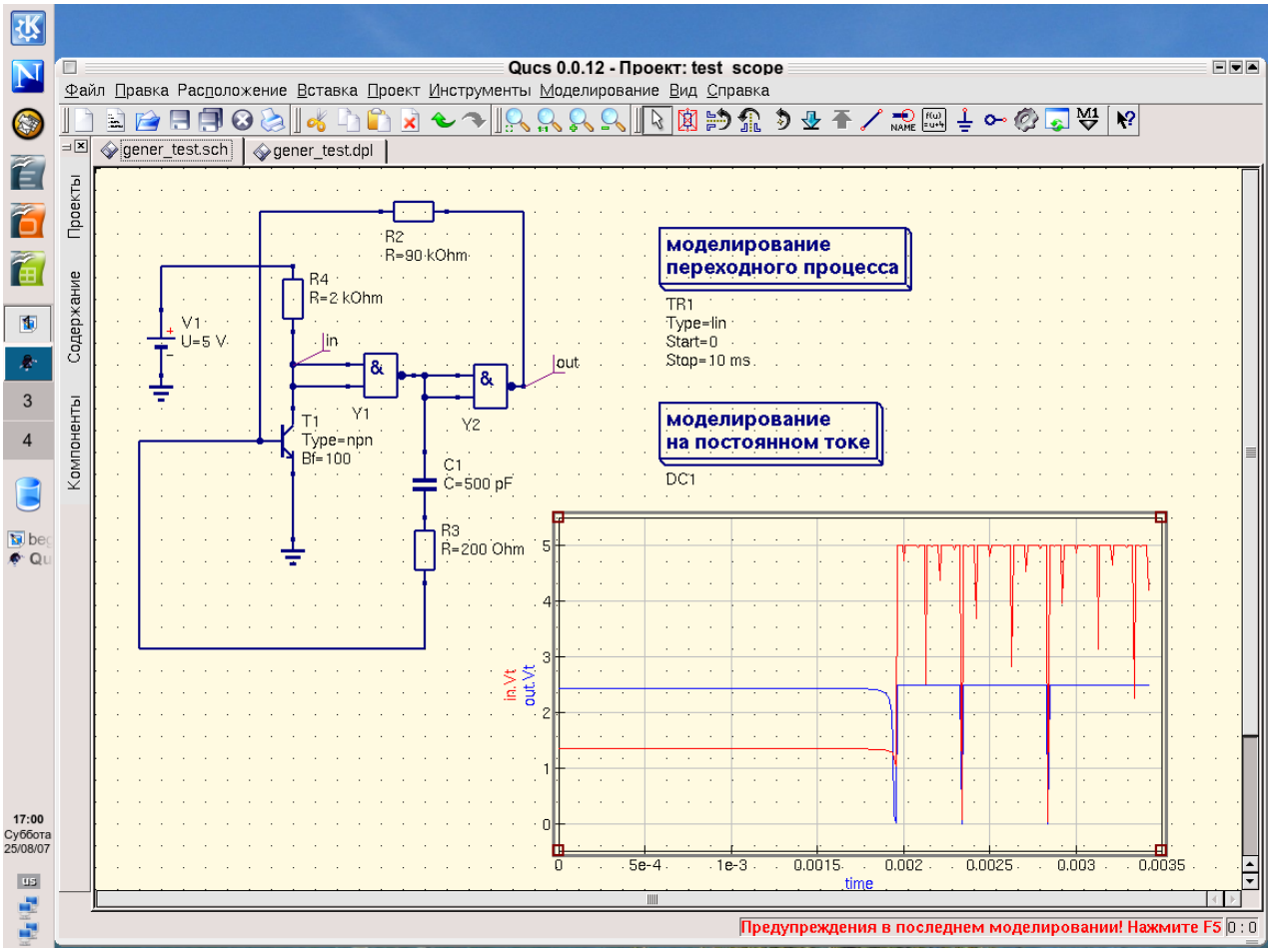


Рис. 3.7. Схема генератора на логических вентилях

Осциллограмма в программе Qucs не совсем то, что должно происходить. Хотя я пытался подбирать величины элементов схемы, конденсатор и резисторы, пытался поменять свойства вентилей, чтобы обнаружить что-то похожее на генерацию импульсов, я не удовлетворен результатом. По правде говоря, я не ожидал даже такого эффекта, поскольку от программы, использующей логические вентили, НЕЛОГИЧНО ожидать, что они будут работать вне логических отношений. Кстати, меняя свойства логических вентилей, я заменил графическое изображение. Схема «И-НЕ» сейчас выглядит более привычно, чем в предыдущем начертании.

И все-таки хочется увидеть работу именно генератора импульсов. У меня есть такая возможность – использовать другую программу САПР, но о разных программах и их возможностях я уже написал в книге «Наглядная электроника», не повторять же все еще раз. Попробую поступить иначе. Если открыть справочник по цифровым микросхемам, то в нем можно найти схему базового элемента «И-НЕ» серии ТТЛ. Эта схема имеет все знакомые нам элементы: транзисторы, резисторы, диод. А что если повторить эту базовую схему, с которой пока только одна проблема, первый транзистор имеет специальную конструкцию, он многоэмиттерный. Но эти эмиттеры образуют входы многовходовых элементов, а мы используем, в сущности, один единственный. Попробуем использовать обычный транзистор. Значения резисторов в схеме не приводится, попробуем подобрать их.

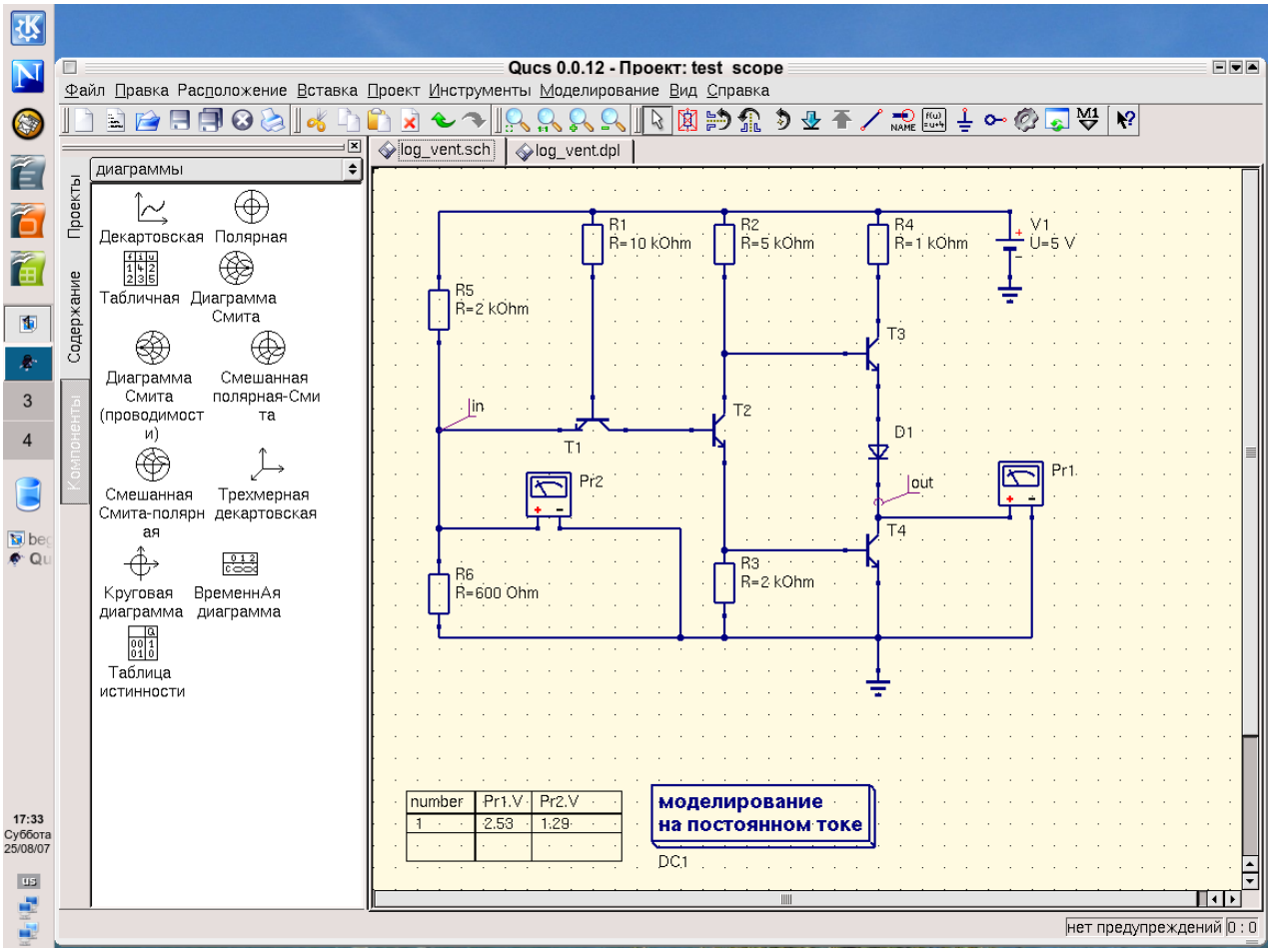


Рис. 3.8. Схема базового элемента «И-НЕ» ТТЛ на обычных компонентах

Я добавил резисторы R5 и R6 для подачи на вход напряжения близкого к 1.5 В и измерители напряжения Pr1, для измерения напряжения выхода, и Pr2, показывающий напряжение на входе. При тех значениях элементов схемы, которые есть, напряжения близки к моему представлению о том, какими они должны быть. Остается проверить, удалив из схемы не рисунке 3.8 резистор R6, что соответствует подаче на вход логической «1», напряжение на выходе. Оно близко к нулевому. И проверить соответствие при подключении входа к земле, что равносильно подаче логического «0» на вход. Напряжение на выходе близко к питающему, что тоже вполне похоже на нормальную работу логического инвертора.

Чтобы подтвердить предположение о том, что логический вентиль остается усилителем, я использую генератор синусоидального напряжения с частотой 1 кГц и выходным напряжением 0.1 В, который через конденсатор 100 мкФ подключу ко входу инвертора с делителем питающего напряжения на резисторах R5 и R6. Результат можно наблюдать на экране осциллографа. Логические инверторы есть во многих сериях цифровых микросхем. В отечественной серии К155 (аналогичной серии SN74) микросхема К155ЛН1 содержит шесть логических инверторов в одном корпусе (аналогично, SN7404).

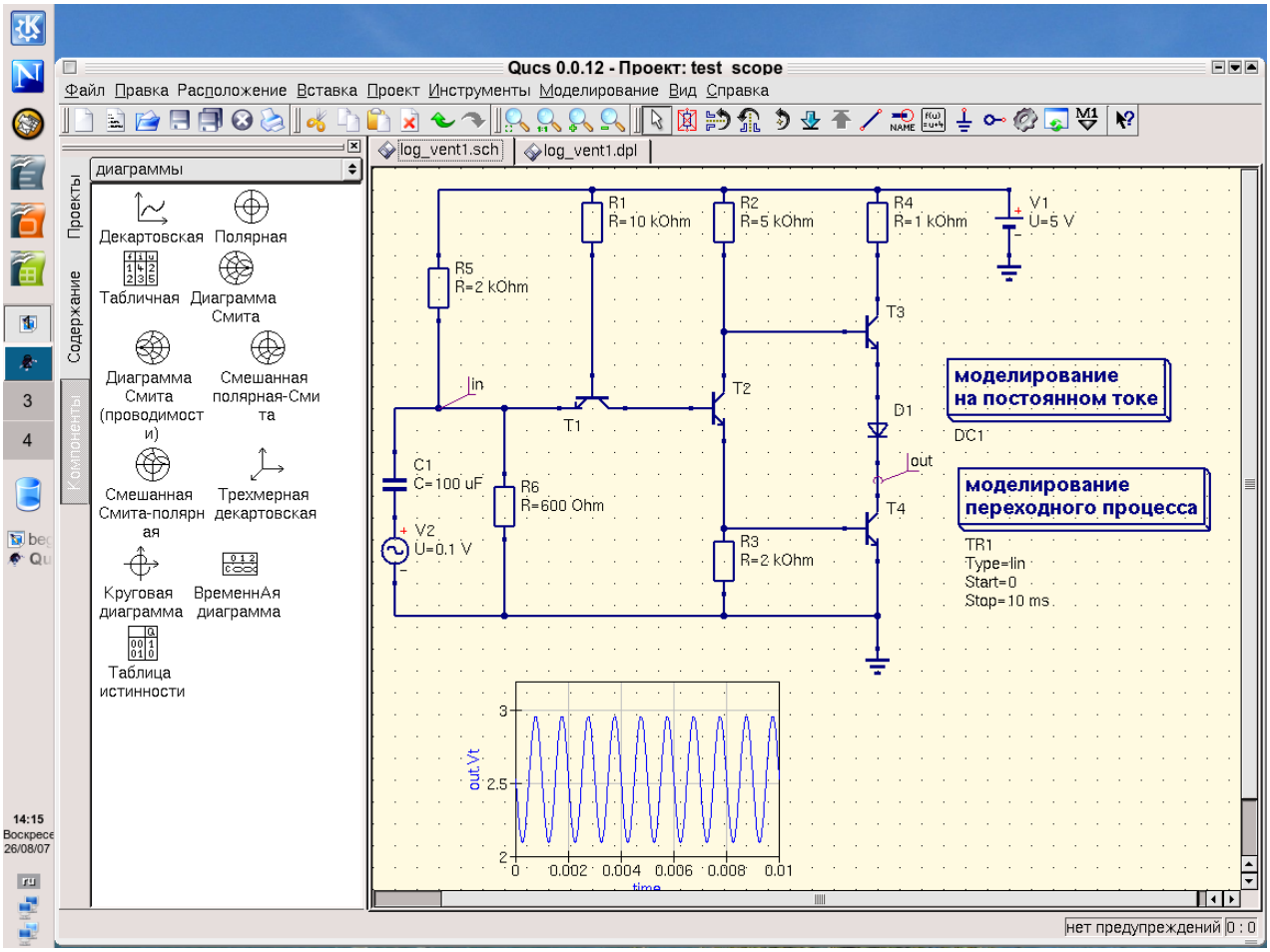


Рис. 3.9. Проверка логического вентиля в усилительном режиме

При входном напряжении 0.1 В схема обеспечивает усиление по напряжению около 5, а форма сигнала по виду не отличается от синусоидального.

Рассуждения мои казались мне правильными, но добиться работы схемы, изображенной на рисунке 3.7, даже после замены логических вентилей на их транзисторный эквивалент в программе Qucs не получилось.

Мне хотелось повторить схему генератора, как она нарисована, но не получилось, возможно из-за того, что микросхема К561ЛА7 выполнена по КМОП технологии, а повторить схему я пытался с ТТЛ микросхемой. Чтобы показать, что я увидел на выходе задающего генератора после замены транзистора и микросхемы на плате прибора, мне пришлось нарисовать типовую схему синхрогенератора на цифровых микросхемах. В реальном приборе задающий генератор снабжен еще одной цифровой микросхемой, в корпусе которой два D-триггера. Триггер формирует хорошие, с крутыми фронтами, прямоугольные импульсы, и служит для коммутации ключей с использованием прямого и инверсного выходов триггера.

Мне жаль, что не получилось все задуманное. Такое бывает. Хотя схема на рисунке 3.8, без вспомогательных элементов, соответствует схеме базового элемента ТТЛ, параметры транзисторов и номиналы резисторов, использованные мной, могут очень сильно отличаться от реальных, что, в конечном счете, тоже могло стать причиной неудачи. Однако, что теперь гадать?

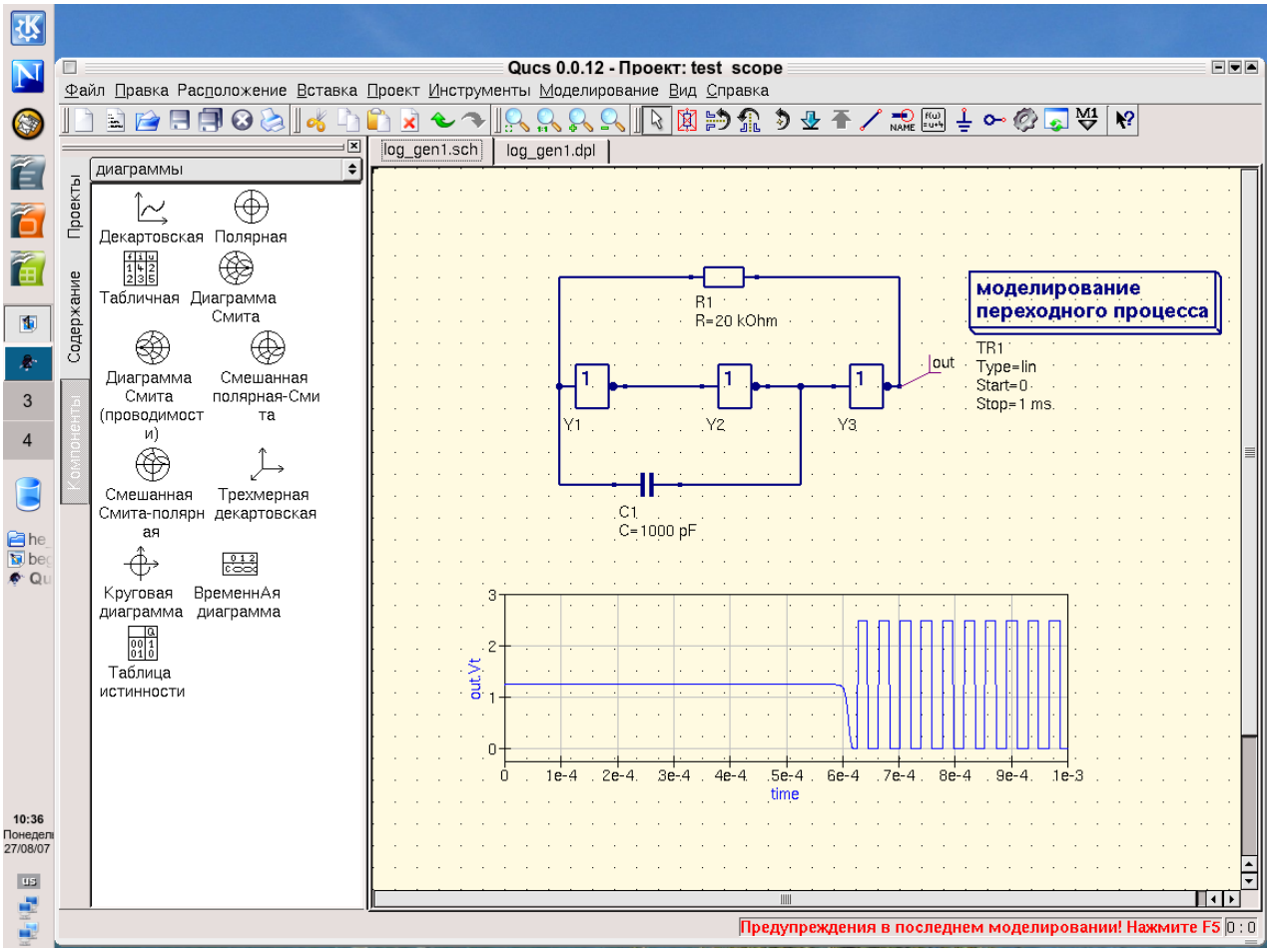


Рис. 3.10. Типовая схема синхрогенератора на цифровых микросхемах

Вот такого вида сигнал задающего генератора используется в приборе. Как я объяснял выше, я был готов к тому, что не получу положительного результата от эксперимента в программе Qucs. Логике – логическое, программе – программное, реальной микросхеме – все ее возможности. Тем приятнее, что с типовой схемой программа Qucs не подвела.

Что ж, все, что могло представлять интерес в смысле работы с осциллографом, мы в разобранном мною устройстве посмотрели. Остальное может представлять интерес только и тогда, когда необходимо восстанавливать работоспособность прибора. В приборе использована очень интересная схема выпрямителя, но это уже из мира схемотехники, и не относится к наблюдениям с помощью осциллографа за работой электрических компонент устройства. Выпрямитель в приборе используется для обслуживания микроамперметра, со шкалы которого считывается измеряемая величина. Микроамперметр – устройство магнитоэлектрическое. А не пора ли немного поговорить о магнитных проявлениях электричества?

Глава 4. Электричество и магнетизм

Ученые, исследовавшие электрические явления, заметили, что проводник с током окружает поле, природа и характер которого отличается от электростатического поля, ответственного за взаимодействие двух заряженных объектов. Это поле, возникающее при движении электрических зарядов, назвали магнитным. Всем с детства знакомы магнитики, которые так ловко прилипают к металлическим дверям холодильника. Если два магнита сближать друг с другом, то при одном их положении они будут притягиваться, при другом отталкиваться. Кто не играл с ними в детстве! А их свойство, если магнит выполнен в виде тонкого легкого стерженька, показывать направления на северный и южный полюса Земли использовали давным-давно для навигации – поиска правильного пути в путешествиях.

Свойство магнитного поля, появляющегося вокруг проводника с током, давно используется в электрических устройствах. Хотя у г-на Прибора их представительство не столь весомо, но один из них есть – микроамперметр. Я уже говорил, что этот стрелочный прибор представляет собой рамку с намотанным на нее проводом, которая помещена в поле постоянного магнита. Дело в том, что любой проводник с током, помещенный в поле постоянного магнита, будет двигаться в этом поле. Выключается ток, движение прекращается. Со стороны магнитного поля на проводник с током действует сила. Подобное проявление свойств электричества используется не только в измерительных приборах. Так же работает громкоговоритель. Он имеет диффузор, к которому приклеена катушка с проводом, помещенная в сильное магнитное поле, создаваемое постоянным магнитом. Когда через катушку протекает переменный ток, катушка движется в магнитном поле, а характер ее движения определяется законом, по которому изменяется переменный ток. Свое движение катушка с проводом передает диффузору громкоговорителя, заставляя его двигаться в такт со своим движением. Диффузор громкоговорителя при своем движении создает разряжения и уплотнения воздуха, а это не что иное, как звук. Так громкоговоритель преобразует переменное напряжение в звук.

Интересен и такой момент: если на проводник с током в магнитном поле действует сила, заставляющая проводник двигаться, то на концах обесточенного проводника, если перемещать его в магнитном поле, появляется ЭДС. Если наш громкоговоритель отключить от выхода схемы усилителя, включить его на вход, то при достаточной чувствительности усилителя можно убедиться, что громкоговоритель работает, как микрофон – устройство, преобразующее звук в переменное напряжение. Действительно, если вам приходилось разбирать электродинамические микрофоны, то вы знаете, что легкая подвижная мембрана такого микрофона имеет катушку с проводом, помещенную в поле постоянного магнита. Почти как диффузор громкоговорителя. Звуковая волна – череда разряжений и уплотнений воздуха – доходя до мембраны микрофона, заставляет ее двигаться, а с ней и катушку с проводом, прикрепленную к мембране. Катушка в поле постоянного магнита движется по тому же закону, что и звуковая волна, отчего появляющееся на выводах катушки переменное напряжение повторяет закон изменения звука. Микрофон преобразует звук в переменное напряжение.

Громкоговорители и микрофоны – не единственные элементы, где используются магнитные поля. Катушка с проводом, по которому течет ток, снабженная металлическими сердечником, скобой и коромыслом, будет притягивать коромысло, которое вторым концом при движении может замыкать контакты. Так получится реле постоянного тока. Реле – элемент электрической схемы, который по сей день находит широкое применение, хотя появились и успешно с ним конкурируют бесконтактные коммутаторы.

Еще один интересный аспект: если вокруг проводника с током возникает магнитное поле,

то любой движущийся проводник без тока, находящийся на любом расстоянии от первого, будет чувствовать это магнитное поле, и в проводнике возникнет электрическое поле, создающее в нем ЭДС. Значит электрическое и магнитное поля связаны, и мы можем говорит об электромагнитном поле. При движении проводника в магнитном поле в нем образуется ЭДС, а что произойдет, если мы будем не двигать проводник, а станем менять магнитное поле? Мы получим в обесточенном проводнике ЭДС. На любом расстоянии от проводника, создающего это меняющееся магнитное поле. То есть, мы можем рассматривать проводник с переменным током, как генератор электромагнитного поля уходящего далеко в пространство, о чем говорят так: переменный ток создает электромагнитную волну в пространстве. Поместив на ее пути проводник, мы обнаруживаем эту электромагнитную волну. Конечно, чем дальше от источника размещен наш приемник, тем меньше возникающая ЭДС. Но это не помешает нам так устроить приемник и передатчик электромагнитной волны, чтобы иметь возможность передавать полезную информацию без проводов. Примерно так работает радио. Чем мощней передатчик, чем чувствительней приемник, чем правильнее выбрана частота переменного тока передатчика, тем дальше от передатчика можно принять радиосигнал. Мы относительно недавно стали пользоваться электромагнитными полями для передачи информации, но сегодня трудно себе представить, как мы могли без этого обходиться раньше.

Но вернемся к реле. Чем реле нам интересно? Когда мы говорили о транзисторном усилителе, то польза от него была несомненной. Транзисторный усилитель позволял усиливать сигнал, несущий полезную информацию, по току, по напряжению, а, значит, и по мощности. А что реле? Реле тоже способно усиливать. Магнитное поле, создаваемое обмоткой реле, требует, в зависимости от конструкции реле, подачи на обмотку напряжения в несколько вольт, а ток, протекающий по обмотке, может составлять несколько десятков миллиампер. Но при этом контакты реле могут коммутировать напряжение в несколько десятков (а то и сотен) вольт, пропуская ток в несколько ампер (а то и десятков ампер). Чем не усилитель? Есть одно неудобство, переменное напряжение сигнала, усиливаемого транзистором, может меняться по любому нужному нам закону. А сигнал, усиливаемый с помощью реле, всегда импульсный. Проведем эксперимент с реле. Для реального эксперимента потребуется реле постоянного тока с рабочим напряжением 5-10 В, источник такого напряжения (батарейка или блок питания), выключатель и мультиметр. Но эксперимент настолько прост, что проводить его на макетной плате, пожалуй, нет смысла. Не всегда у начинающего любителя найдется подходящее реле, а если покупать реле, то не бесспорно, что удастся использовать его в последствии. Для любой полезной схемы реле выбирается с конкретными параметрами: рабочее напряжение, потребляемый ток, рабочее напряжение и ток, коммутируемый контактами реле, количество групп контактов. В некоторых случаях определяющими параметрами могут быть габариты или вес реле, или тип реле, хотя бы с точки зрения срока службы схемы. Последнее может зависеть от характера работы реле. Любое реле имеет срок службы, определяемый количеством срабатываний. Например, реле имеет допустимое количество срабатываний 1000. Много это или мало? Если вы с помощью реле включаете свет в комнате, и делаете это 5 раз в день, то реле обязательно прослужит вам не менее года. Может прослужить и много дольше, но это уж как повезет. Но, если вы хотите использовать реле в качестве преобразователя постоянного напряжения в переменное (реле можно использовать и в таком качестве), то при частоте переключения 200 Гц данное реле безусловно прослужит не менее 5 секунд. Конечно, многие реле имеют большее на порядок или несколько порядков гарантированное количество срабатываний, но то, какое выбрать реле, определяется конкретной схемой, поэтому закупать реле впрок, смысла нет. Будем считать, что эксперимент у нас «мысленный»!

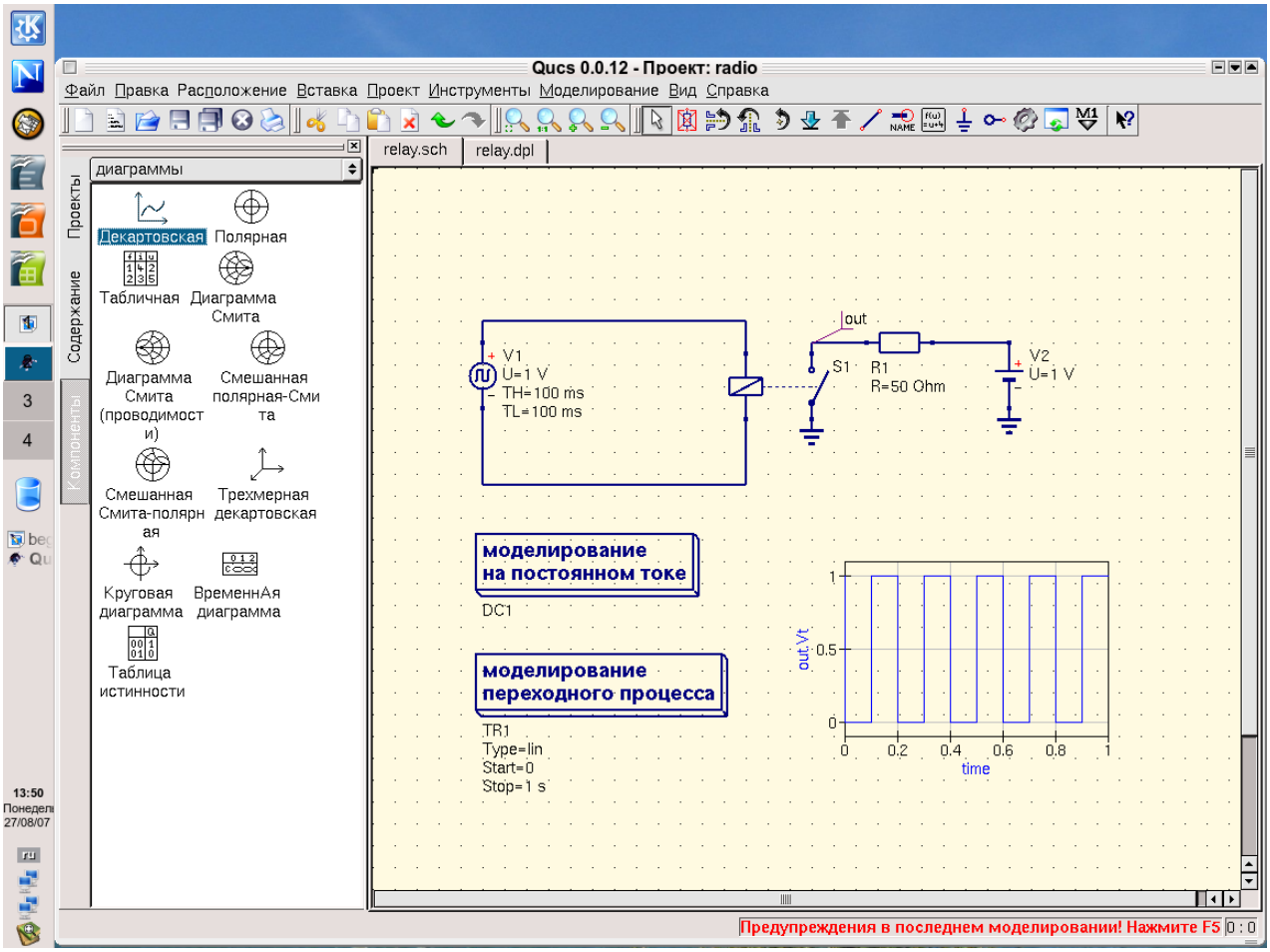


Рис. 4.1. Эксперимент с реле постоянного тока

Здесь в качестве сигнала используется генератор прямоугольных импульсов, а осциллограмма показывает сигнал в точке *out*. Реле повторяет форму сигнала генератора, но ток, напряжение и мощность сигнала в точке *out* могут быть много больше исходного.

Может возникнуть вопрос, а что полезного может нести такой сигнал? Может нести информацию, которую мы можем передать, например, по проводам, подключенным к контакту S1. Если вместо генератора прямоугольных импульсов мы используем батарейку с выключателем, выключатель, включенный кратковременно будет передавать точку, а включенный подольше, тире, то, как сами понимаете, мы можем использовать азбуку Морзе для передачи полезной информации на расстояние по проводам. А если вместо резистора R1 и батарейки V2 к контактам реле подключим мощный генератор радиочастоты, то сможем передавать полезную информацию и без проводов. Все в наших руках.

Конечно, реле сегодня не используется для подобных целей, но возможность «усиливать» сигнал, то есть, коммутировать большие токи и напряжения используется и по сегодняшний день. У реле перед другими коммутаторами есть ряд преимуществ. Любой коммутатор, транзисторный ли, тиристорный ли, рассеивает достаточно большую мощность. Вспомним закон Ома и то что рассеиваемая мощность равна произведению падения напряжения на коммутаторе на протекающий через него ток. Если на транзисторе при токе в 1 А падает напряжение в 1 В, то рассеивается мощность в 1 Вт. А при токе в 10 А! Транзистор придется охлаждать, иначе он расплавится. Сопротивление контактов, даже не самых удачных, порядка 0.01 Ом. При токе в 10 А падение напряжения 0.1 В. Рассеиваемая мощность 1 Вт. И нагрев

металлических контактов при такой рассеиваемой мощности их никоим образом не расплавит.

Очень часто реле постоянного тока включают в цепь коллектора транзистора. Входной ток транзистора может быть порядка сотен микроампер, а контакты реле могут коммутировать ток в десятки ампер. Десять ампер, деленные на сто микроампер дадут коэффициент усиления по току в сто тысяч раз. Всего два компонента электрической схемы, а такой впечатляющий результат!

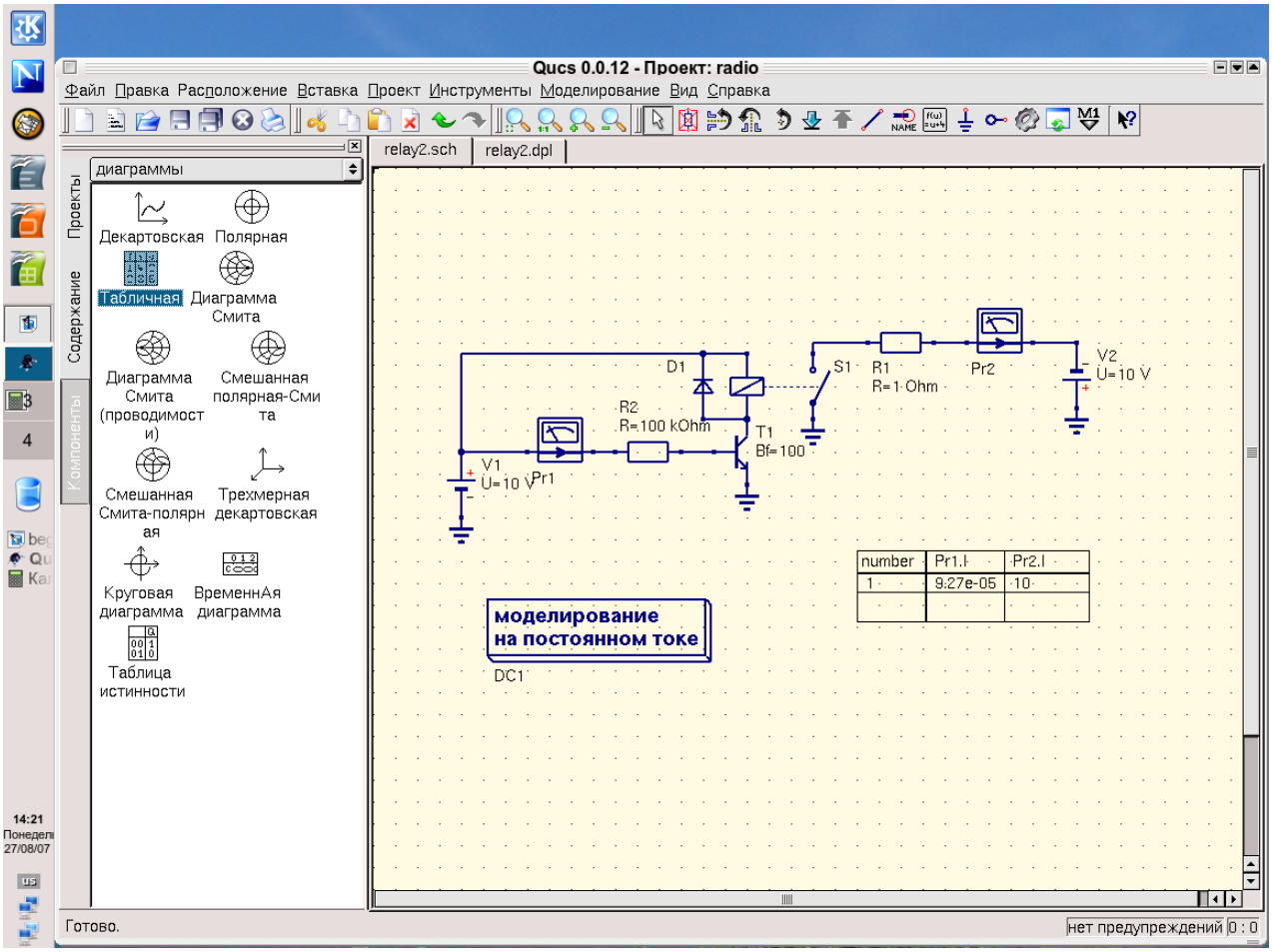


Рис. 4.2. Включение реле в цепь коллектора транзистора

Измерители тока Pr1 и Pr2 позволяют оценить возможности реле в усилении тока. Очень советую повторить подобный эксперимент без участия реле (лучше в программе Qucs), включив резистор R1 с сопротивлением равным сопротивлению обмотки реле в коллекторную цепь транзистора и сравнить токи на входе и выходе схемы.

На схеме рисунка 4.2 я добавил параллельно реле диод D1. Зачем я это сделал? Чтобы вы не забывали включать диод (встречно) параллельно обмотке реле, даже если и без него все работает хорошо. Обмотка реле, как любая индуктивность, имеет реактивных характер сопротивления, и как для конденсатора, о чем я говорил раньше, для индуктивности существует реакция при включении источника ЭДС. Это время реакции очень мало, а процесс, протекающий при этом называют переходным процессом, так вот переходной процесс для индуктивности выглядит обратно тому, что происходит с конденсатором. В начальное мгновение после подключения индуктивности к источнику постоянного напряжения ток через индуктивность очень мал, но начинает быстро нарастать, пока не

достигнет величины, определяемой активным сопротивлением провода катушки. Такая есть реакция у индуктивности. Но самое интересное, что происходит, когда мы отключаем источник напряжения. Магнитное поле, созданное катушкой, не исчезает сразу, а быстро убывает. Но в проводнике, помещенном в меняющееся магнитное поле, появляется ЭДС. В катушке эта ЭДС будет направлена так, что она будет противоположна по знаку падению напряжения при включенной катушке. Поскольку в схеме на рисунке 4.2 выключение реле будет производится не отключением источника питания, а обрывом базовой цепи транзистора (точнее подсоединением ее к земле), то между эмиттером и коллектором транзистора оказываются включены два источника напряжения: батарейка V1 и противоэдс (так называют реактивную ЭДС катушки) обмотки реле. Суммарное напряжение может оказаться больше предельно допустимого напряжения (есть такой параметр у транзистора) на коллекторе транзистора, что может вывести его из строя. Чтобы этого не происходило параллельно обмотке реле включают диод. Возникающая при выключении реле противоэдс подключена к диоду и напряжение ее не превышает напряжения включенного в прямом направлении диода, а это 0.5-0.7 В. Ток через диод ограничивается активным сопротивлением обмотки реле и не будет слишком большим.

До появления полупроводниковых приборов реле применяли очень широко. Даже вычислительные устройства, ранние цифровые компьютеры, использовали реле. От тех времен в современном компьютерном лексиконе даже сохранилось такое понятие, как «баг», ошибка (bug – жук, насекомое). Когда в контакты реле забиралось насекомое, мешая работе компьютера, приходилось долго отыскивать этого любителя вычислительной техники, осуществляя «дебагинг» при отладке программы. Сегодня любая среда программирования в память о событиях тех лет имеет средства «дебагинга» программ.

Реле настолько простое и настолько неприхотливое электромеханическое устройство, что и сегодня, создавая свои схемы, не следует о нем забывать. Во многих случаях самым простым и надежным решением будет применение реле. Помимо выгоды от использования свойств контактов реле для коммутации высоковольтных потребителей, есть то, что особенно удобно для начинающих — схему можно полностью отладить без использования опасного для жизни напряжения. Работа даже с бытовым силовым напряжением, даже для профессионалов, требует неукоснительного соблюдения правил безопасности и существенного опыта работы, а с учетом обычной увлеченности любителя при налаживании схемы, трудно ожидать от него выполнения всех правил работы под напряжением. Использование реле позволяет избежать проблем — достаточно проверить, замкнуты или разомкнуты контакты реле, что можно сделать мультиметром, включенным в режим измерения сопротивления.

Что же касается упрощения схемы, то посмотрим так ли это?

Положим, мы хотим, нажав одну кнопку включить некоторое устройство, а, нажав другую, выключить его. Как выглядит схема с применением реле:

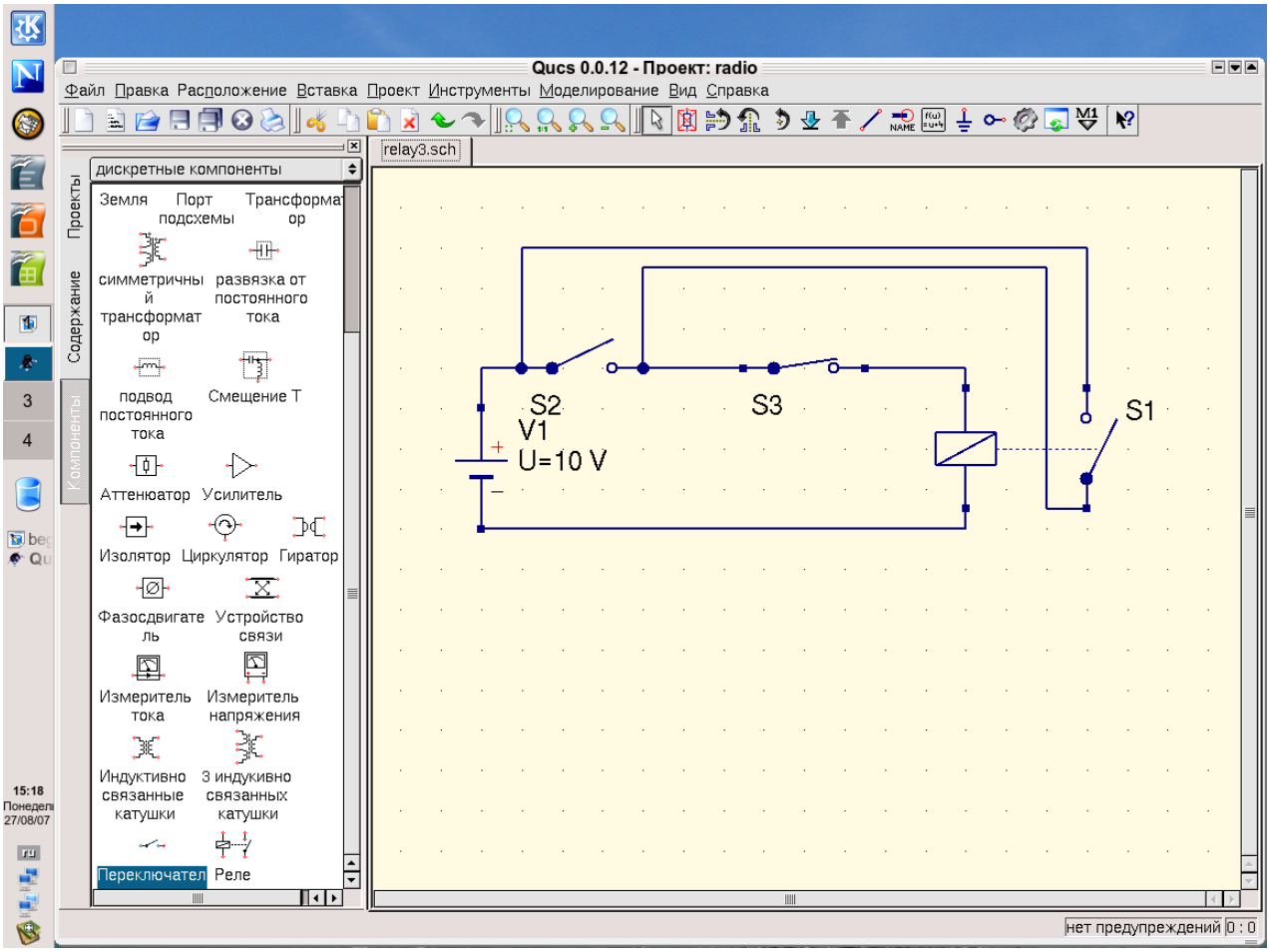


Рис. 4.3. Коммутатор с применением реле

Здесь контакты S2 принадлежат кнопке включения. Когда мы нажимаем эту кнопку, реле подключается к батарейке V1, включается и своими контактами S1 замыкает цепь питания реле. Реле остается включенным, пока мы не нажмем кнопку выключения S3, разрывающую цепь питания реле и выключающую реле. Попробуйте, используя любые другие элементы: транзисторы, резисторы, микросхемы, – получить такой же эффект, учитывая, что реле может иметь несколько пар контактов (на схеме изображена только одна), и оставшиеся могут коммутировать остальную часть схемы, и потребляющую достаточно большую мощность. Думаю, другие решения не будут столь лаконичны.

Эксперименты по замене, скажем, релейной схемы на транзисторную полезны тем, что есть все исходные данные. Мы знаем заранее, что мы должны получить. А то, как мы это сделаем, зависит от наших предпочтений, от наших знаний, и их, наши знания, пополняет.

Электромагнитные явления используются не только в измерительных приборах, микрофонах, громкоговорителях и реле: есть магнитные усилители, преобразователи, стабилизаторы и генераторы, наконец, не будем забывать о трансформаторах! Достаточно давно появились такие компоненты, как магнитодиоды и магнитотранзисторы. И еще не исчезли из обращения магнитофоны, аудио и видео. И есть целый класс устройств, которые называются электродвигатели.

Электродвигатели постоянного тока похожи по принципу работы на измерительный прибор – рамка с обмоткой (катушкой провода) поворачивается в магнитном поле под действием протекающего тока. Если рамка не одна, а несколько на металлическом

сердечнике, и если мы снимаем ток с той, что уже повернулась и подаем на ту, что еще только вошла в поле магнитов, то, продолжая этот процесс, мы заставим наше «многорамочное» устройство вращаться. А, вспоминая работу громкоговорителя и микрофона (взаимообратное преобразование), если мы начнем крутить наш двигатель постоянного тока, то можем ожидать, что на его выводах появится ЭДС, то есть, мы используем двигатель, как генератор. Кроме двигателей постоянного тока есть электродвигатели переменного тока. Есть и генераторы переменного тока...

Я еще раз просмотрел эту главу. Она получилась совсем короткой. Даже обидно. А взглянув на книжный шкаф, даже в его видимой части я обнаружил с десяток толстых книг, от «Электродинамики» до «Электрических машин». Можно было бы выписать много интересного, пополнив, благо текстовый процессор позволяет, рассказ красивыми формулами с поверхностными интегралами и ссылками на оригинальные труды Максвелла и Фарадея, что придало бы мне вес в собственных глазах, но... оставлю все как есть. Мне нравится математика своей строгостью и хотя бы тем, что она просто захватывающе интересна. Мне нравится физика своей непредсказуемой силой. Однако вернемся лучше к сигналам.

Глава 5. Сигналы или переменный ток на практике

Каждый раз, когда я вспоминаю о сигналах, я вспоминаю книгу «Радио-технические цепи и сигналы» Гоноровского И.С. Недавно мне вновь напомнили о ней. Конечно, она сложна для начинающих, но пройдет время, вы станете опытным любителем или профессионалом, вспомните, что есть такая хорошая книга, и обязательно почитайте ее.

Разные законы изменения переменного тока

Самый простой закон изменения тока – синусоидальный. Я уже говорил, что синусоидальное напряжение генерируют многие приборы для исследования электрических схем. Почему именно синусоидальное напряжение? Скорее всего, потому что сигналы другой формы можно представить как смесь сигналов синусоидальной формы. Поэтому, при испытаниях, например, усилителей, подавая от генератора сигнал синусоидальной формы, мы можем оценить воздействие собранного нами усилителя на испытательный сигнал. Если сигнал не изменил своего вида, то усилитель работает в хорошем режиме и не вносит искажений. Иногда мы намеренно вносим искажения, но это особый случай, чаще любые искажения фактор нежелательный. Как могут выглядеть искажения синусоидального сигнала на практике? Соберем простую схему, скажем, на двух транзисторах.

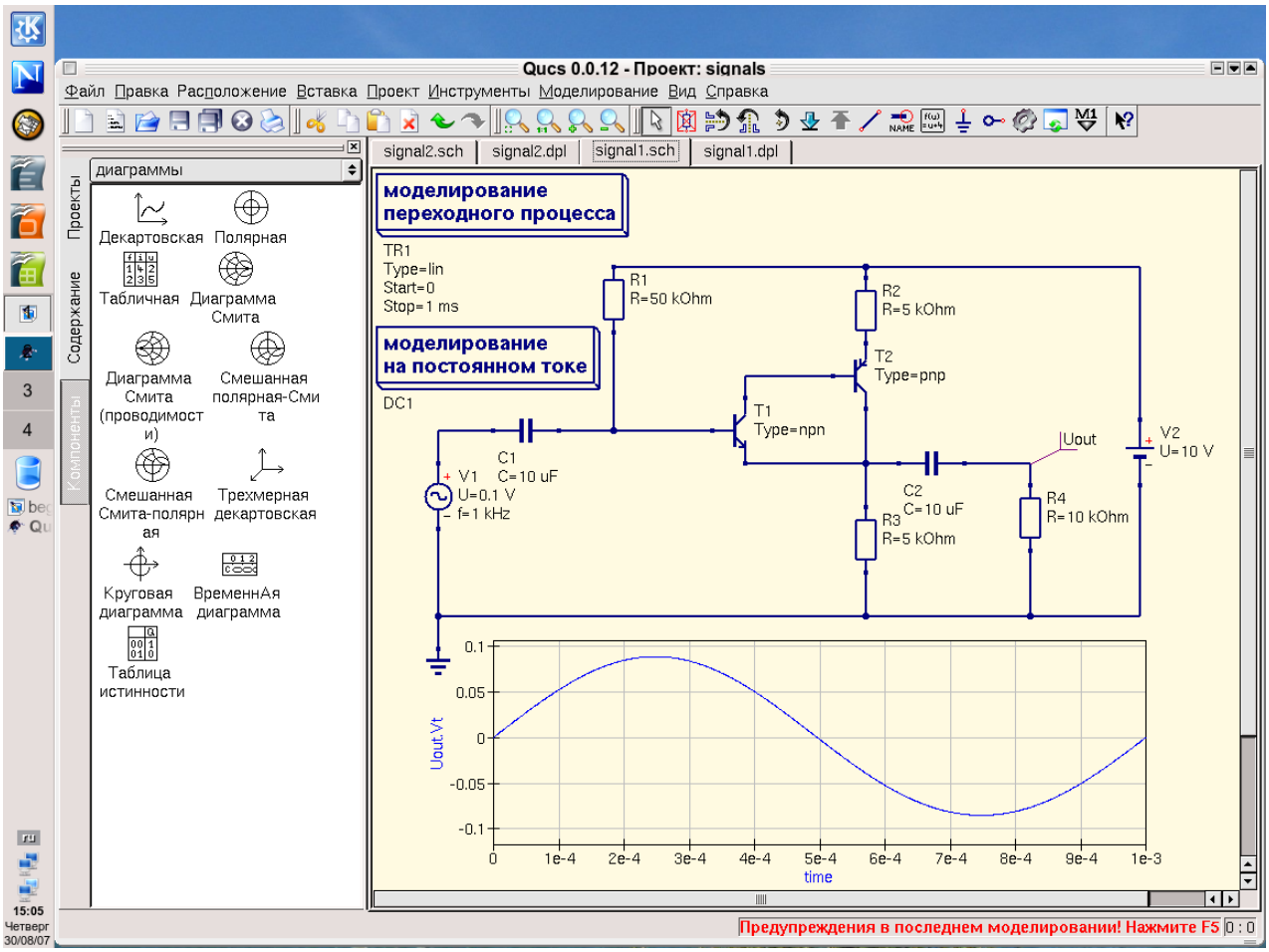


Рис. 5.1. Схема усилителя на двух транзисторах

Сигнал от генератора V1 подается на усилитель, резистор R4 служит нагрузкой, с которой

снимается сигнал Uout, изображенный на диаграмме. Сигнал синусоидальной формы. В данном случае, это еще следовало бы проверить, но пока примем все на веру, в данном случае искажений сигнала нет. Но что произойдет, если изменить параметры схемы.

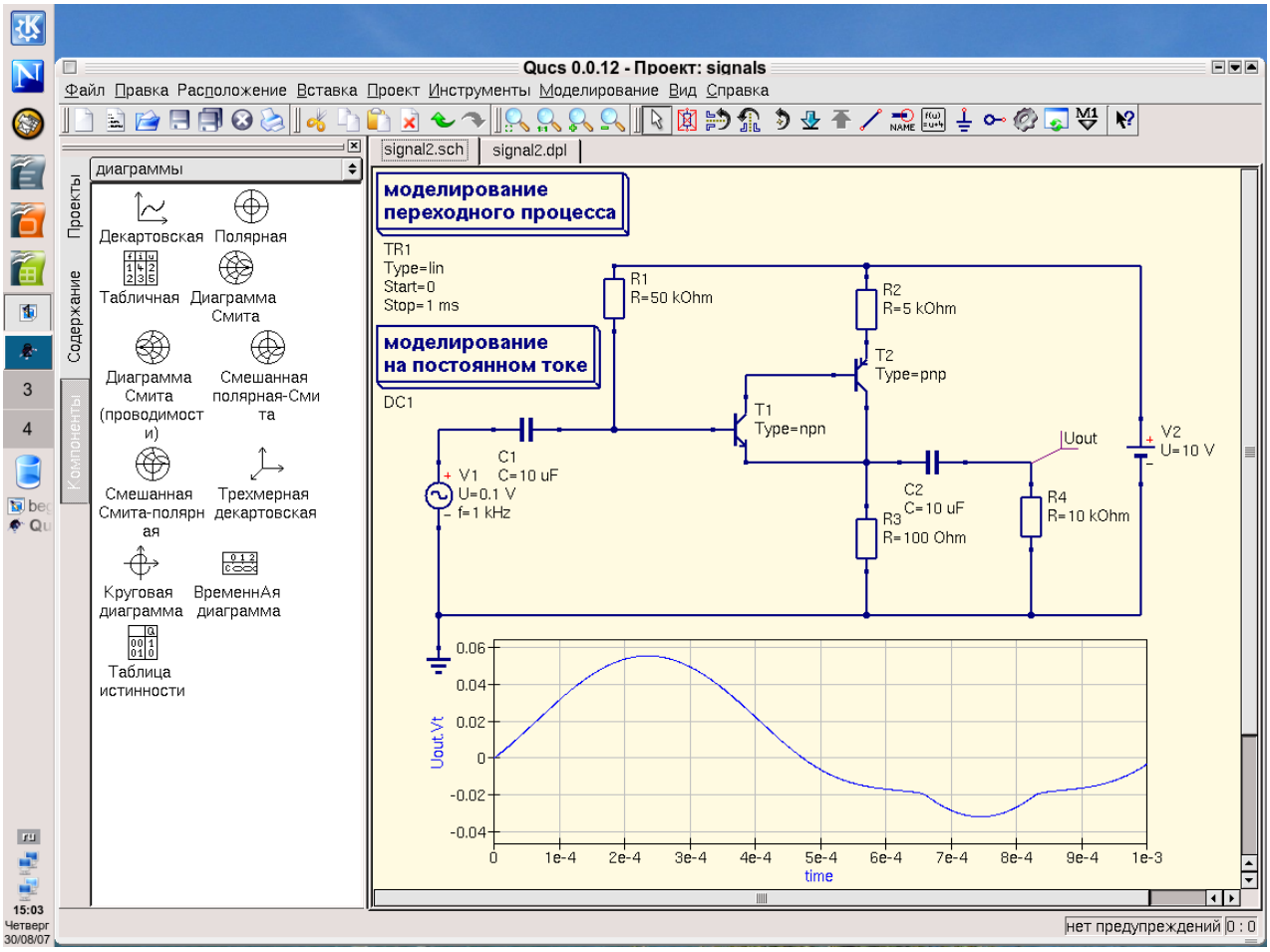


Рис. 5.2. Искажения сигнала, вносимые схемой усилителя

Даже я вижу, как искажается сигнал после того, как сопротивление R3 изменило свое значение с 5 кОм на 100 Ом. Если бы мы использовали осциллограмму реального звукового сигнала, имеющего весьма сложную форму, то эти искажения не были бы столь заметны, хотя на слух проявились бы в виде хрипов или каких-то других призвуков.

Слух – очень чувствительный инструмент проверки. Он подмечает все наши недоделки в усилителях звуковой частоты, но не слишком помогает определиться с тем, где мы ошибаемся, не может подсказать, что следует изменить в схеме. А приборы, генератор низкой частоты и осциллограф, при умении ими пользоваться могут не только показать наличие проблем, но и подсказать, как этих проблем избежать.

Я расскажу, как бы я стал рассуждать, увидев такое на экране осциллографа, а потом покажу, как бы я проверил свои рассуждения. Во-первых, исходный сигнал симметричная синусоида. Полученный мною на выходе сигнал имеет верхнюю полу-волну похожую на исходную, а нижнюю явно искаженную. Для получения симметричного сигнала на выходе желательно подобрать такой режим, когда постоянное напряжение на выходе в отсутствии сигнала равно половине питающего напряжения. В этом случае выходному сигналу «есть куда меняться», а он не может в своем размахе превысить возможности питающего напряжения. Если напряжение на выходе усилителя в отсутствии сигнала не равно половине

питающего напряжения, то одна полу-волна не будет иметь равные с другой возможности. Что, похоже, и произошло.

Чтобы проверить это, измерим постоянное напряжение на резисторе R3 относительно земли (общего провода), удалив генератор V1.

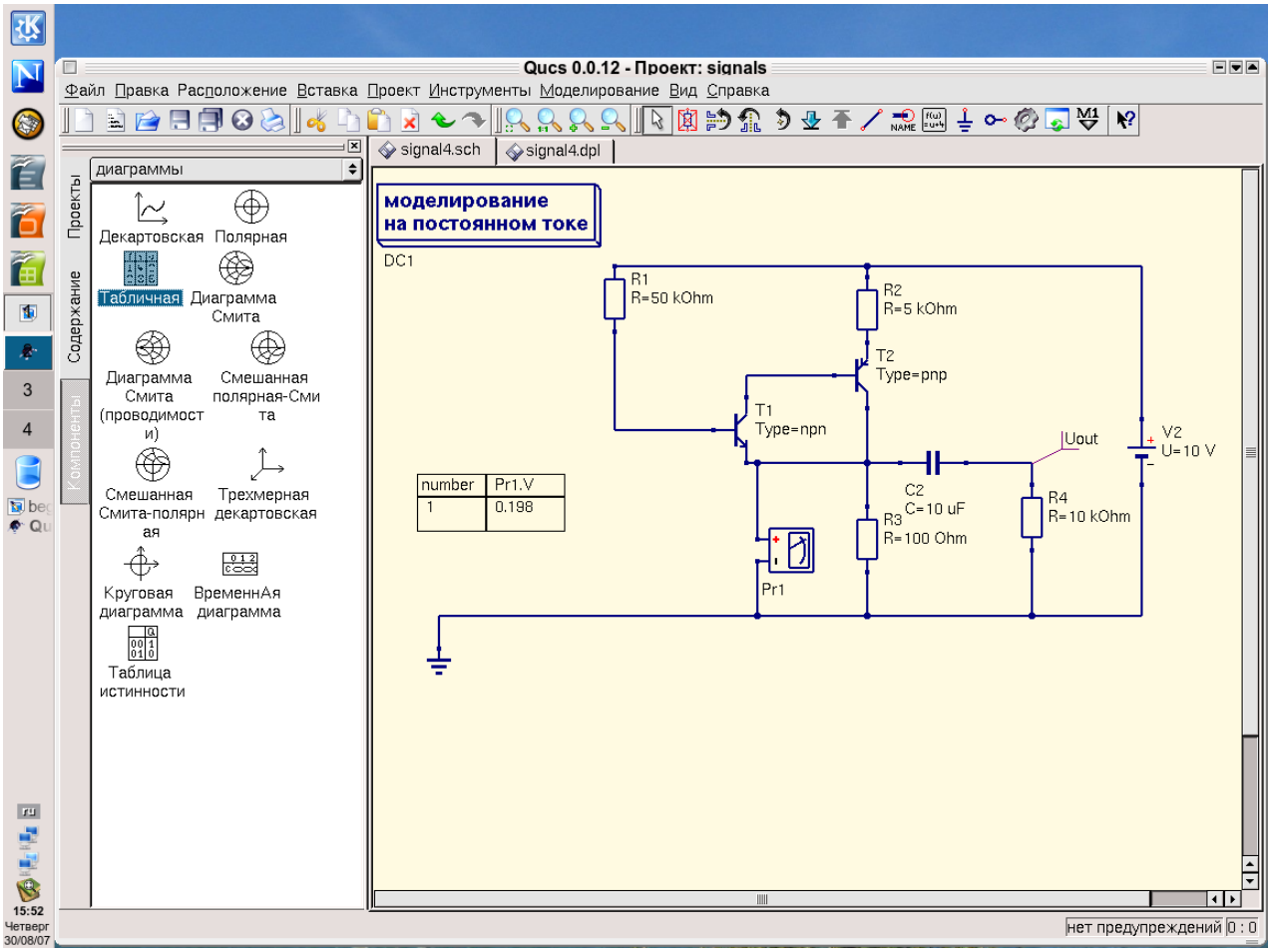


Рис. 5.3. Измерение постоянного напряжения в отсутствии сигнала

Как этого следовало ожидать, напряжение на резисторе R3 очень маленькое. А сигнал, усиленное переменное напряжение, на сопротивление нагрузки R4 должен сниматься именно отсюда. Следовательно, амплитуда сигнала будет ограничиваться этим маленьким значением напряжения. Напомню, переменное напряжение на выходе усилителя будет меняться только по величине, но не по направлению относительно общего провода. Меняться же по величине оно может в одну сторону существенно, а в другую, очень незначительно. Если нет особых соображений, нет особого использования схемы усилителя, резонно так подобрать параметры схемы (величины сопротивлений), чтобы напряжение в этой точке было близко к половине питающего напряжения. Особые соображения, которые могли бы помешать этому, могут касаться сигнала. Если предстоит усилить, скажем положительный импульс, то нет нужды смещать рабочую точку, или если входной сигнал заведомо мал, тоже можно не очень беспокоиться, хотя 0.2 В — это слишком мало.

Чтобы улучшить положение дел, следует либо изменить параметры схемы, увеличив это напряжение, либо уменьшить значение входного сигнала. Проще сделать второе.

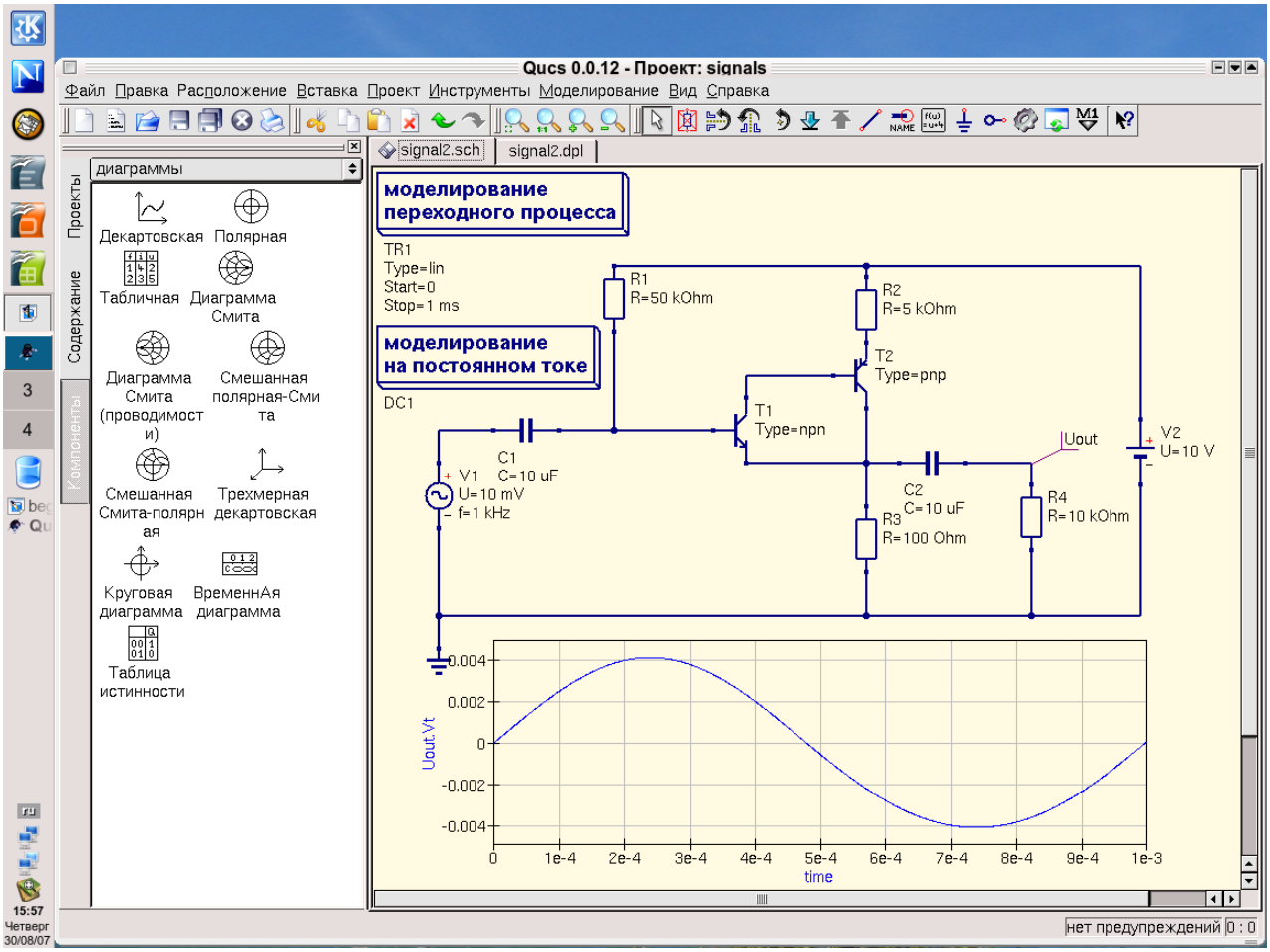


Рис. 5.4. Сигнал на выходе усилителя после уменьшения входного сигнала

Этот вид выходного сигнала после уменьшения входного сигнала до 10 мВ больше напоминает синусоиду.

Искажения синусоидального сигнала, видимые на рисунке 5.2, называют нелинейными искажениями. Иными словами, переменное напряжение, поданное на вход усилителя, на его выходе изменяется несколько по другому закону из-за того, что преобразование сигнала не было линейным. Математический аппарат на основе работ Фурье позволяет сказать, что и насколько изменилось. На практике для этой цели применяют анализатор спектра выходного сигнала. А при работе с усилителями звуковой частоты чаще используют другой прибор, показывающий общий вклад составляющих искажений, который называется измерителем нелинейных искажений.

Искаженный на выходе сигнал можно представить как совокупность нескольких сигналов: основной синусоидальный сигнал, как тот, что был на входе, и несколько синусоидальных сигналов с меньшей амплитудой и частотой в 2, 3, 4 и т.д. раз больше, возможно с разными фазами. Их называют гармониками или гармоническими составляющими.

Проделаем такой эксперимент. Возьмем несколько источников переменного напряжения с кратными частотами, разными амплитудами и смешаем их вместе. Посмотрим, что происходит с основным сигналом в таком коктейле (смешаем, но не будем взбалтывать).

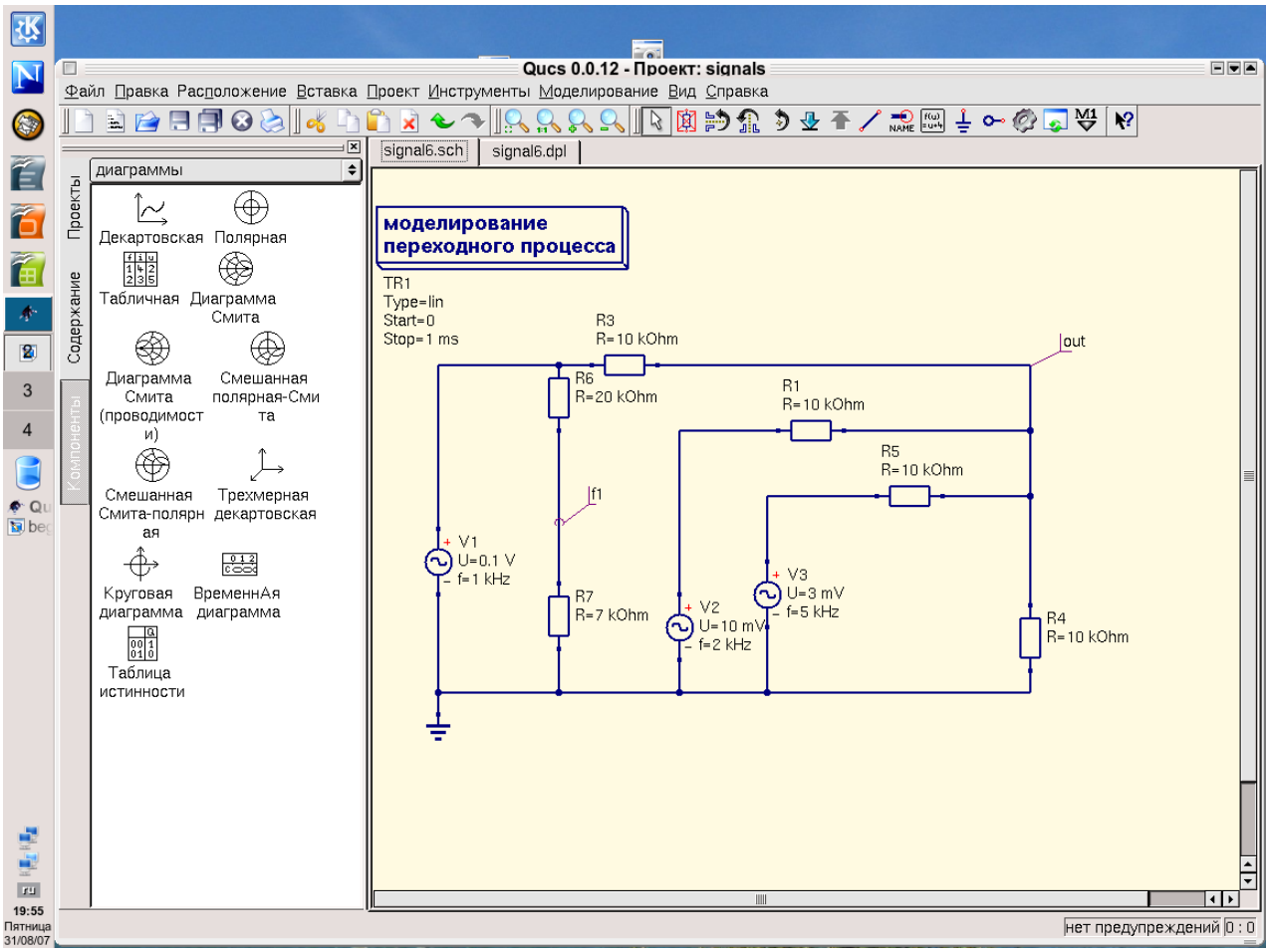


Рис. 5.5. Смешивание синусоидальных сигналов кратных частот

На диаграмме ниже показан результирующий сигнал и основной с частотой 1 кГц.

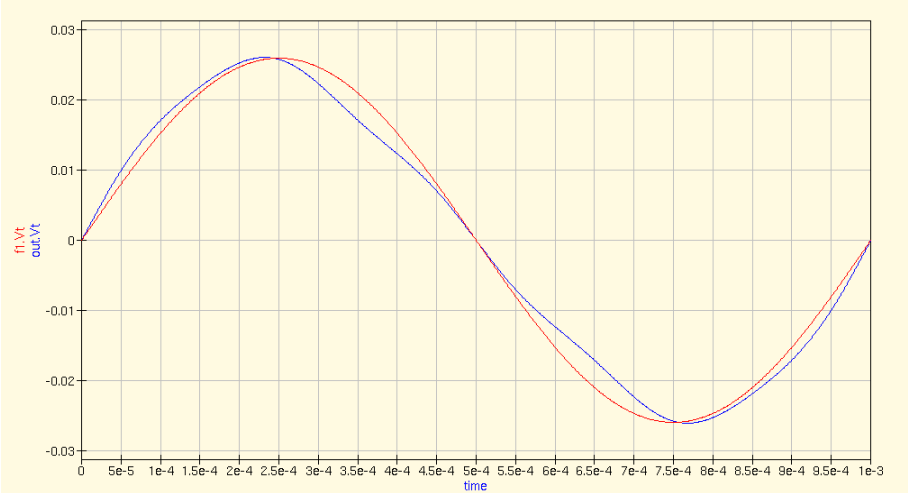


Рис. 5.6. Результат проведенного эксперимента

Появление сигналов с частотой 2 кГц и 5 кГц привело к изменению вида сигнала. Многие периодические сигналы, например, прямоугольные или треугольные, можно представить подобной смесью синусоидальных сигналов. На практике, за редким исключением, нас интересует, как избежать появления искажений. Есть графические приемы, поясняющие

причины возникновения нелинейных искажений в усилителях, есть математические методы, но можно пользоваться достаточно простым объяснением. Исходный сигнал, полученный с помощью генератора синусоидального сигнала, не бывает идеальным. В нем всегда есть примесь гармоник, сигналов кратных частот. С другой стороны, возможности усилителя выполнить свою работу, усилить входной сигнал, не безграничны. Они ограничены величиной питающего напряжения. Если увеличивать сигнал на входе усилителя, то в какой-то момент сигнал на выходе достигает наибольшего возможного значения. Для определенности скажем, что двойная амплитуда сигнала стала равна питающему напряжению. При дальнейшем увеличении входного сигнала основная частота на выходе увеличиваться не будет. Но гармонические составляющие входного сигнала имеют значительно меньшую амплитуду, они могут усиливаться, и они продолжают усиливаться усилителем. Получается так, что с каждым увеличением сигнала на входе, мы получаем новую смесь основной частоты и гармоник на выходе, где амплитуда гармоник увеличивается, тогда как основная частота остается на прежнем уровне. Попробуйте менять величину напряжения источников V2 и V3 на схеме рисунка 5.5. Посмотрите, как будет меняться вид сигнала. На макетной плате этот эксперимент проделать труднее, потребуется несколько генераторов, но результат должен получится похожим.

Каким образом оценивается величина искажений? Я уже говорил о таком приборе, как измеритель нелинейных искажений. Его принцип работы достаточно прост. На входе прибора стоит фильтр, который подавляет основную частоту. То, что остается, попадает на вольтметр. По показаниям вольтметра оценивают величину нелинейных искажений. Отношение измеренного напряжения после подавления основной частоты к напряжению исходного сигнала, выраженное в процентах, в первом приближении и будет оценкой искажений – коэффициентом нелинейных искажений. Для усилителя звуковой частоты коэффициент нелинейных искажений очень важный параметр. Хотя человеческое ухо не слишком хорошо слышит искажения порядка 1-2%, коэффициент нелинейных искажений качественного усилителя мощности желательно иметь не более 0.1-0.05%. Очень большие искажения звука при усилении вносят громкоговорители 1-2%, но если усилитель будет иметь искажения 1-2%, они добавятся к искажениям громкоговорителей и результат будет хорошо заметен на слух. Для борьбы с нелинейными искажениями при создании усилителей стараются тщательно установить режимы работы всех каскадов по постоянному току, применяют элементную базу, линеаризующую работу усилителя, вводят отрицательную обратную связь. Эти методы мы рассмотрим позже, а сейчас проведем несколько экспериментов, позволяющих лучше понять работу измерителя нелинейных искажений. Возьмем колебательный контур (параллельно включенные индуктивность и емкость), настроенный на частоту 1 кГц. Я говорил, что при таком включении два реактивных элемента имеют особенную частоту, на которой их сопротивления равны. Эту частоту называют резонансной. В параллельном колебательном контуре на частоте резонанса сопротивление контура максимально. Если последовательно с контуром включить резистор, мы получим делитель напряжения, но делитель частотно-зависимый. На одних частотах напряжение, снимаемое с резистора, будет одним, на других, другим. Проверим это с помощью программы или на макетной плате.

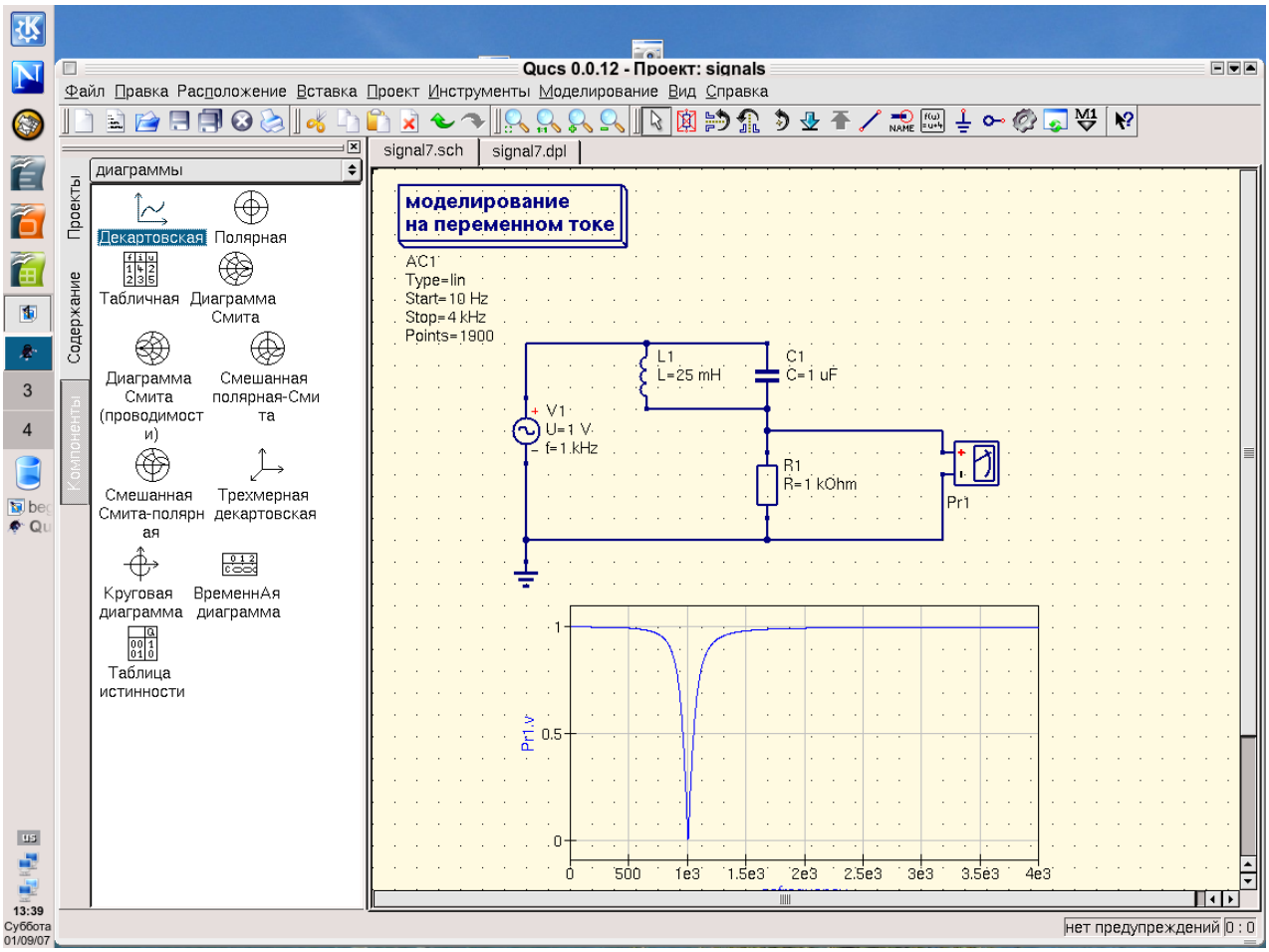


Рис. 5.7. Параллельный колебательный контур

Как видно из рисунка, напряжение после фильтра на частоте близкой к 1 кГц минимально. Эта частота «вырезается» из спектра сигнала, или «подавляется» фильтром. Используем полученную схему для того, чтобы посмотреть результат подавление частоты 1 кГц на выходе схемы на рисунке 5.2. В первом эксперименте настроим генератор V1 на частоту 100 Гц и осциллографом посмотрим сигналы на входе фильтра (*out1*) и на его выходе (*out2*). Следуя нашим рассуждениям фильтр не должен мешать прохождению сигнала, и вид сигнала на выходе фильтра не должен отличаться от вида на входе. В настройках моделирования при частоте сигнала 100 Гц время моделирования приходится увеличить с 1 мС до 30 мС, в противном случае видна только часть сигнала. Фильтр на выходе усилителя включен вместо нагрузочного резистора R4, в остальном обе схемы, усилителя и фильтра, не подверглись изменениям. Посмотрим, что у нас получится.

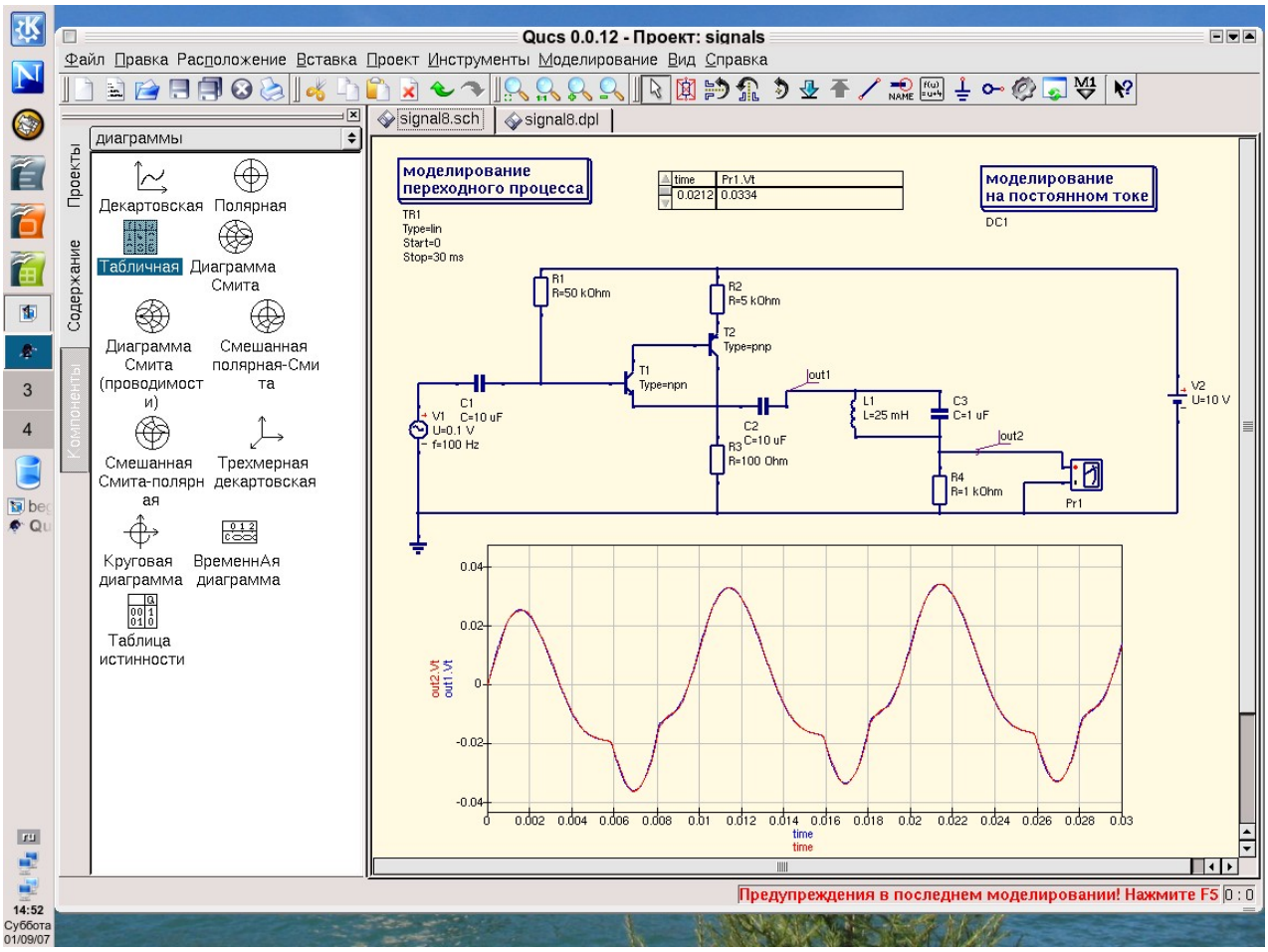


Рис. 5.8. Вид сигналов на входе и выходе фильтра на частоте 100 Гц

Как мы и предполагали, оба сигнала слились в один. В верхней части рисунка есть таблица, показывающая значения напряжений измерителя на выходе фильтра. При выборе показаний, зависящих от времени, в таблицу записываются все показания на каждом шаге моделирования. Я выделил максимальное значение 0.0334 В. Аналогично поступим и во втором эксперименте.

Для этого изменим частоту задающего генератора V1 со 100 Гц на 1 кГц. Время моделирования несколько уменьшим, до 10 мС. Посмотрим, что изменится.

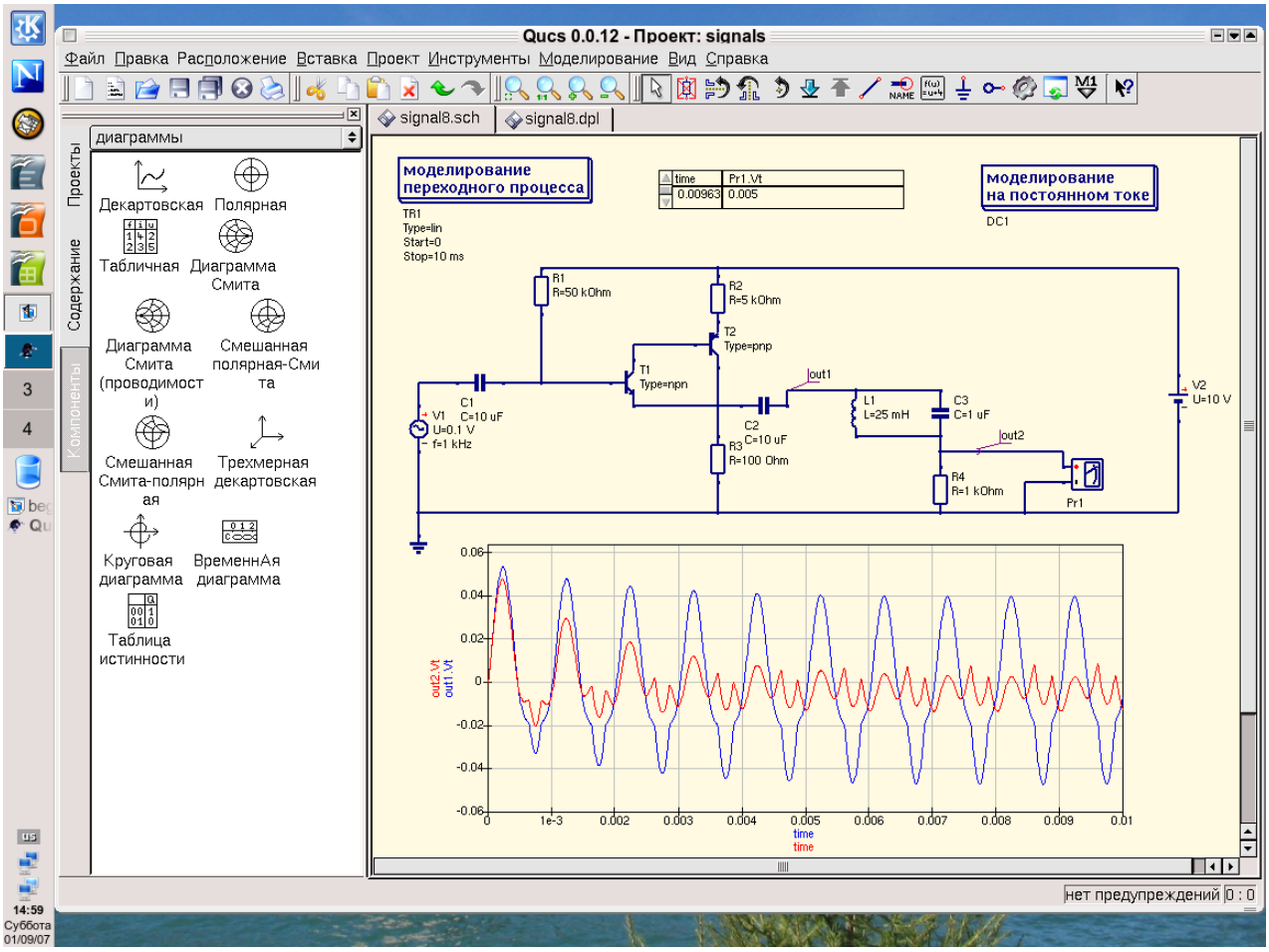


Рис. 5.9. Вид сигналов на входе и выходе фильтра на частоте 1 кГц

На выходе фильтра остались гармонические составляющие, а измеритель напряжения показывает значение порядка 0.005 В. Полученные измерителем значения напряжения в двух экспериментах можно использовать для оценки коэффициента нелинейных искажений. Разделим второе показание на первое и умножим на 100. Получим величину порядка 15 %. Нелинейные искажения такой величины должны быть заметны на слух.

Хотя промышленные приборы для измерения коэффициента нелинейных искажений строятся несколько иначе, для практических целей можно использовать именно такой подход. При этом достаточно иметь один-три фильтра, скажем на частоту 100 Гц, 1 кГц и 10 кГц. Фильтры с LC контуром получаются громоздкими, но можно сделать активные RC фильтры. Они гораздо компактнее. Если вы не ставите своей целью создание усилителя низкой частоты, предварительного или усилителя мощности, очень высокого качества, то оценки нелинейных искажений при разных настройках усилителя может оказаться достаточно. Если же вы собираетесь строить высококачественные усилители, вам понадобится качественный прибор для измерения нелинейных искажений. Есть схемы любительские, есть промышленные приборы. При создании своих схем прибора следует учесть несколько немаловажных факторов – генератор испытательных сигналов сам должен иметь очень маленький коэффициент нелинейных искажений. Построение широкополосного генератора даже низкой частоты с малыми нелинейными искажениями задача не столь простая, как может показаться на первый взгляд. Чтобы ее упростить, можно также строить несколько генераторов на несколько частот. Точная настройка фильтра на заданную частоту потребует подстройки всех элементов фильтра, их следует брать такими, которые легко позволяют

менять значение компонента, скажем, параллельным включением подстроечного. И, наконец, измеряя напряжение на входе и выходе фильтра с помощью вольтметра переменного напряжения, не следует забывать, что большая часть таких вольтметров показывают переменное напряжение с обусловленной точностью только тогда, когда переменное напряжение синусоидальное, чего никак не скажешь о форме выходного сигнала после фильтра.

Не следует переоценивать влияния нелинейных искажений. Во многих случаях суммарные искажения всего тракта усиления звука 1-1.5% вы можете не заметить на слух. Для карманного радиоприемника они будут вполне приемлемы, а для усилителя лучше использовать готовую микросхему УНЧ, чем делать схему на транзисторах. С другой стороны, научиться строить усилители и разбираться в их работе трудно в опытах с готовой микросхемой.

К слову, осциллограф-приставка к компьютеру, о чем говорилось выше, может иметь встроенную функцию анализатора спектра. Если обратиться к рисунку 3.6, видно, что программа переключается для работы в разных режимах, не только в режиме осциллографа. Наличие анализатора спектра позволит достаточно качественно оценивать нелинейные искажения.

Кроме нелинейных искажений в аналоговых схемах, хорошо выявляемых испытаниями с помощью синусоидальных сигналов, есть искажения, которые оказывают существенное влияние на работу цифровых схем. Как правило, сигналы цифровых микросхем – это прямоугольные импульсы, периодические или нет. Идеальный прямоугольный сигнал имеет мгновенно устанавливающейся уровни, высокий и низкий. Реальный сигнал для установления уровней требует небольшого, но времени. Это в первую очередь связано с наличием емкостей: монтажных, конструктивных и штатных, относящихся к самим элементам схемы. Чем выше быстродействие микросхемы, и чем ниже используемая частота переключений, тем больше сигнал на выходе микросхемы похож на идеальный. Собственно, если учитывать, а при проектировании это всегда принимают во внимание, если учитывать отличие сигнала от идеального, то работа схемы получается правильной. Для приведения сигналов к «правильным» временным рамкам используют, например, цепи задержки импульсов. При наладке, если даже продуманная задержка не помогла, используют самый простой способ — добавляют конденсатор на выход микросхемы, сигнал которой следует «притормозить». Такая необходимость возникает, если вы записываете данные в цифровую микросхему. Сигнал записи должен приходить только в тот момент, когда данные установлены на входе, иначе они могут «прочитаться» неверно. Если же данные, которые предстоит записать, собираются по всей схеме, то может получится так, что они запаздывают. Тогда можно попытаться задержать сигнал записи с помощью конденсатора.

Как это выглядит, посмотрим в самом простом случае. Проведем эксперимент с RC цепочкой.

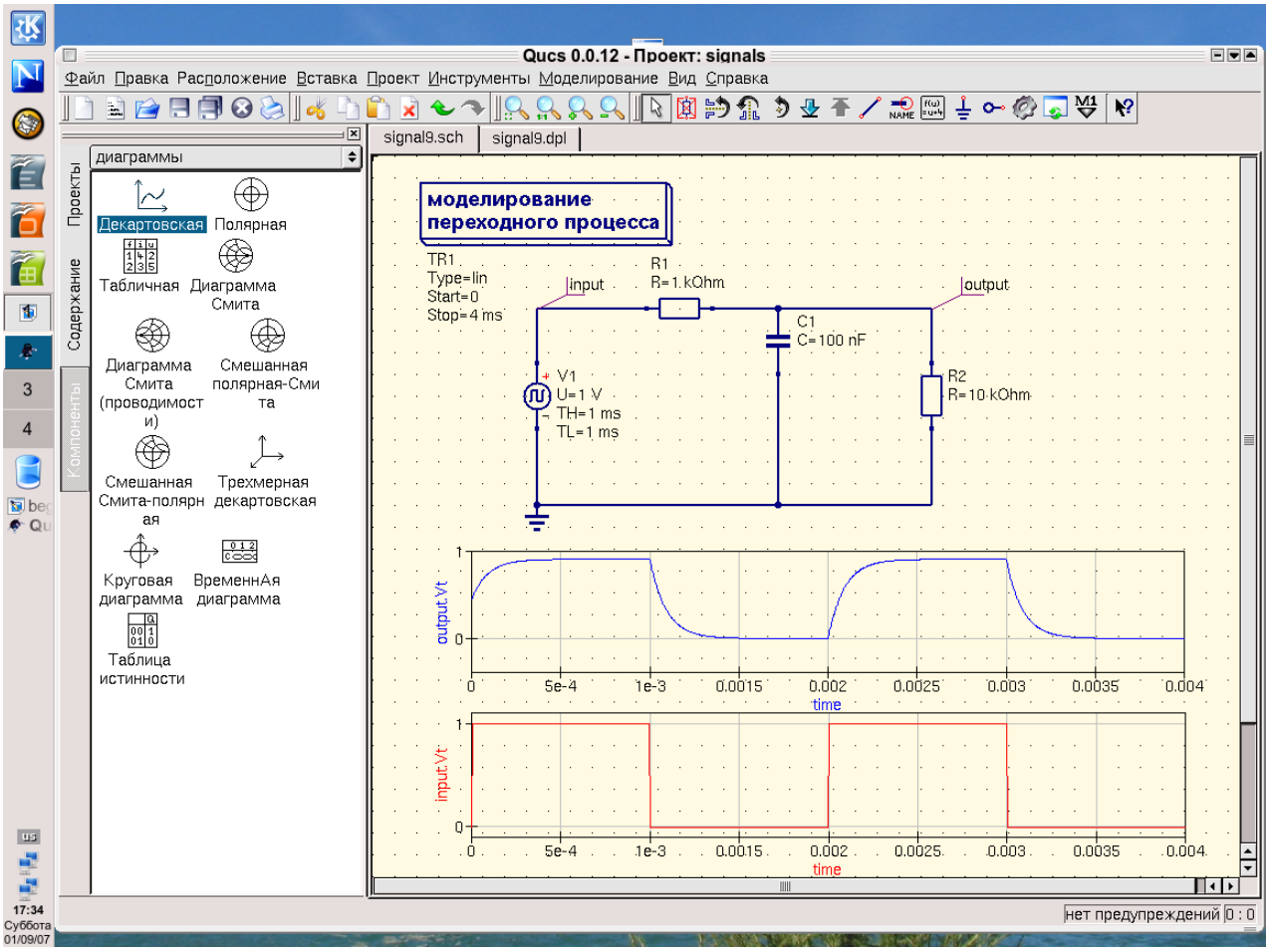


Рис. 5.10. Влияние RC цепи на формирование фронтов прямоугольного импульса

Нижняя осциллограмма соответствует сигналу генератора, а верхняя – сигналу, прошедшему через RC цепь. Конденсатор C1, напомню, в начальный момент времени полностью пропускает через себя ток, при этом напряжение на нем близко к нулевому. Затем конденсатор заряжается, а напряжение на нем растет по мере заряда. Время, которое требуется конденсатору для того, чтобы он зарядился до напряжения источника, прямо пропорционально его емкости и величине сопротивления, через которое конденсатор заряжается.

Похожий процесс происходит и тогда, когда напряжение источника становится нулевым. В этом случае конденсатор начинает разряжаться, на схеме через резистор R1 (внутреннее сопротивление источника напряжения в данном случае равно нулю) и резистор R2. Это тоже требует некоторого времени. Таким образом, конденсатор задерживает переход сигнала из состояния низкого уровня в высокий и наоборот.

Чтобы посмотреть, как на работе цифровой схемы могут сказаться подобные временные искажения, проведем исследование.

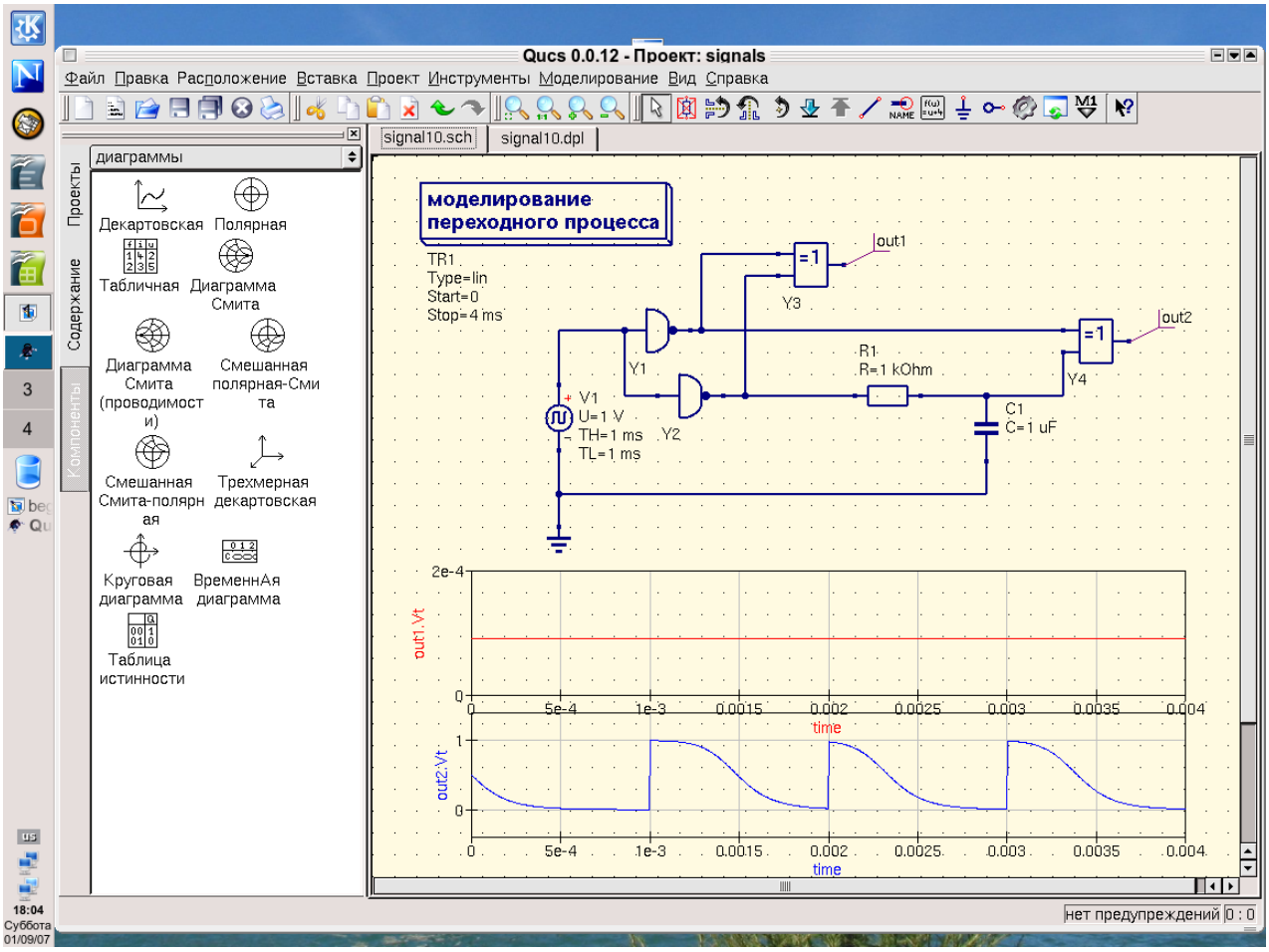


Рис. 5.11. Влияние емкостей на работу цифровых схем

Две одинаковые микросхемы Y3 и Y4 (исключающее ИЛИ) подключены к генератору прямоугольных импульсов через инверторы Y1 и Y2. Первая подключена непосредственно к инверторам, и на ее выходе сохраняется уровень логического нуля, а у второй микросхемы один из входов подключен через RC цепь. На выходе этой микросхемы появляются импульсы. Разница заметная.

Любая реальная цифровая микросхема имеет такой параметр, как время задержки импульса от входа к выходу. На рисунке 5.11 инверторы Y1 и Y2 имеют одинаковое время задержки распространения импульса. Но если изменить время задержки, а программа Qucs позволяет менять этот параметр, у одного инвертора, оставив второй без изменений, то вместо низкого уровня на выходе Y3 мы вновь получим импульсы. На рисунке ниже время задержки распространения сигнала у первого инвертора равно нулю, а у второго 100 мкС.

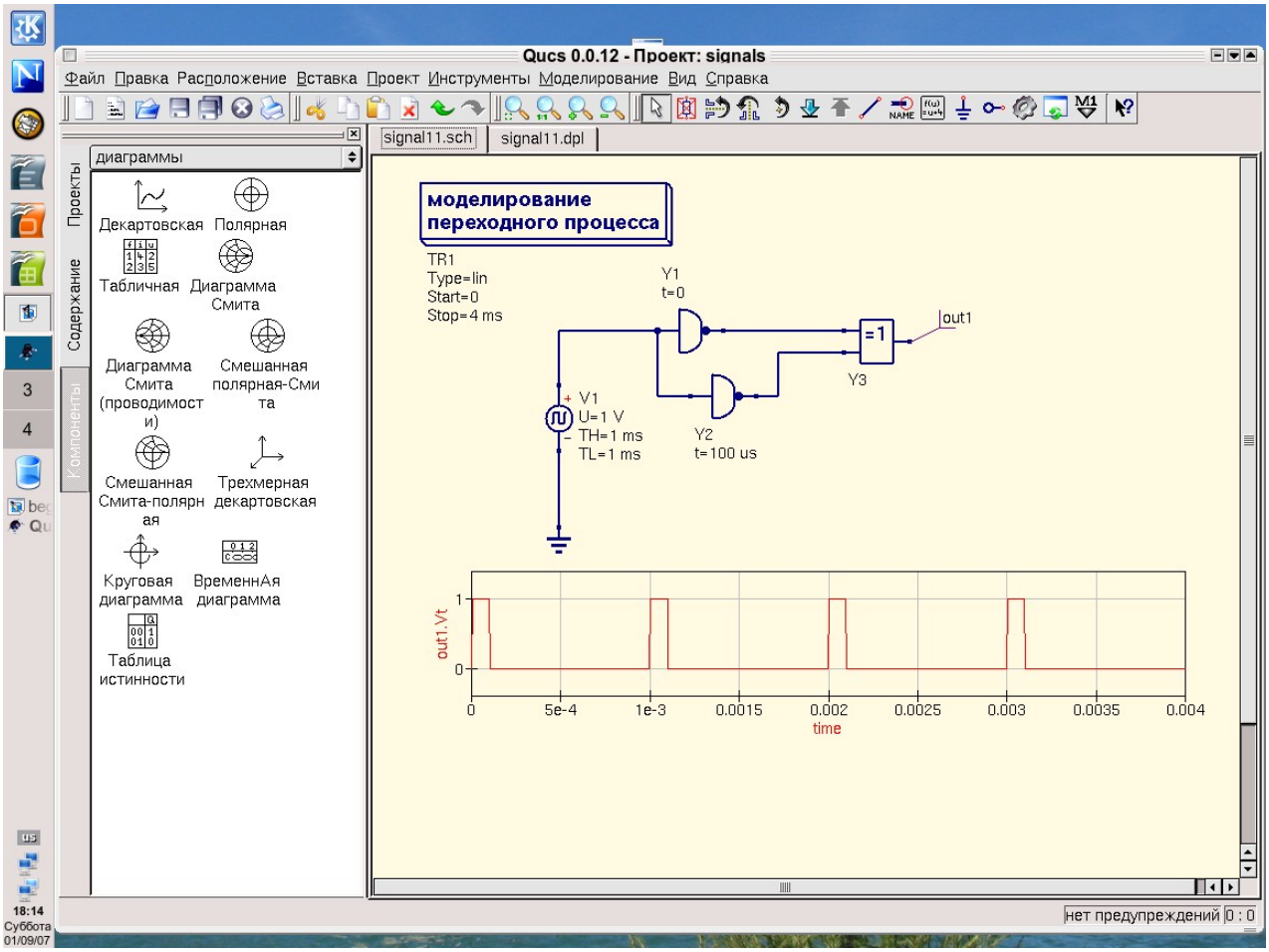


Рис. 5.12. Влияние времени задержки распространения на работу цифровой схемы

Подобные искажения, изменение времени становления фронтов и временные задержки, могут отрицательно сказаться на работе сложной и полезной схемы, их обязательно нужно будет отыскать и устранить. Но эти же искажения с успехом используются для создания схем формирователей импульсов, формирователей задержек и т.п. Важно знать, как работают эти искажения, чтобы из их вредного влияния на работу схемы извлечь пользу.

Затягивание фронтов импульсов из-за паразитных емкостей может сильно сказываться на правильности работы цифровых устройств. Например, из-за неодинакового времени задержки могут появляться короткие импульсы, которые в быстродействующих сериях цифровых микросхем могут вызывать срабатывание триггеров. Очень короткие импульсы, практически, не видны на экране осциллографа, и об их существовании приходится скорее догадываться, чем их наблюдать. Впрочем, и искажения синусоидального сигнала порядка 1-2% не будут заметны на экране осциллографа. Эти особенности искажений сигналов следует учитывать, иначе ошибка неизбежна.

Генерация сигналов

Одни радиоэлектронные устройства предназначены для обработки сигналов, которые они получают извне. Таков радиоприемник. Он принимает радиосигнал, сформированный за многие километры от него, и его задача преобразовать радиосигнал в звук. Но многие устройства в своем составе имеют узлы, где сигнал определенного вида должен быть сформирован «на месте». Узлы, создающие переменное напряжение, меняющееся по

заданному закону, обычно называют генераторами. Генераторы тактовых импульсов в цифровых устройствах, генератор несущей частоты в радиопередатчике, гетеродин радиоприемника, генератор испытательного сигнала в приборе – всех применений генераторов не перечесть в одной фразе.

Как работает генератор?

Самым наглядным образом это можно понять, как мне кажется, если вернуться к схеме операционного усилителя. Синусоидальный сигнал от источника переменного напряжения V1 подадим на инверсный вход операционного усилителя. На выход усилителя включим делитель напряжения из резисторов R2 и R3.

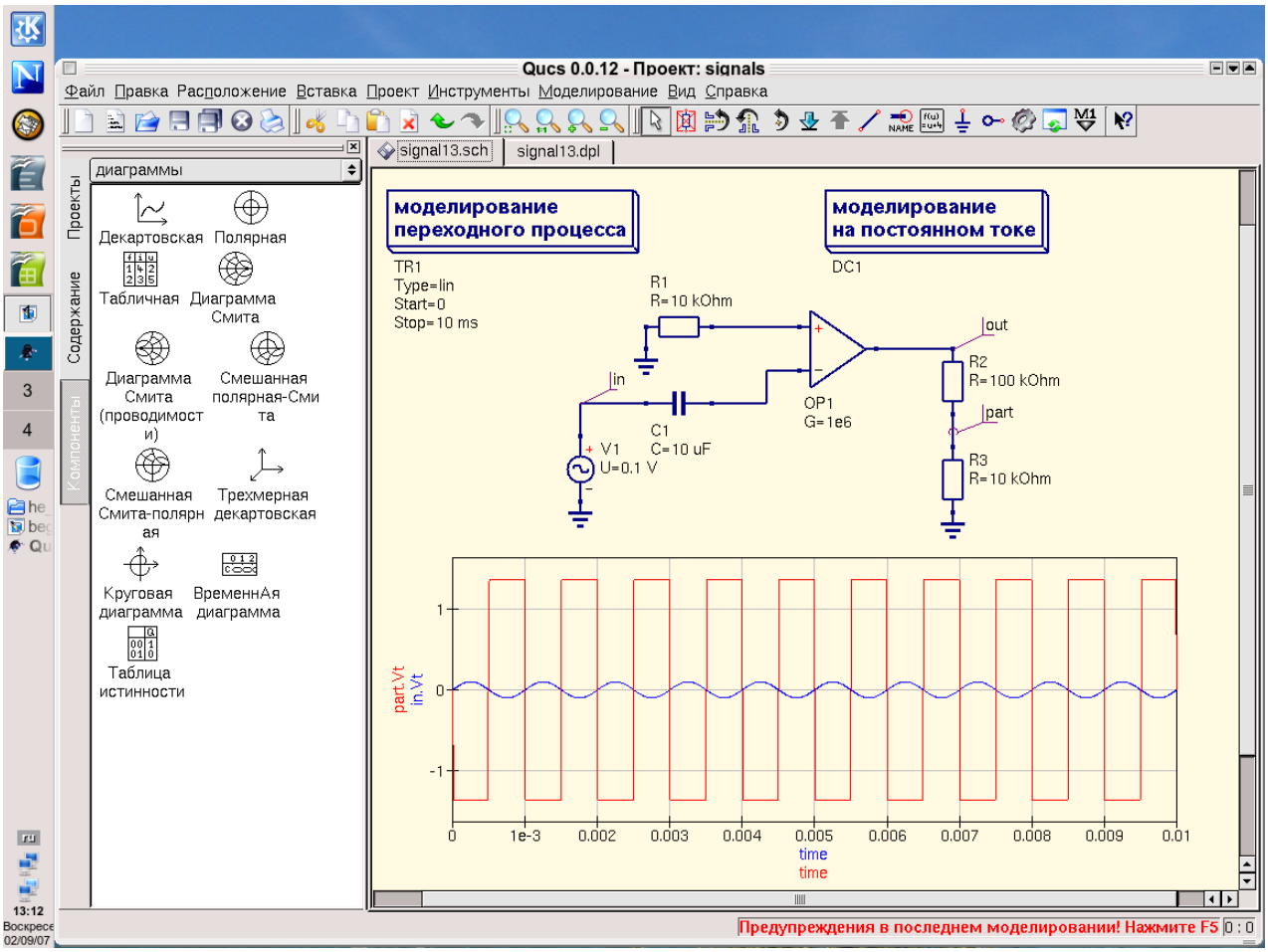


Рис. 5.13. Фазовое соотношение выходного и входного, на инверсном входе, сигналов

Хотя сигнал на выходе, явно с большими нелинейными искажениями, по форме не похож на входной синусоидальный, он, кроме того, находится в противофазе со входным. На осциллограмме показан не весь выходной сигнал, а его часть (*part*), снятая с делителя напряжения.

Снимем сигнал с инверсного входа и подадим его на прямой вход операционного усилителя, не меняя структуры схемы.

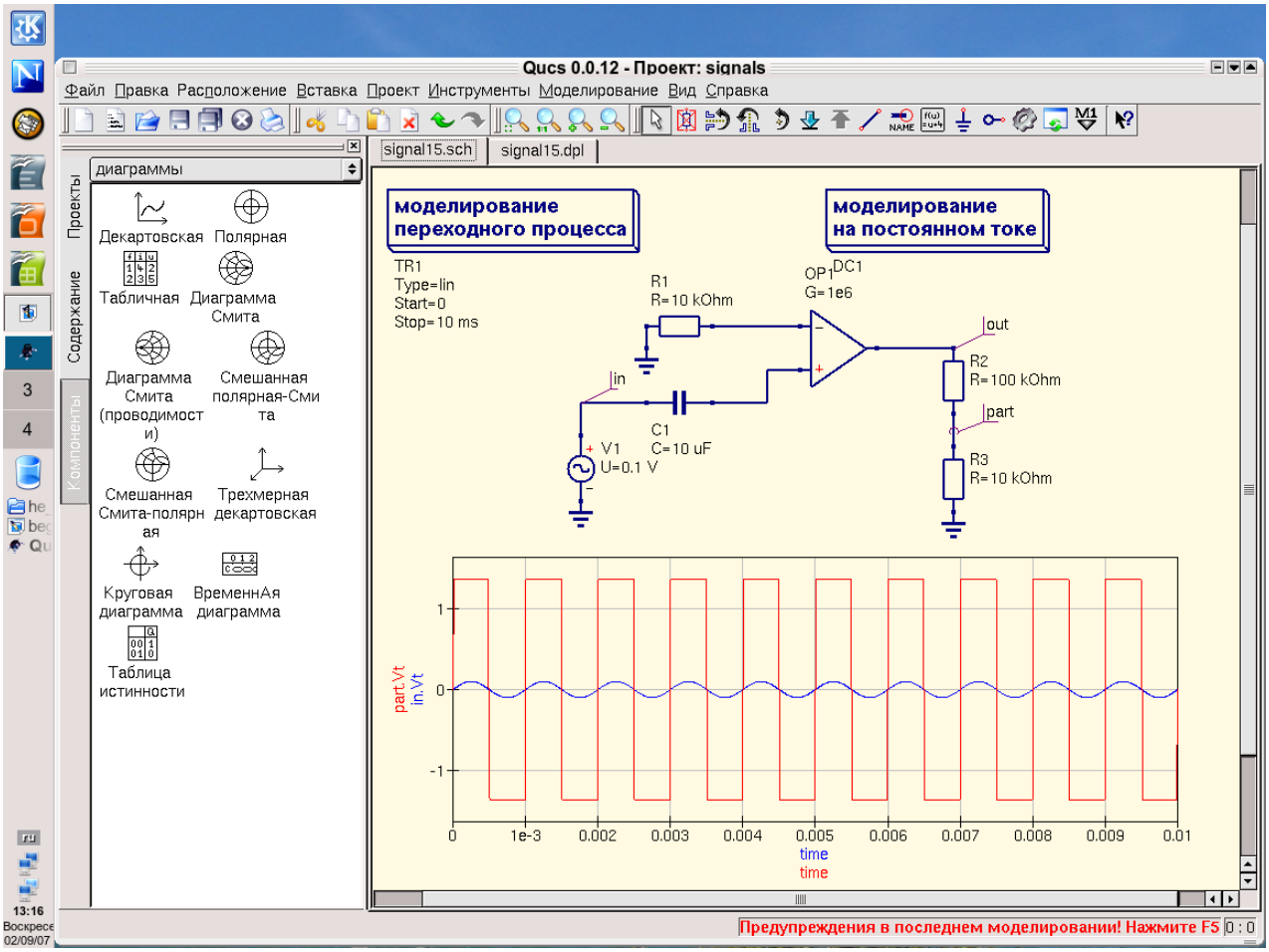


Рис. 5.14. Осциллограмма при подаче сигнала на прямой вход ОУ

Обе осциллограммы схожи, но сигналы стали синфазными. Если мы подадим сигнал с делителя напряжения на один из входов, мы получим обратную связь. Когда сигналы совпадают по фазе, как на рисунке 5.14, обратная связь называется положительной. При противофазных сигналах, как на рисунке 5.13, говорят об отрицательной обратной связи. Чтобы сразу оценить влияние отрицательной обратной связи проведем два эксперимента. В первом эксперименте мы получим частотную характеристику конкретного операционного усилителя μA741, точнее амплитудно-частотную характеристику. На практике такая характеристика получается измерением напряжения на выходе при подаче на вход усилителя от генератора сигналов последовательно растущей частоты с неизменным напряжением. Для каждой частоты удобно определять коэффициент усиления и отмечать его на графике. Очень часто для коэффициента усиления используют не отношение напряжений на выходе и на входе, а логарифм этого отношения, умноженный на 20. Такое выражение имеет специальное название, вернее специально названную единицу – децибел. Чем полезно использование децибел? Если коэффициенты усиления двух каскадов для получения результирующего усиления нужно перемножать, то выраженные в децибелах эти коэффициенты складываются. Проще складывать, чем перемножать.

Программа Qucs позволяет провести такое исследование проще. Есть моделирование на переменном токе. В свойствах этого вида моделирования мы изменим тип с линейного на логарифмический, зададим начальную частоту, положим, 1 Гц, а конечную 10 кГц, и увеличим количество точек до 1100. Кроме того на полученной осциллограмме выходного сигнала *out* мы на вкладке *Свойства* установим опцию *логарифмическая разметка оси X*.

Частотная характеристика операционного усилителя без обратной связи получается следующей.

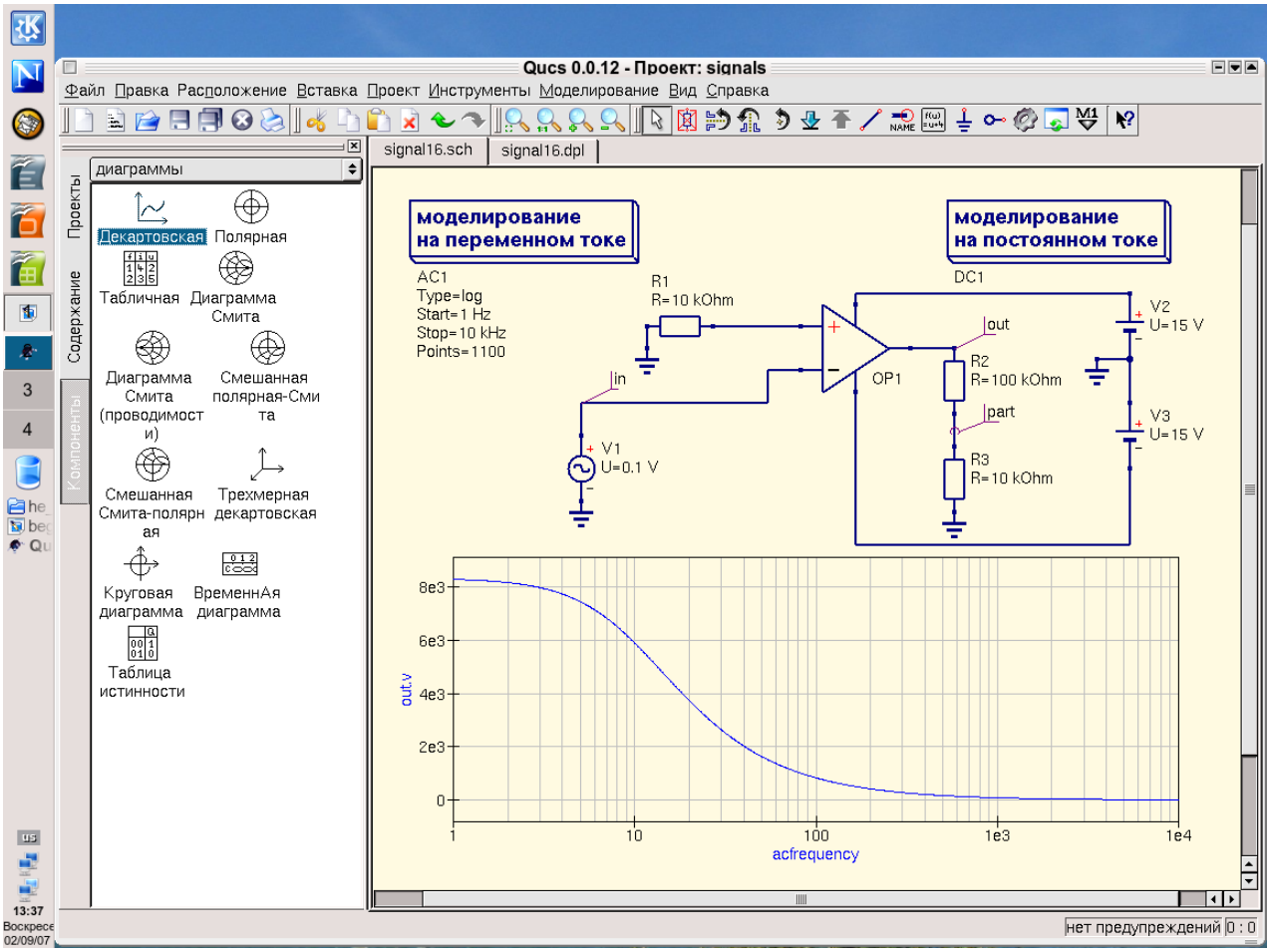


Рис. 5.15. Амплитудно-частотная характеристика ОУ без обратной связи

Начальный коэффициент усиления около 80000 (входное напряжение 0.1 В). С ростом частоты он уменьшается и на частоте 1 кГц почти равен нулю. Забегая вперед, хочу обратить внимание на тот факт, что коэффициент усиления на частоте 10 Гц равный 60000 на частоте в десять раз большей, 100 Гц, превращается в 6000 (или близко к этому). Спад частотной характеристики в десять раз (или на 20 дБ) на декаду (изменение частоты в десять раз), а не больше, оказывается очень полезен при введении отрицательной обратной связи. В операционных усилителях такой вид частотной характеристики получается благодаря специальному построению схемы.

Введем в нашу схему рисунка 5.15 отрицательную обратную связь. Для этого мы соединим точку, помеченную как *part*, делителя выходного напряжения с инверсным входом операционного усилителя. Схема в таком виде в точности соответствует типовой схеме включения операционного усилителя, как масштабирующего. После этого повторим моделирование на переменном токе.

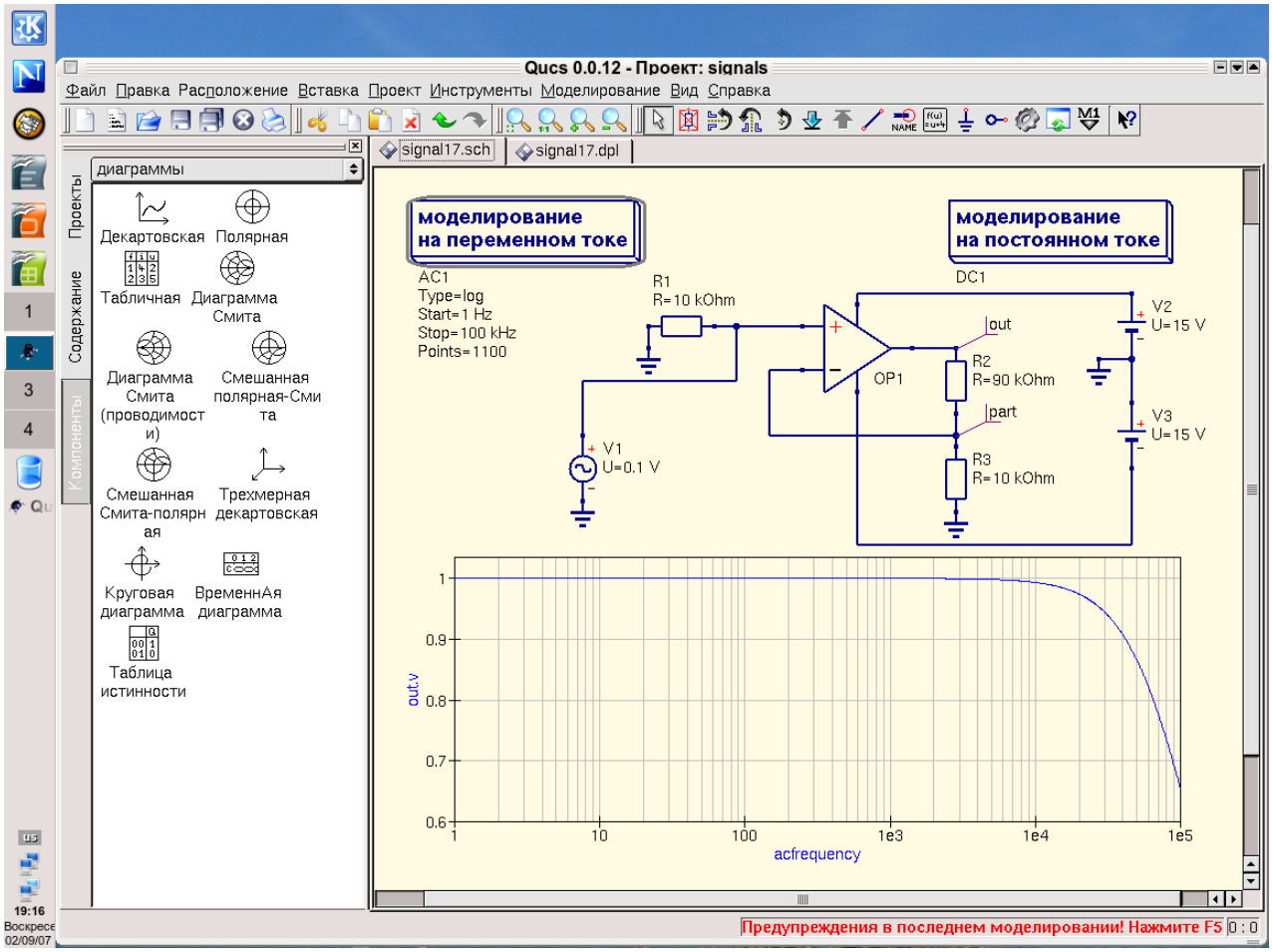


Рис. 5.16. Амплитудно-частотная характеристика ОУ с отрицательной обратной связью

Усиление, и это видно на диаграмме, снизилось, но полоса пропускания значительно расширилась, усиление спадает до нуля на частоте большей, чем 100 кГц. Полоса пропускания определяется плоской частью амплитудно-частотной характеристики. А верхняя граничная частота усиления определяется по спаду на 3 дБ. На рисунке 5.15 эта верхняя граничная частота приблизительно равна 10 Гц. На рисунке 5.16 она стала равна 100 кГц. За счет чего это удалось сделать обратной связи. За счет того, что она отрицательная – часть сигнала с выхода, с делителя напряжения, попадает на вход, где складывается в противофазе со входным сигналом, то есть, складывается, но с обратным знаком, или, что тоже самое, вычитается. Чем больше усиление, тем больше сигнал на выходе, тем больше его часть, попадает на вход, уменьшая входной сигнал. Когда усиление падает, эта возвратная часть сигнала уменьшается, словно увеличивая входной сигнал. В результате, независимо от начальной величины входного сигнала, отрицательная обратная связь осуществляет автоматическую коррекцию выходного сигнала, пока есть запас по усилению, определяемый исходным усилением схемы. Отрицательная обратная связь не только расширяет полосу пропускания усилителя, но и благотворно сказывается на остальных параметрах схемы, корректируя и стабилизируя их. А положительная обратная связь? Все делает с противоположным знаком, то есть, наоборот.

Здесь получается, что часть сигнала с выхода попадает на вход, складываясь с исходным сигналом. А это, в данном случае, приводит к увеличению выходного сигнала... и той части, что снимается с делителя напряжения на выходе, и что вновь приходит на вход, еще больше увеличивая входной сигнал. Возникает некое самоусиление или самовозбуждение. Именно на

этом принципе строятся многие генераторы. Но если рост усиления при положительной обратной связи еще как-то понятен, поскольку есть источник сигнала, то что усиливается, когда его нет, ведь генератор сам выступает в качестве источника переменного напряжения? Во-первых, не следует забывать о переходных процессах, возникающих при подключении питания к схеме. Кроме того, на входе любого усилителя есть особенный сигнал, который называется шум. Любой резистор на входе усилителя является источником шума – его электроны в хаотическом движении, а любой движущийся заряд порождает ток, создают переменные напряжения в широком спектре частот. Таким же свойством обладают полупроводниковые материалы, из которых сделаны транзисторы, особенно это касается транзисторов входных каскадов. К шумам относятся и все паразитные наводки. Так что, любой усилитель шумит. Шумы – вредный параметр для усилителя. Их стараются максимально снизить. Усилители звука по качеству классифицируются с учетом шумов, поскольку реальный звук по громкости изменяется в очень широком диапазоне, а усилитель со стороны самого громкого звука ограничен напряжением питания, а со стороны самого тихого звука ограничен шумами. Отношение максимального неискаженного выходного напряжения усилителя звука к напряжению шума, выраженное в децибелах, называют динамическим диапазоном усилителя. Чем этот параметр больше, тем качественнее усилитель. Но вернемся к положительной обратной связи.

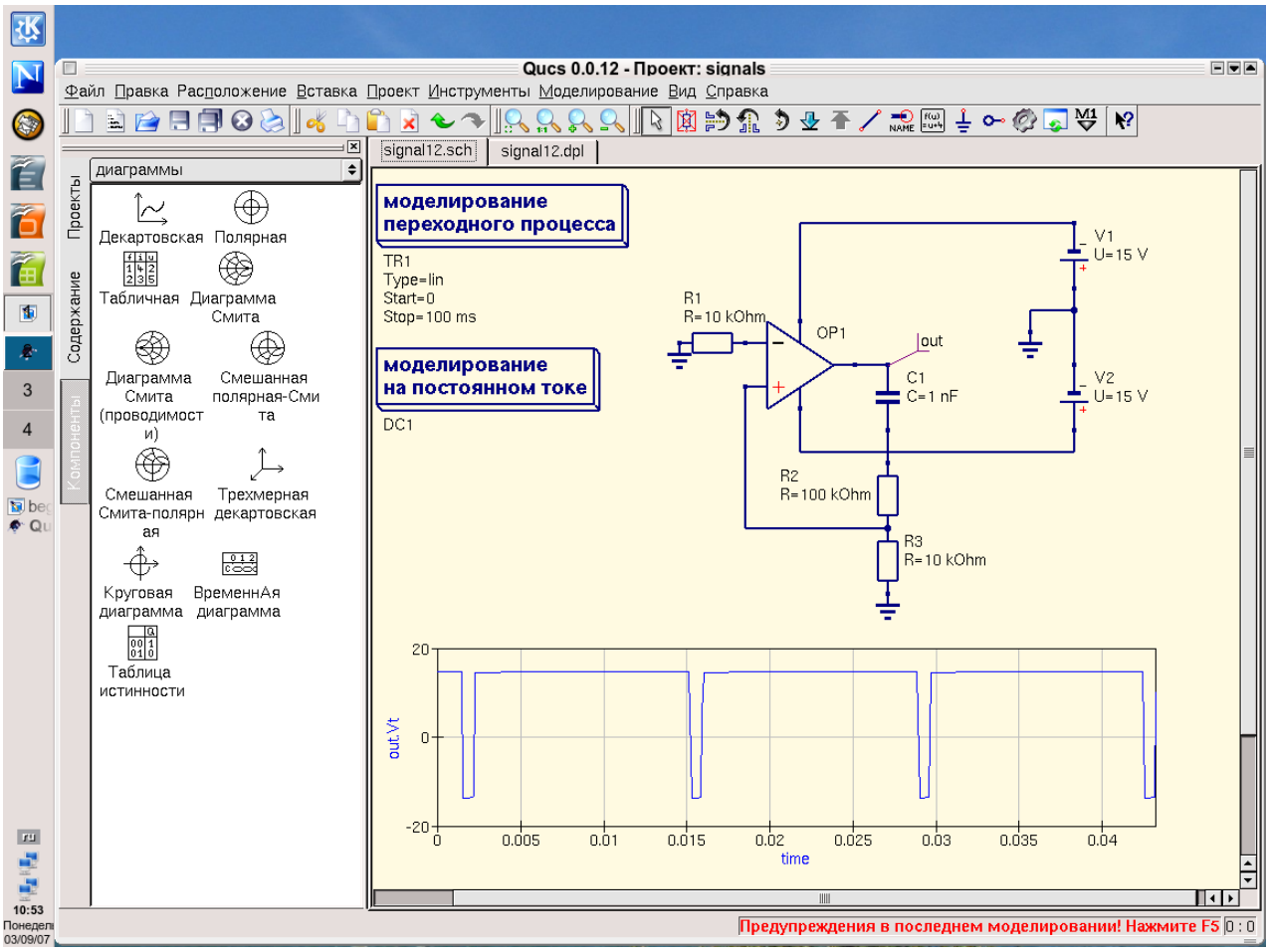


Рис. 5.17. Операционный усилитель с положительной обратной связью

Делитель напряжения на выходе операционного усилителя частотно-зависимый, поскольку кроме резисторов содержит конденсатор. Но в остальном, схема отличается от предыдущей только отсутствием источника сигнала и наличием положительной обратной

связи. Как можно видеть на осциллограмме на выходе (*out*) присутствует сигнал, генерированный операционным усилителем. Введя положительную обратную связь, мы превратили усилитель в генератор.

Я уже говорил, что на практике чаще других используют синусоидальный сигнал. Для формирования синусоидальных сигналов звуковой частоты применяют фильтры. Но использовать LC фильтр на частотах 50-400 Гц затруднительно из-за больших габаритов катушки индуктивности. Поэтому применяют более компактные RC фильтры. Посмотрим, какую амплитудно-частотную характеристику имеют два фильтра, имеющих по два элемента – резистор и конденсатор.

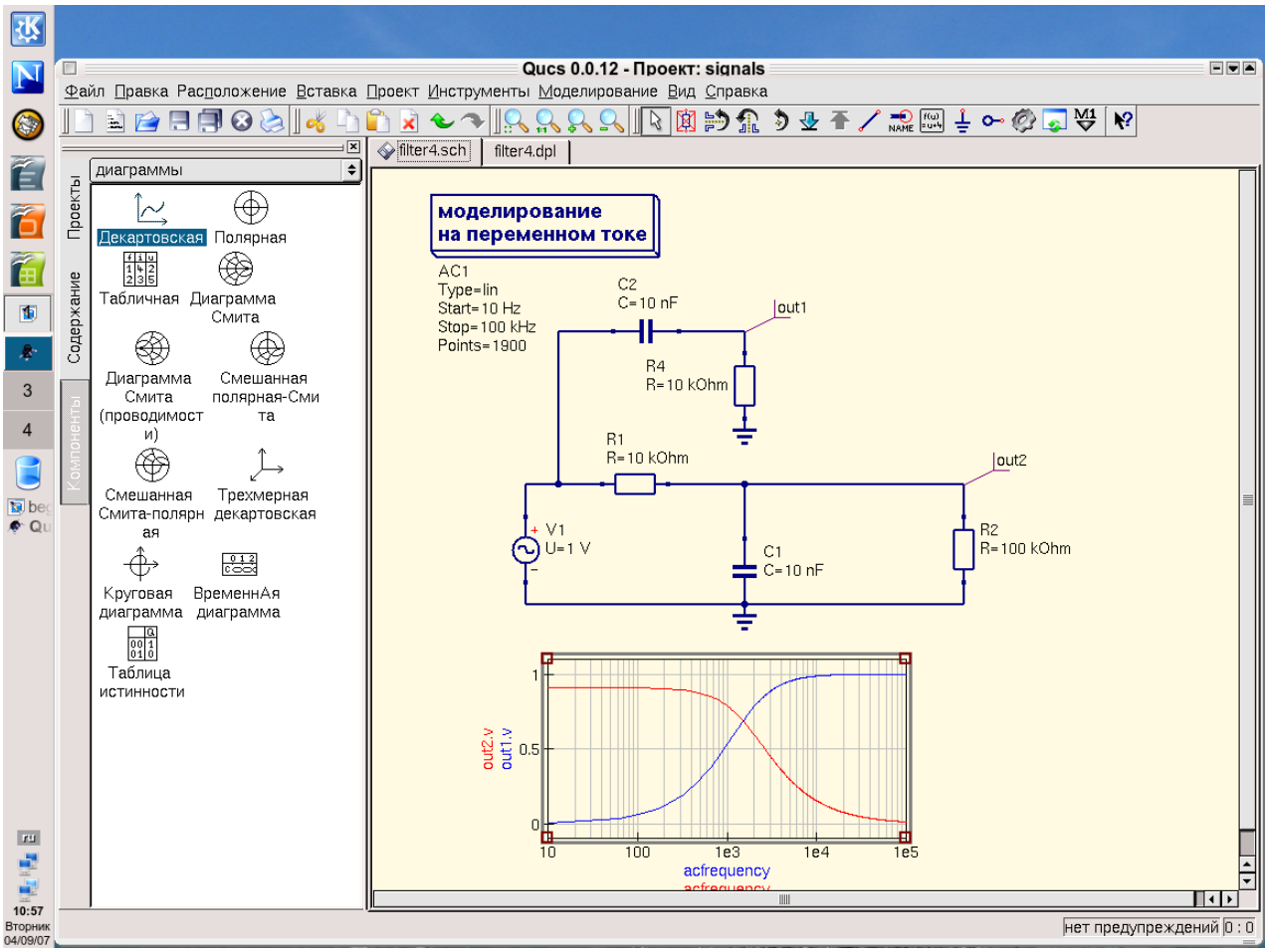


Рис. 5.18. Амплитудно-частотные характеристики двух фильтров

Из диаграммы видно, что один фильтр хорошо пропускает низкие частоты, но плохо высокие, другой наоборот, хорошо пропускает высокие частоты и плохо низкие. Их называют фильтрами высоких и низких частот. Но особенно меня интересует та точка графика, где обе кривые пересекаются. Нельзя ли это использовать для получения сигнала только одной частоты?

Попробуем объединить оба фильтра в один, и посмотрим на его амплитудно-частотную характеристику.

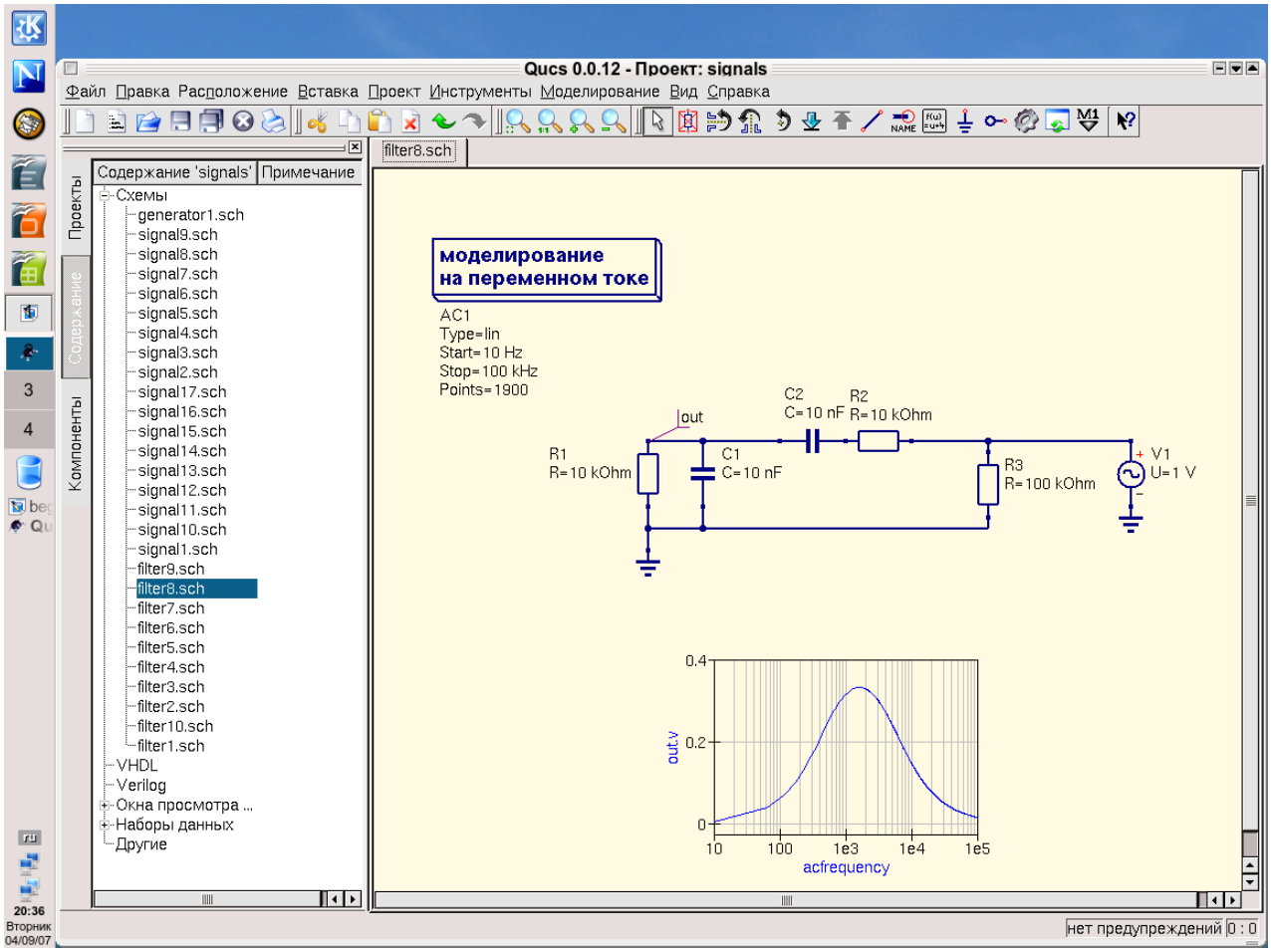


Рис. 5.19. Амплитудно-частотная характеристика комбинированного фильтра

Такой фильтр, похоже, действительно наилучшим образом пропускает одну частоту. А если такой фильтр употребить в цепи положительной обратной связи усилителя?

В качестве усилителя возьмем два одинаковых каскада усиления на транзисторах. Правда эти транзисторы я выбрал из библиотеки программы, как реальные 2N2222, для верности. В программе Qucs в разделе *Инструменты* есть подраздел *Библиотека компонентов,* где располагаются модели реальных элементов – транзисторов, операционных усилителей и т.д. Мне сейчас не хочется особенно задумываться о настройке каждого каскада, поэтому я выбираю самую типовую схему включения транзистора и проверяю только работоспособность одного каскада усиления, включив источник переменного напряжения на его вход и посмотрев сигнал на выходе. Значения резисторов я выбираю, как говорится «на вскидку», конденсаторы беру «побольше». На всякий случай, задав еще несколько точек наблюдения, проверяю те точки, где будет включена цепь обратной связи, чтобы убедиться, что обратная связь будет положительной. Проверив один каскад, я копирую его и вставляю в схему, как второй каскад. Позже, экспериментируя с двухкаскадным усилителем, я возможно изменю значения резисторов и конденсаторов, но позже, а сейчас мне хочется убедиться в работоспособности схемы. На схеме ниже пара C1R2 соответствует R1C1, а C3R8 – C2R2.

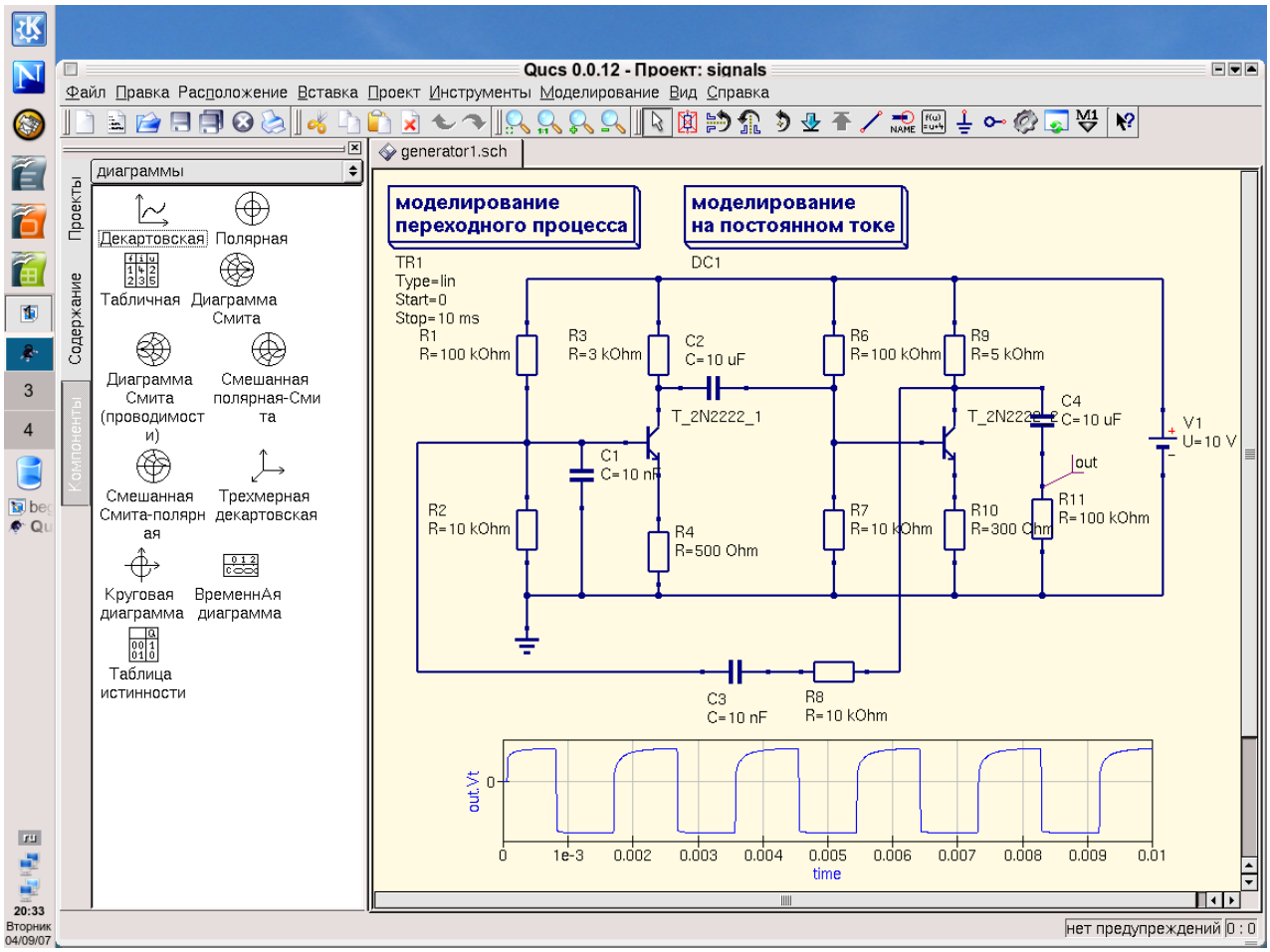


Рис. 5.20. Схема транзисторного усилителя с положительной обратной связью

Уменьшая величину конденсаторов C1 и C3 в десять раз, до величины 1000 пФ, можно получить десятикратное увеличение генерируемой частоты.

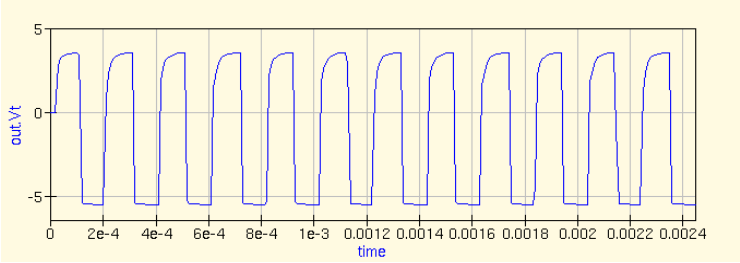


Рис. 5.21. Увеличение частоты в десять раз при уменьшении величины конденсаторов

И, естественно, сигнал так же похож на синусоиду, как я на старательного разработчика – «тяп, ляп, клеточка». Но я хитрый, я попробую добавить к положительной обратной связи отрицательную. Не зря же я добавлял точки проверки. Не пропадать же труду впустую!

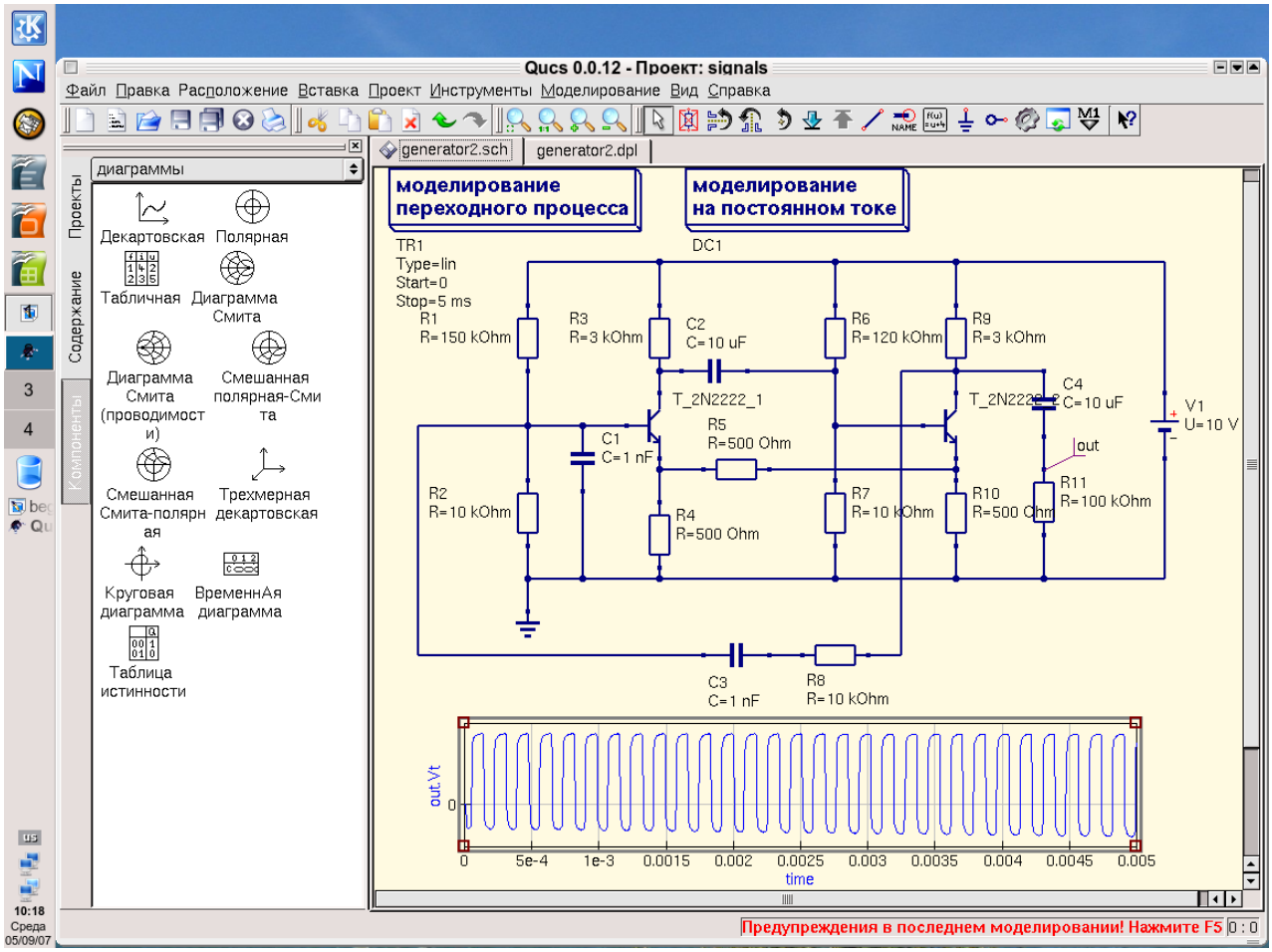


Рис. 5.22. Усилитель с положительной и отрицательной обратными связями

Хотя и не синус, но немного лучше. В реальных схемах, такой тип генератора называется схемой Вина, где применяют комбинацию положительной с фильтром и отрицательной обратной связи для получения синусоидального сигнала. Для стабилизации выходного напряжения и улучшения условий запуска генератора в отрицательную обратную связь вводят дополнительный нелинейный элемент. Но это уже история о генераторах, а не о сигналах. Сейчас появилось много хороших функциональных генераторов, вырабатывающих сигналы синусоидальной, прямоугольной и треугольной формы. И параметры у них достаточно хорошие. Но, все-таки, это другая история.

Использование сигналов

Один из самых популярных способов использования сигналов – радиосигнал. Мобильный телефон, без которого многие сегодня не способны обходиться, радиоприемник, телевизор, компьютер с Wi-Fi подключением или спутниковым Интернетом, разного рода беспроводные датчики в автоматике, навигационное и локационное оборудование, устанавливаемое даже в автомобиле. И даже привычное «радиолюбитель» для тех, кто интересуется современной электроникой. Это все — радиосигнал.

Для использования радиосигнала нужны два устройства: передатчик и приемник. Если передатчик преобразует информацию, скажем звук, в радиоволну, то приемник должен превратить радиоволну в звук. Как это происходит?

Самые первые радиоприемники, которые создавались радиолюбителями, бывшими тогда

именно РАДИОлюбителями, были детекторные приемники. Радиостанции в то время в основном использовали амплитудную модуляцию несущей радио частоты. То есть, амплитуда радиосигнала менялась по закону изменения звука. Когда радиоволна достигала приемника, то вызвала появление тока в антенне, этот ток проходил через колебательный контур, настроенный на частоту несущей радиоволны, выделяя нужную частоту из множества других. Чаще настройка на станцию производилась с помощью конденсатора переменной емкости, но были и решения с перестройкой индуктивности. После выделения колебательным контуром нужной частоты радиосигнала смесь из полезного сигнала и слабых сигналов других станций попадала на детектор. Детектор – это, в сущности, выпрямитель. Мы знаем, что выпрямитель превращает переменное напряжение в постоянное. Не превратим ли мы после детектирования радиоволну в очень плохой источник питания с постоянным напряжением в несколько милливольт? Посмотрим.

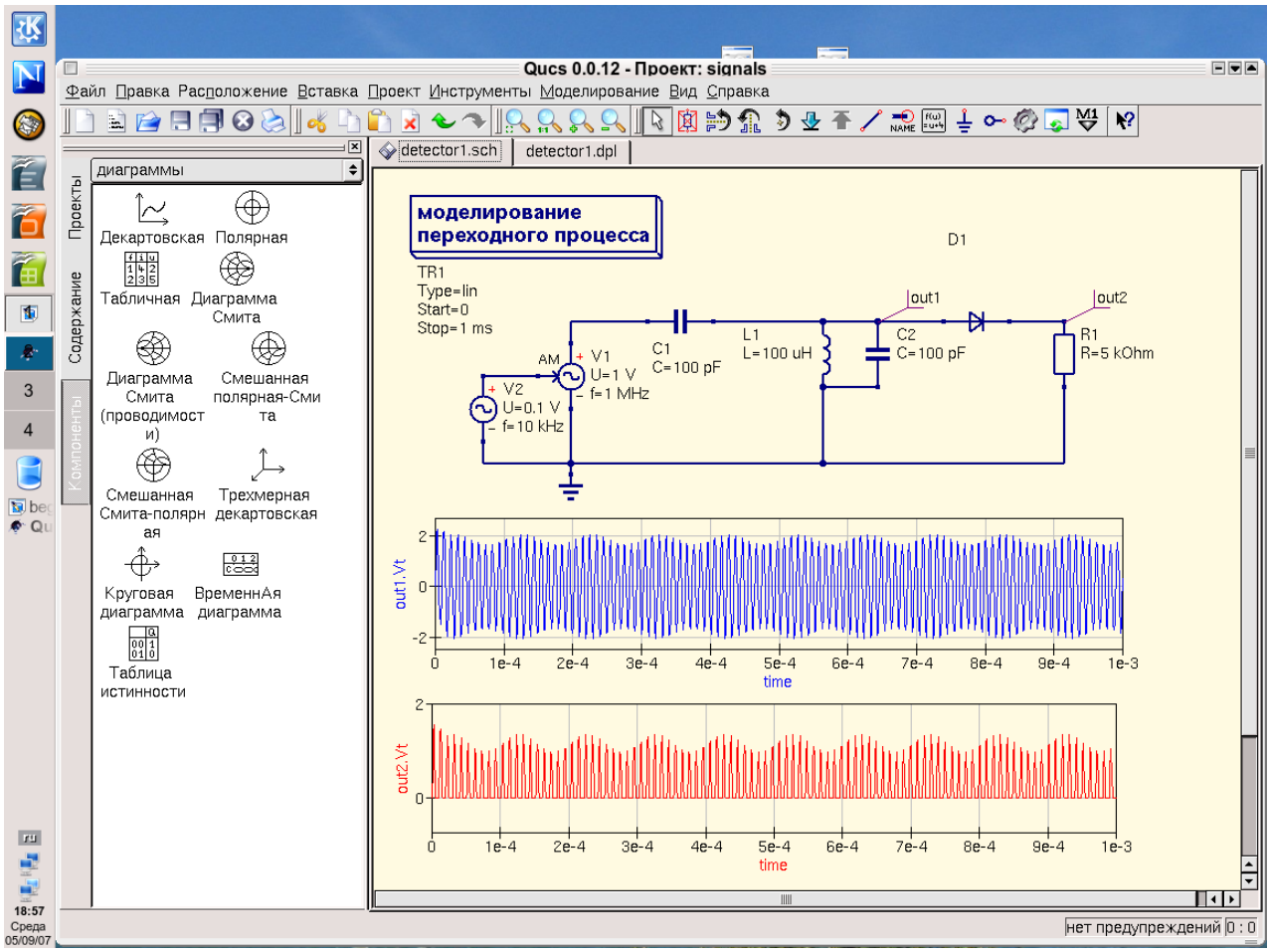


Рис. 5.23. Детекторный приемник

Здесь в роли передающей радиостанции выступают два источника напряжения: V1, как передатчик с несущей частотой 1 МГц и V2, как диктор, умеющий свистеть на частоте 10 кГц и сидящий у микрофона передатчика V1. Конденсатор C1 изображает передающую антенну, безбрежное пространство и приемную антенну, растянутую в комнате, где на столе разложен детекторный приемник из контура L1C2, детектора D1 и наушников R1 (нас интересует пока только активное сопротивление наушников). Сигнал, полученный антенной детекторного приемника (*out1*), изображен на верхней диаграмме, а в наушниках (*out2*) на нижней. После детектирования, к счастью, мы получаем не постоянное напряжение, а переменное, изменяющееся по величине, и к нашему удовлетворению, это видно на диаграмме,

изменяющееся по тому же закону, по которому менялась амплитуда несущей частоты передатчика. «Огибающая» на обеих диаграммах – это синусоидальный сигнал с частотой 10 кГц.

По своей всегдашней привычке, родственнице лени, я значения элементов колебательного контура задал «на вскидку». Но в реальном детекторном приемнике настраиваются с помощью переменной емкости. В программе Qucs осуществить похожую подстройку тоже можно.

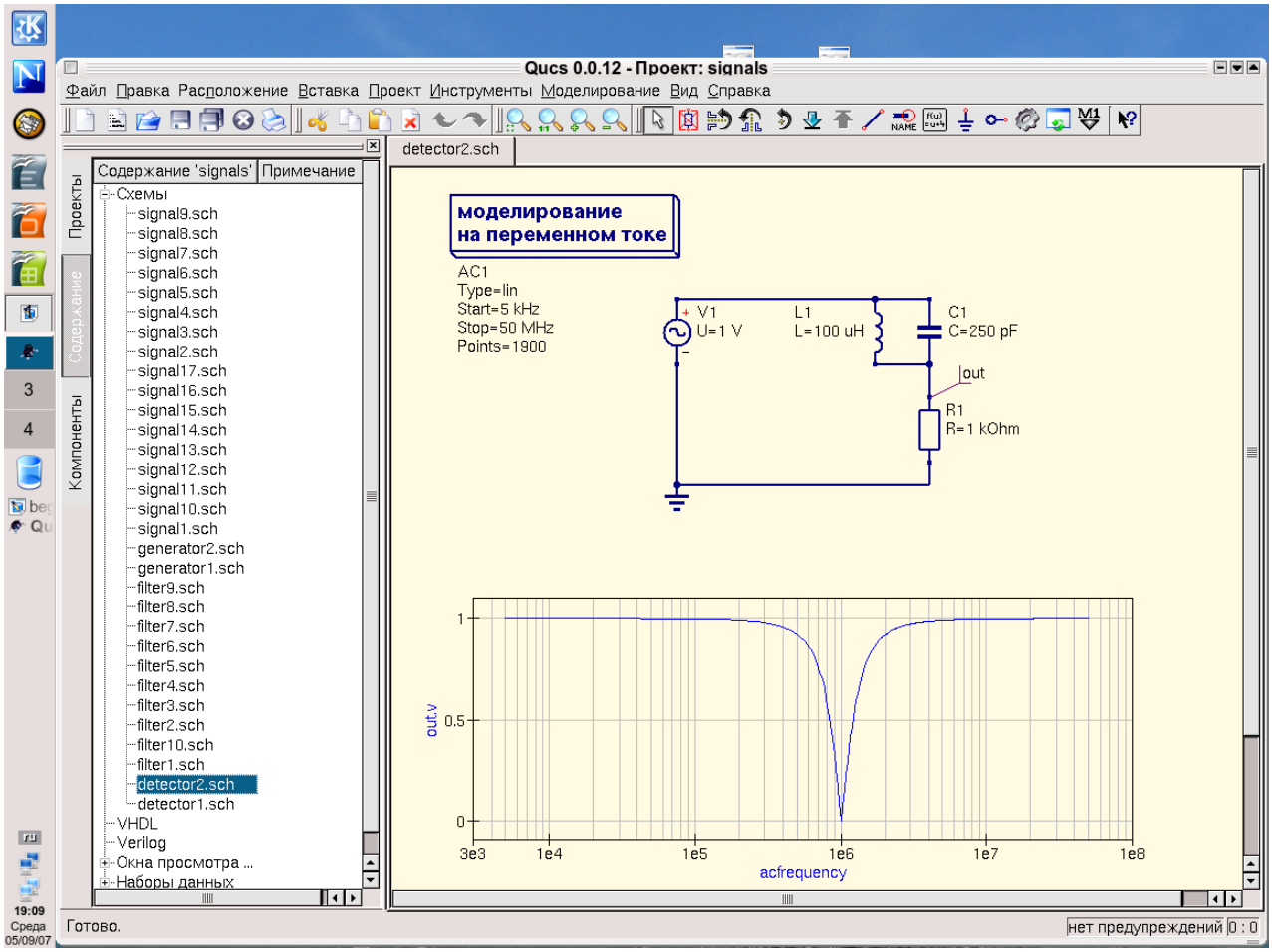


Рис. 5.24. Настройка колебательного контура на частоту 1 МГц

Чтобы полезный сигнал, огибающая, был больше похож на синусоиду параллельно резистору R1 в схеме детекторного приемника включим конденсатор. Эту операцию можно рассматривать двояко – на выходе выпрямителей включают конденсатор, который «сглаживает» выпрямленное напряжение, делая его больше похожим на то, что получается от батарейки; а можно рассуждать так, что конденсатор имеет маленькое сопротивление на частоте несущей и шунтирует наушники, а на частоте огибающей конденсатор имеет сопротивление во много раз большее, и не мешает полезному сигналу.

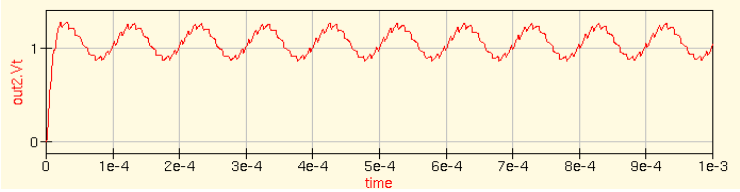


Рис. 5.25. Шунтирование несущей частоты конденсатором

Вот так изменяется вид сигнала после шунтирования наушников конденсатором емкостью 3000 пФ.

Простота конструкции детекторного приемника вызывает глубокий восторг. Когда-то первые радиолюбители ловили первые передачи радиостанций на такой приемник. Не знаю, что они использовали в качестве конденсаторов, но не думаю, что они могли использовать в качестве детектора германиевые диоды, и даже ламповые детекторы. Но что-то использовали. Жаль, что я не застал те времена, мне кажется они были славными. Когда мой сын был маленьким, я рассказывал ему о детекторном приемнике, и мне кажется, пытался повторить подвиг первых радиолюбителей, хотя не уверен, что успешно. Больше я не пытался это сделать. Не только потому, что плохо умею это делать, но сегодня эфир так заполнен радиосигналами, что даже, добавив к детекторному приемнику каскад усиления высокой частоты и каскад усиления низкой частоты (мне кажется, такая схема называется 1-V-1), трудно получить полезный сигнал хорошего качества на фоне мешающих радиосигналов.

С развитием вещания, с появлением все большего количества радиостанций схемы приемников стали усложняться. Каждая радиостанция помимо основной частоты вещания, несущей, занимает некоторую полосу, а в каждом частотном диапазоне вещания таких полос выделено ограниченное количество. Приемники требовали все более точной настройки на частоту передающей станции, иначе прослушивались, мешая, соседние станции, необходимо было усиливать выделенный сигнал несущей частоты, но не за счет увеличения каскадов усиления высокой частоты. В некоторых схемах радиоприемников применяли положительную обратную связь. Это улучшало условия настройки, но приемники имели тенденцию к самовозбуждению. И тогда...

Появился приемник супергетеродинного типа. Сущность такого приемника проста. И как все простое, это решение, как мне кажется, трудно было найти. Идея новой схемы в том, чтобы превратить радиосигнал любой радиостанции в сигнал с фиксированной несущей частотой. Для такой частоты можно было использовать усилители с большим коэффициентом усиления, то есть, очень значительно усилить полученный сигнал. Для получения этой фиксированной, или промежуточной частоты, использовали смешивание сигналов двух частот на нелинейном элементе. В результате кроме основных частот должно получиться еще две частоты, разностная и суммарная. Разностная, ниже частоты радиосигнала, выделяется колебательным контуром, настроенным на эту частоту, а затем может усиливаться каскадами усиления промежуточной частоты. Усилитель, предназначенный для усиления одной частоты, может иметь значительно больший коэффициент усиления, чем широкополосный, особенно, если это не слишком высокая частота. Выглядеть это должно, приблизительно, так.

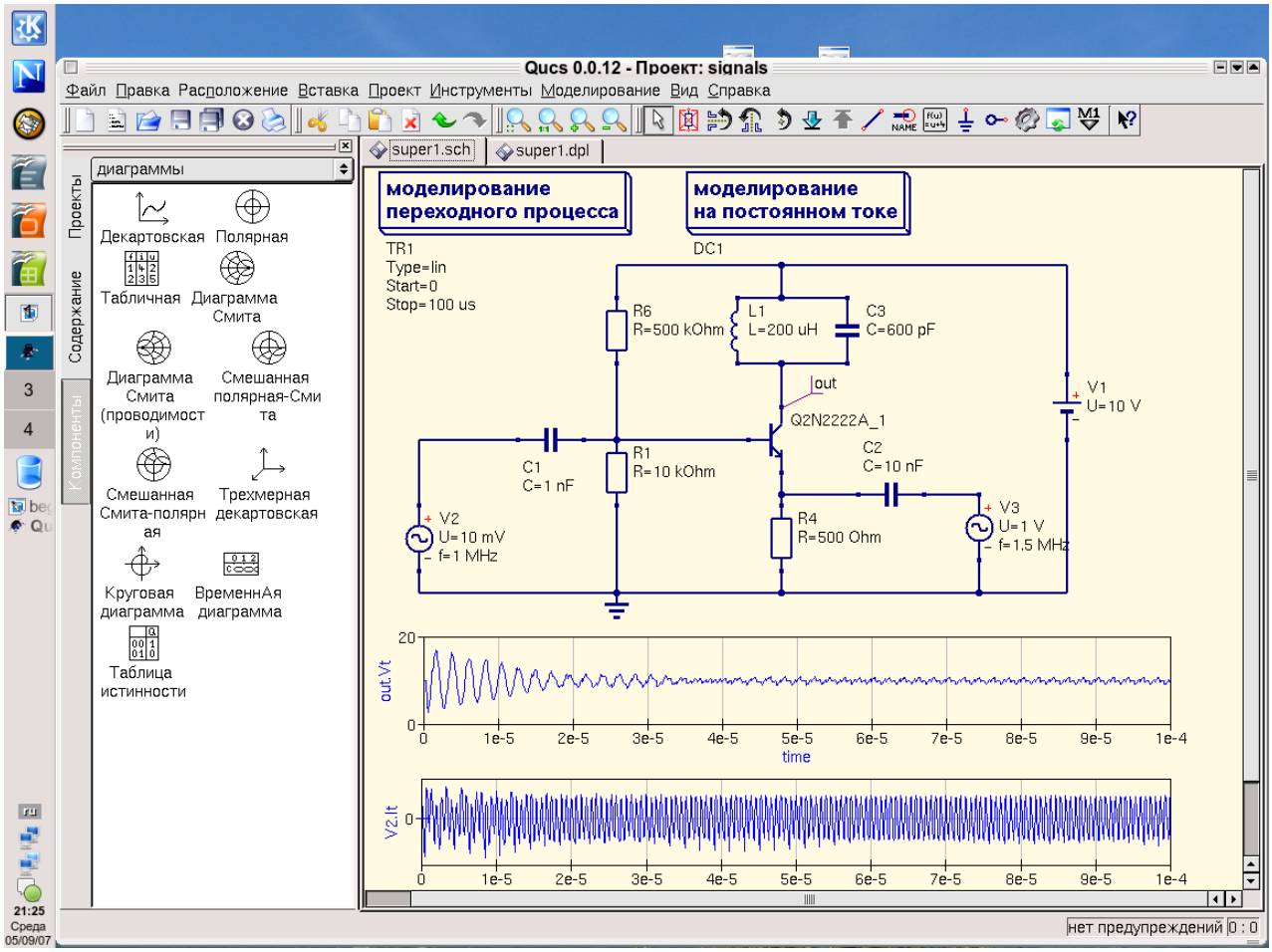


Рис. 5.26. Смешивание двух частот на нелинейном элементе

Верхняя диаграмма (сигнал *out*) отображает сигнал промежуточной частоты, а нижняя, скорее, смесь всех сигналов: от генераторов V2 и V3, и сигналов суммарной и разностной частоты. Если нелинейные искажения в усилителях звуковой частоты мешали, то искажения в этом случае оказываются очень полезны.

Может сложиться впечатление, что чем сложнее, тем полезнее сигнал. Это не так. Самый простой вид сигнала – это сигнал с делителя напряжения, один из резисторов которого терморезистор. Зная, каким температурам соответствуют какие напряжения, можно следить за напряжением на терморезисторе, отображая его в виде температуры. Мы получаем электронный термометр. Напряжение на терморезисторе, очень медленно изменяющееся, тоже вполне можно назвать сигналом. Если мы хотим управлять температурой, положим в помещении, то, используя этот сигнал, мы можем включать и выключать обогреватель.

При управлении температурой в технологических процессах, когда нужна большая точность, а температуры могут достигать больших значений, вместо терморезистора используют термопары. Этот элемент генерирует постоянное напряжение очень маленькой величины. Для его использования сигнал необходимо усиливать. Усиление маленького постоянного напряжения задача не из самых легких. Температура сказывается не только на состоянии датчика температуры, но и на всех элементах усилителя, самопроизвольно меняя коэффициент усиления. Чтобы обойти эту проблему, маленькое медленно меняющееся напряжение превращают, например, в импульсное переменное напряжение, амплитуда которого равна исходному напряжению. Теперь можно усиливать быстро меняющееся напряжение усилителем переменного напряжения, коэффициент усиления которого

значительно стабильнее. Как происходит преобразование одного сигнала в другой?

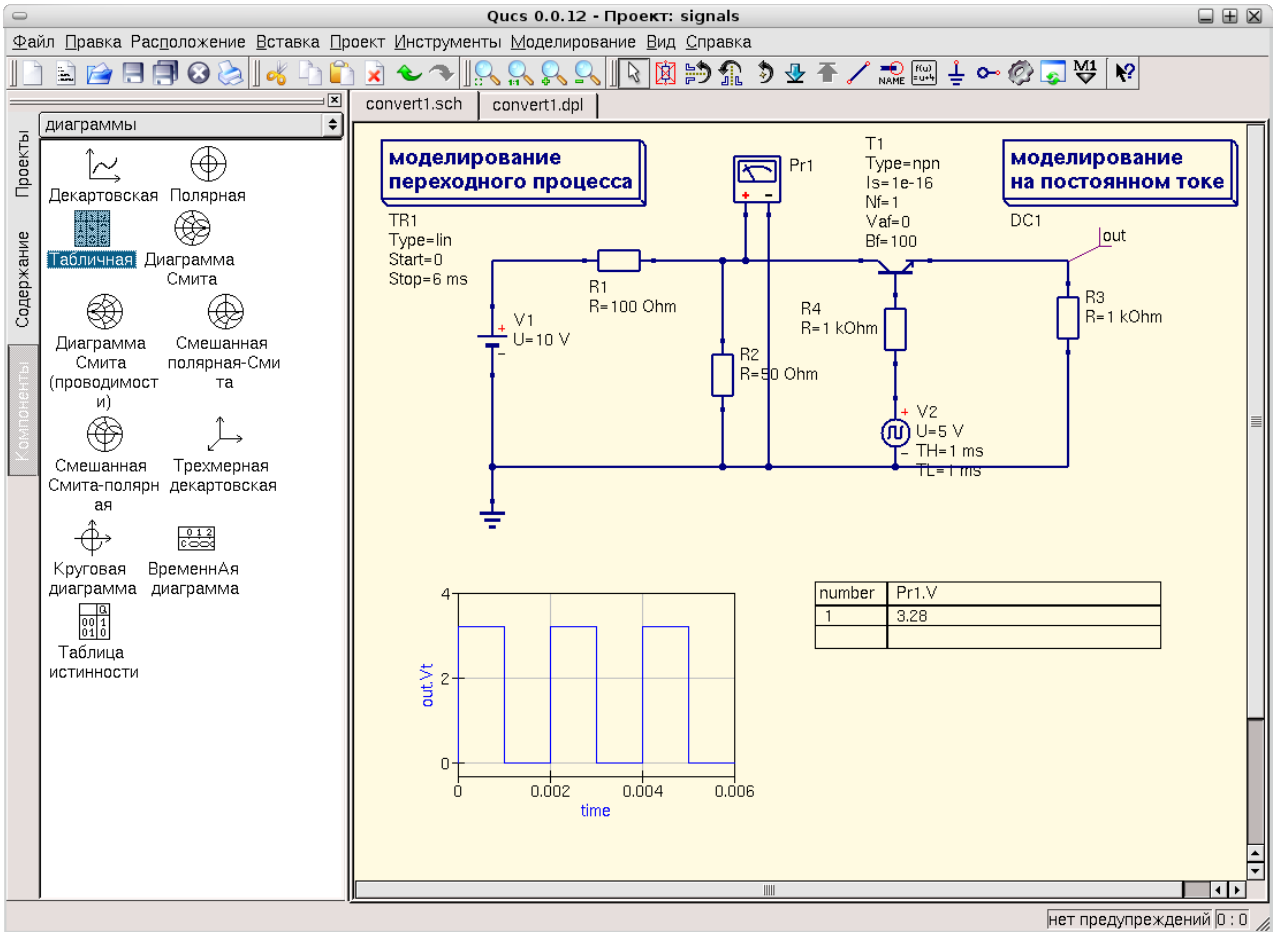


Рис. 5.27. Преобразование сигнала делителя постоянного напряжения в импульсный

На рисунке резистор R2 – это терморезистор. Транзистор T1 работает в режиме ключа, управляемый генератором прямоугольных импульсов V2. Напряжение на терморезисторе с помощью ключевого транзистора преобразуется в импульсы прямоугольной формы, амплитуда которых соответствует напряжению на терморезисторе. Чтобы проверить это, я включил измеритель напряжения, который показывает 3.28 В.

Теперь изменим напряжение на терморезисторе. Для этого можно изменить величину этого резистора, что происходит при изменении температуры и отвечает реальности, или можно изменить напряжение питающего источника постоянного напряжения V1, что не изменит, или не должно изменить, результат, но не происходит в реальности. Попробуем пока не лукавить и изменить величину резистора. Если это не получится, у нас останется «лукавый» вариант в запасе.

Изменение напряжения на терморезисторе должно привести к увеличению амплитуды прямоугольных импульсов.

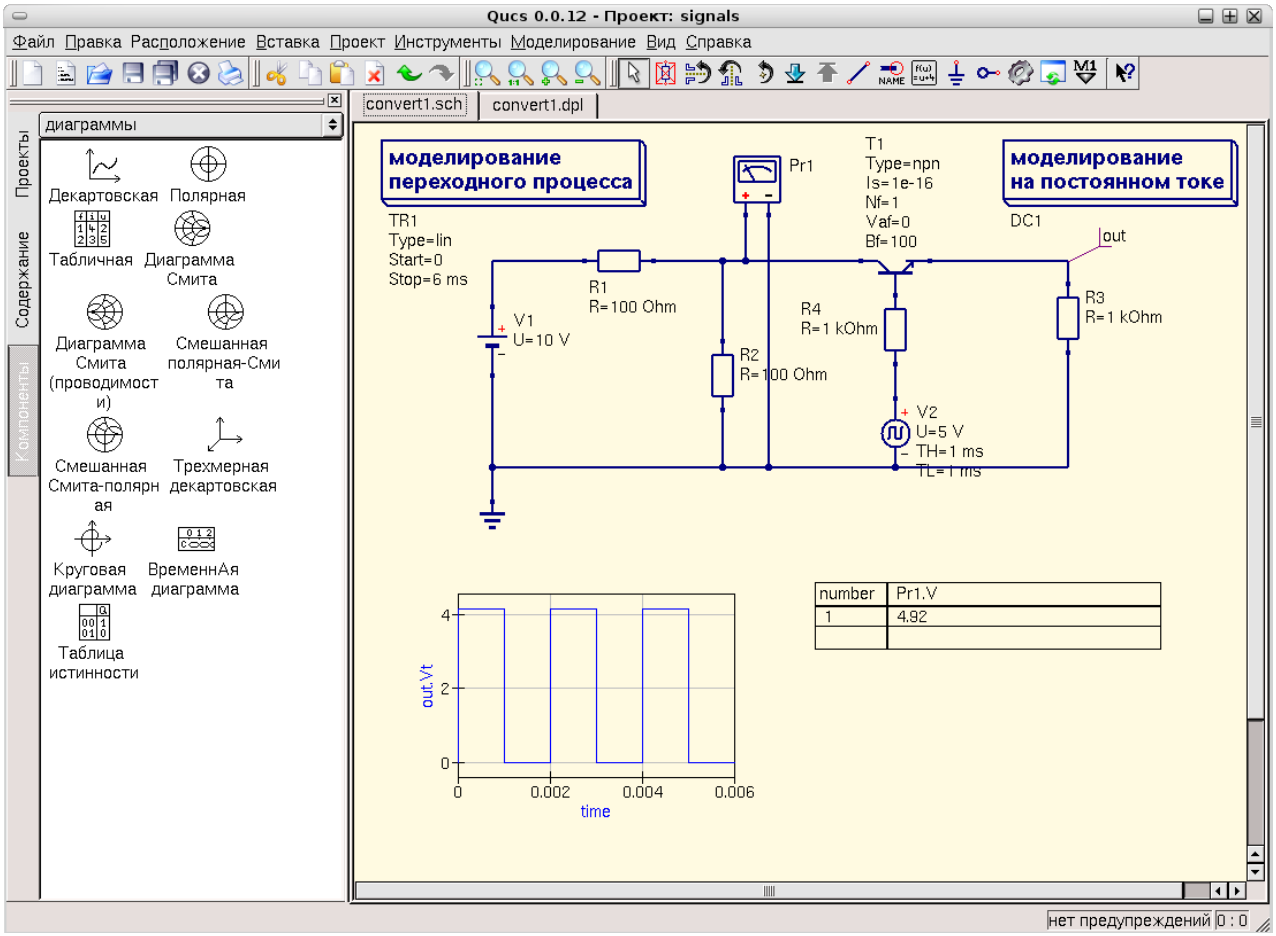


Рис. 5.28. Изменение амплитуды импульсов при изменении «температуры»

Кстати похожее преобразование используется в современных блоках питания, где отказ от традиционного понижающего трансформатора снижает вес блока во много раз.

Импульсные сигналы особенно интенсивно используются в цифровых схемах. Как правило, это сигналы уровня логических единицы и нуля, разные для цифровых микросхем, выполненных по разным технологиям, но зато в цифровых сигналах еще больше разнообразия: сигналы данных, адресные сигналы, сигналы управления (и помех).

Забегая вперед, я хочу показать, как сигналы данных записываются в D-триггер. Триггер – устройство, имеющее два устойчивых состояния, любое из них может сохраняться до прихода следующего управляющего импульса, пока триггер подключен к источнику питания. В цифровой схемотехнике применяют несколько подвидов триггеров: RS-триггер, T-триггер, JK-триггер, D-триггер и т.д. На основе триггеров можно легко построить схему для записи, например, байта данных. Подобные микросхемы носят название регистров и являются непременным атрибутом всех устройств, входящих в состав компьютера.

Триггер можно рассматривать как нечто целостное, как неделимую сущность с определенными свойствами. И для кого-то подобный подход может оказаться полезнее, чем рассмотрение устройства триггера. Такая точка зрения близка к современному подходу в программировании, в том что касается объектно-ориентированного программирования. Но я не люблю «черных ящиков», всегда испытываю непреодолимое желание посветить внутрь хотя бы фонариком. Посмотрим, как сигналы используются в цифровой технике, а потом решим, как поступить с триггером.

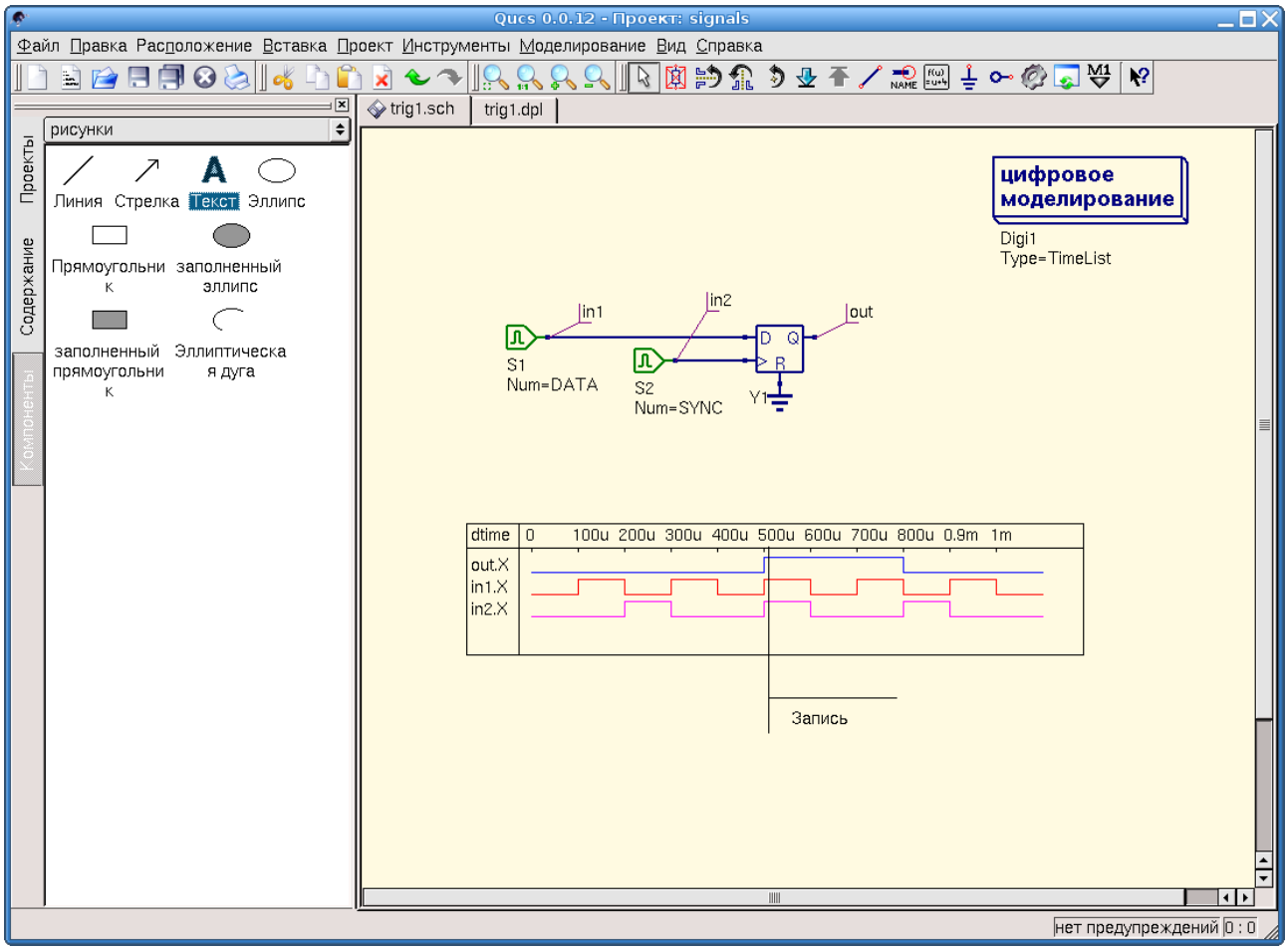


Рис. 5.29. Запись данных в D-триггер

Исходно выход триггера находится в состоянии логического нуля (*out*). На его вход D (*in1*) от источника сигнала S1 приходят импульсы, которые я назвал DATA, на вход записи (*in2)* приходят тактовые импульсы. Когда на входе D появляется сигнал с уровнем логической единицы, фронтом тактового импульса этот сигнал записывается в триггер. Я уже говорил о временных искажениях, возникающих в цифровых устройствах. Если, как это видно на диаграмме в момент времени, обозначенный как «Запись», сигнал данных чуть-чуть запоздает, то состояние выхода триггера не изменится. Чтобы избежать подобных ситуаций, стараются так организовать схему, чтобы данные устанавливались несколько раньше прихода сигнала записи. Конечно, это замедлит работу устройства в целом, но повысит его надежность. Очень часто построение схемы устройства – это компромисс между скоростью и надежностью, качеством и стоимостью, простотой и габаритами и т.д. Оптимальное соотношение, как правило, результат многократных экспериментов, опыта и знаний. И даже любителю не стоит пренебрегать проверкой любого решения на макетной плате, и не однократной, а многократной, по возможности в разных условиях эксплуатации. Иначе получается так, что схема, работавшая вполне убедительно при проверке, отказывается работать именно тогда, когда в ней есть нужда, когда пришло время насладится плодами своего труда.

Касательно устройства триггера. Собственно говоря, он не будет содержать никаких элементов, с которыми мы бы уже не познакомились. Транзисторы, резисторы и конденсаторы – все уже знакомо. Не вижу причин, отчего бы не рассмотреть, как устроен триггер изнутри, пусть самый простой. Схема ниже поможет в этом разобраться.

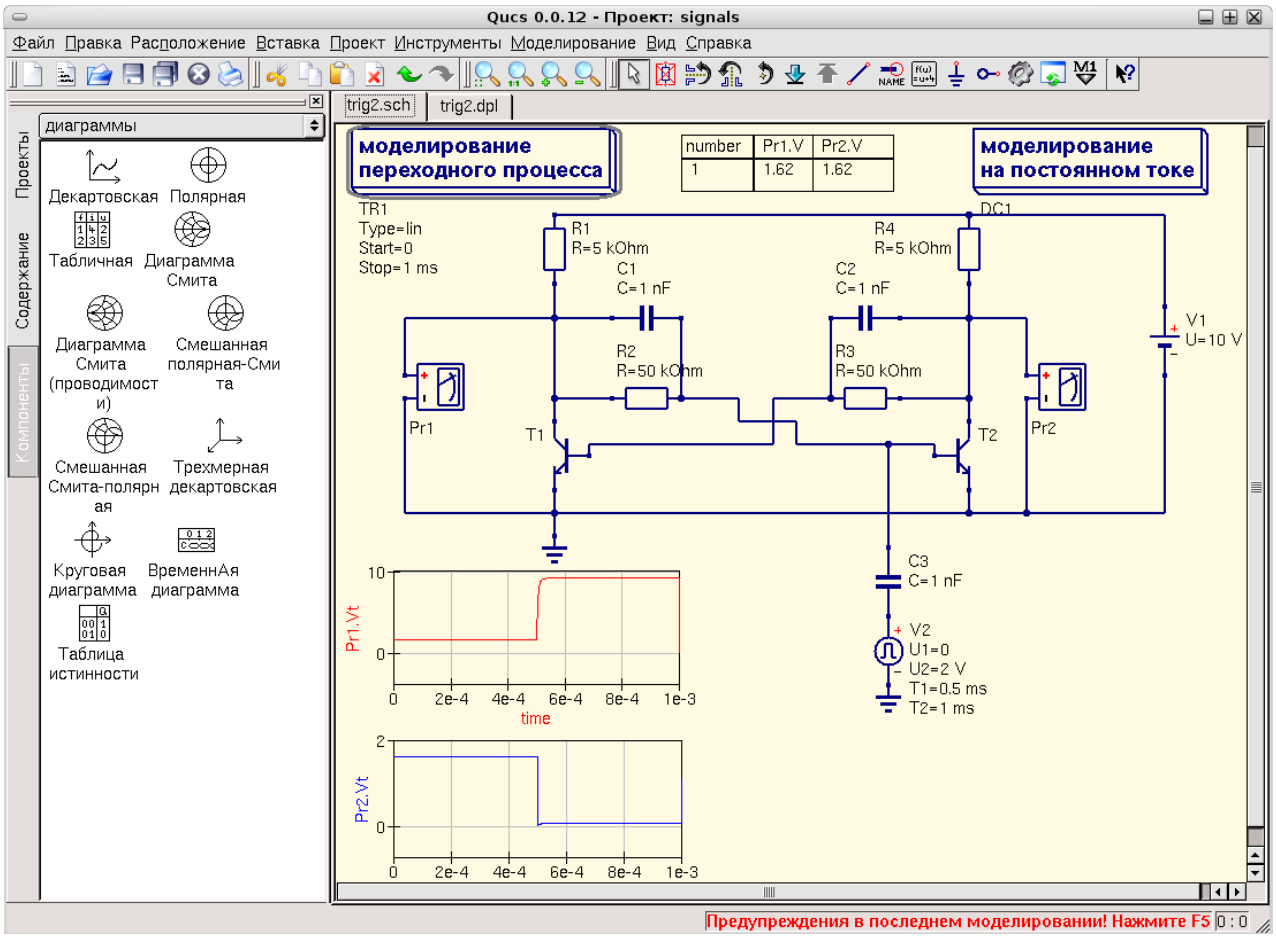


Рис. 5.30. Пояснения к работе триггера

Схема, если не учитывать импульсный генератор V2, абсолютно симметрична. Резисторы R2 и R3 определяют базовые токи транзисторов, транзисторы открываются, а коллекторные токи определяют падения напряжения на резисторах R1 и R4. Для триггера это начальное состояние является неопределенным. В нашей схеме на коллекторах транзисторов напряжения равные (измеренные вольтметрами Pr1 и Pr2) и составляют 1.62 В. Этому состоянию соответствует на диаграммах временной отрезок от 0 до 0.5 мС. В этот момент от генератора V2 приходит управляющий положительный импульс, нарушающий исходную симметрию, равновесие. Базовый ток транзистора T2 увеличивается, увеличивается падение напряжения на резисторе R4, транзистор T2 полностью открывается, а напряжение на его коллекторе оказывается близким к нулю, что видно на нижней диаграмме. Ток через резистор R3 уменьшается (из-за снижения напряжения на коллекторе T2), транзистор T1 закрывается, а падение напряжения на резисторе R1 уменьшается (из-за уменьшения коллекторного тока транзистора T1). Последнее приводит к тому, что резистор R2 оказывается подключен к источнику питания. До этого напряжение на нем определялось разностью напряжения источника питания и падения напряжения на резисторе R1. Базовый ток транзистора T2 увеличивается, транзистор T2 окончательно и бесповоротно полностью открывается, симметрия нарушена, равновесия нет. Триггер принимает устойчивое состояние при котором на одном его выходе (коллектор транзистора T2) напряжение близкое к нулю, а на втором близкое к напряжению питания (что видно на верхней диаграмме).

В реальной схеме симметрия обычно не достигается и из-за разброса параметров всех элементов схемы, и из-за наличия бросков питающего напряжения при включении, и из-за шумов, которые всегда присутствуют в схеме. В этом случае особенной оказывается роль

конденсаторов C1 и C2. В начальный момент оба конденсатора заряжаются через сопротивления в цепи коллекторов транзисторов и их переходы база-эмиттер, работающие как диоды. Положим, что из-за асимметрии схемы транзистор T2 полностью открылся, а транзистор T1 закрылся. При этом конденсатор C2 через открытый транзистор T2 оказывается подключен к базовой цепи транзистора T1 (между его базой и эмиттером), при этом на обкладке конденсатора, соединенной с базой, минус. При таком включении транзистор T1 еще лучше будет закрыт – переход база-эмиттер смещен в обратной полярности, базовый ток почти полностью отсутствует. И эта ситуация поддерживается до тех пор, пока конденсатор не разрядится через резистор R3.

Но вернемся к сигналам.

Их многообразие обусловлено широким применением электроники в нашей жизни. Сегодня многие не представляют своей жизни без телевизора, компьютера или мобильного телефона. Не за горами время, когда все это объединится в одном универсальном коммуникаторе, но чтобы это время пришло, нужны те, кто совершенствует существующие и создает новые устройства. Многие специалисты приходят в свою профессию из детских лет любопытства и школьных лет любительства. И будет весьма прискорбно, если славное племя радиолюбителей не будет пополняться молодыми энтузиастами. Благо сегодняшняя электроника это и математика, и физика, и химия. Всегда можно найти свой круг интересов в этой области.

Глава 6. С чего начать свой первый проект?

Во многом это будет зависеть от двух основных факторов – сферы интересов и оснащенности рабочего места. Я не советую начинать практическую деятельность с нужных, но не интересных вам предметов. Если вас в принципе не интересуют цифровые устройства, то просто запомните, что они есть, что есть множество книг по предмету, но не старайтесь что-то сделать для понимания даже основ работы с этими устройствами. Прежде, чем вы правильно спаяете ваше первое устройство, вы можете так разочароваться, что первое устройство станет и последним. Даже такое простое действо, как пайка, требует практических навыков.

Я не готов угадать, что интересно именно вам, но постараюсь в этой главе реализовать несколько разнородных проектов посильных для повторения начинающими. Это не будут конкретные схемы для повторения с наилучшими результатами, это будут именно проекты по созданию своей первой схемы.

Задумав книгу, я посчитал полезным обратиться на радиолюбительском сайте к начинающим радиолюбителям с просьбой помочь мне определиться в том, что интересно сегодня любителям? Мне показалась интересной сама мысль об интерактивном создании книги, содержание и стиль которой определили бы те, для кого она создается, а в технологии написания использовались современные средства коммуникации, такие как Интернет. Однако пока нашелся только один доброволец. К интерактивной работе с ним я приступаю.

Александр, вы спрашиваете, действительно ли, используя источник питания 12 В, можно от усилителя мощности получить не больше 5 Вт? Если я правильно понял вопрос, и ничего не перепутал, то давайте посмотрим, так ли это в первом проекте, который назовем «Усилитель мощности».

Усилитель мощности

В области звукоусиления есть несколько специфических функциональных названий усилителей. Есть специализированные усилители для отдельных источников сигналов. Например, микрофонный усилитель, или предварительный усилитель, или усилитель-корректор для звукоснимателя. Есть специализированные устройства, такие как микшер или эквалайзер. Все они находятся между источником сигнала и оконечным усилителем. Его задача – подготовить преобразование электрического напряжения от источника сигнала в звук. Подобное преобразование, если в качестве излучателя звука выступают не наушники, требует определенной мощности. Поэтому лучше такой усилитель называть усилителем мощности, даже если эта мощность невелика. Чаще даже уточняют в названии область применения такого усилителя – усилитель мощности звуковой частоты. Это справедливое уточнение, поскольку усилитель мощности может быть и в устройстве, работающем на ультразвуковых частотах, и на радио частотах и т.д. Схемные решения могут быть схожи, но требования к устройствам отличаются весьма значительно. Можно рассматривать типовые схемы, детально разбирая их особенности, но мне хочется подойти к этому проекту иначе.

Итак. Оконечный усилитель низкой частоты в своей сущности это усилитель и не более того. Схема усилителя на транзисторе приводилась выше. Попробуем использовать ее. Если мощности не хватит, возьмем транзистор мощнее, сегодня нет недостатка в транзисторах любых типов. В коллекторную цепь транзистора включим наш громкоговоритель. Получится нечто в следующем роде.

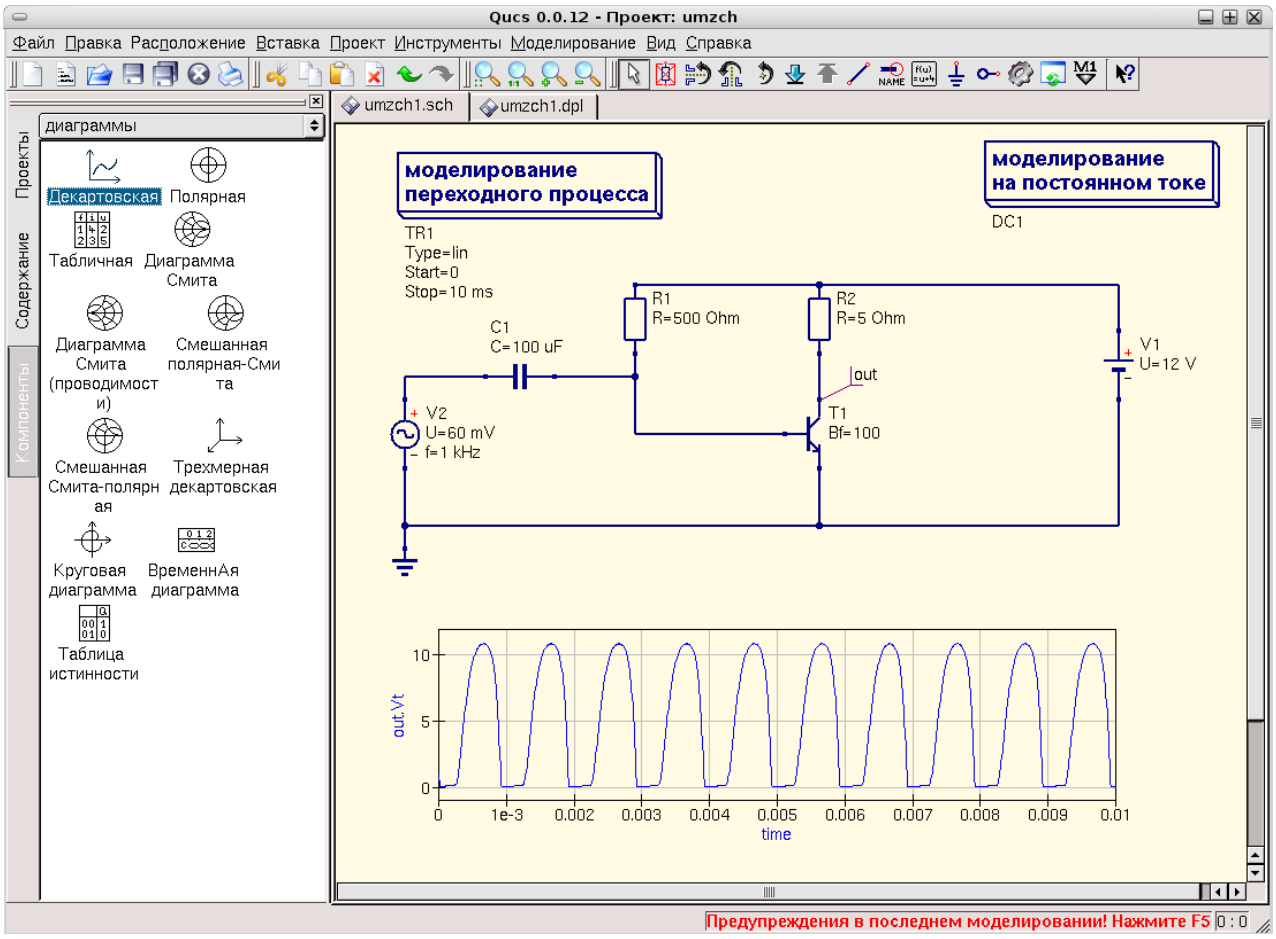


Рис. 6.1. Первая схема усилителя мощности низкой частоты

Если не обращать внимание на некоторые искажения в нижней части осциллограммы, то все выглядит не так уж плохо. Искажения в нижней части свидетельствуют о том, что транзистор полностью открывается при напряжении сигнала на входе 60 мВ и не может повторить форму входного сигнала. Попробуем уменьшать входной сигнал до тех пор, пока сигнал на выходе не станет больше походить на синусоиду.

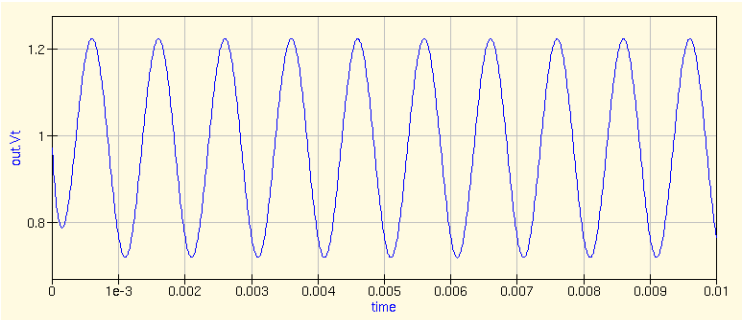


Рис. 6.2. Уменьшение входного сигнала для уменьшения искажений

Сигнал выглядит совсем не плохо. Входное напряжение пришлось уменьшить до 1 мВ, а выходное напряжение (амплитуда) уменьшилось до 200-250 мВ. Это, пожалуй, не интересно. Проверим, не получим ли мы лучший результат, если из набора моделей возьмем не абстрактный транзистор, а достаточно мощный. Например, BF720, у которого допустимая мощность рассеивания 1.5 Вт. Входное напряжение пришлось увеличить до 2.5 В, это не

страшно, но результат не слишком впечатляет.

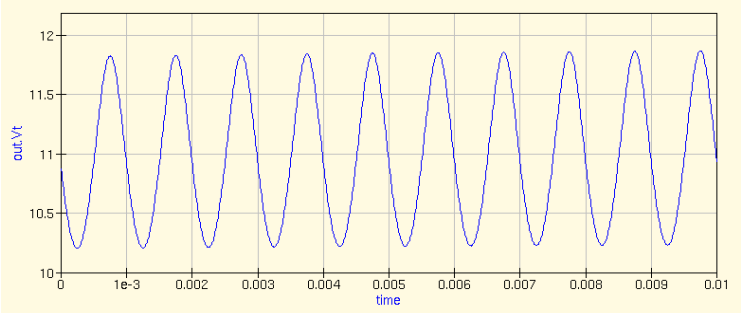


Рис. 6.3. Замена транзистора на более мощный

Выходное напряжение увеличилось до 0.6-0.7 В, но хотелось бы иметь его похожим на то, что было на рисунке 6.1. Это давало бы максимальную мощность в нагрузке. Хорошо, мощный транзистор не помог. Но есть и другие способы улучшения сигнала на выходе. Я много говорил о пользе отрицательной обратной связи. Что если ввести ее в наш усилитель? Добавим в эмиттерную цепь мощного транзистора резистор 1 Ом.

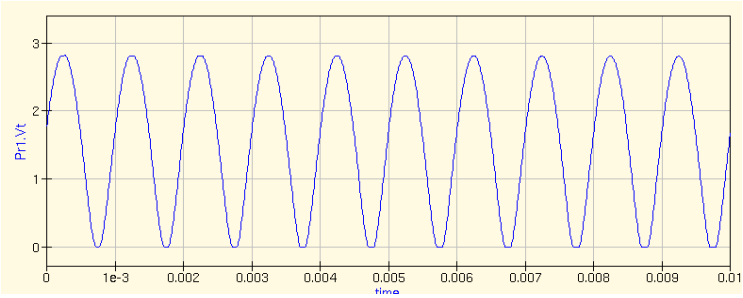


Рис. 6.4. Введение отрицательной обратной связи

Хотя искажения заметны «на глаз», но это можно подправить, а сигнал на выходе вырос почти до 1.5 В. Отрицательная обратная связь помогла. Для верности я даже включил вольтметр параллельно резистору нагрузки. Но это, все-таки, не совсем то, что хотелось бы.

Мощность. Мощность – это напряжение, умноженное на ток. Я даже не хочу измерять ток, поскольку ток будет равен напряжению, деленному на сопротивление. В итоге: $P=U^2/R$. Здесь напряжение – действующее значение, которое, примерно, в полтора раза меньше амплитудного, равного 1.5 В. И мощность получается около 200 мВт. Это даже не 5 Вт. Далеко не 5 Вт!

Часть мощности, насколько я понимаю, рассеивается на резисторе обратной связи, которое я включил в цепь эмиттера транзистора, и обогревает окружающую среду. Но обратная связь улучшила параметры усилителя, отказываться от нее не хочется. А что, если вместо резистора обратной связи включить в эмиттерную цепь транзистора сам громкоговоритель. По идее действие отрицательной обратной связи должно еще больше сказываться на параметрах усилителя в сторону их улучшения, а мощность будет расходоваться на получение звука, а не на обогрев. Стоит попробовать.

Предварительно хочу отметить, что громкоговоритель, присутствующий во всех схемах, я рассматриваю как активное сопротивление, величину которого можно измерить мультиметром, включенным в режим измерения сопротивления. Почему я так поступаю, я попробую объяснить несколько позже, когда, и если, удастся справиться с задачей получения подходящей мощности усилителя.

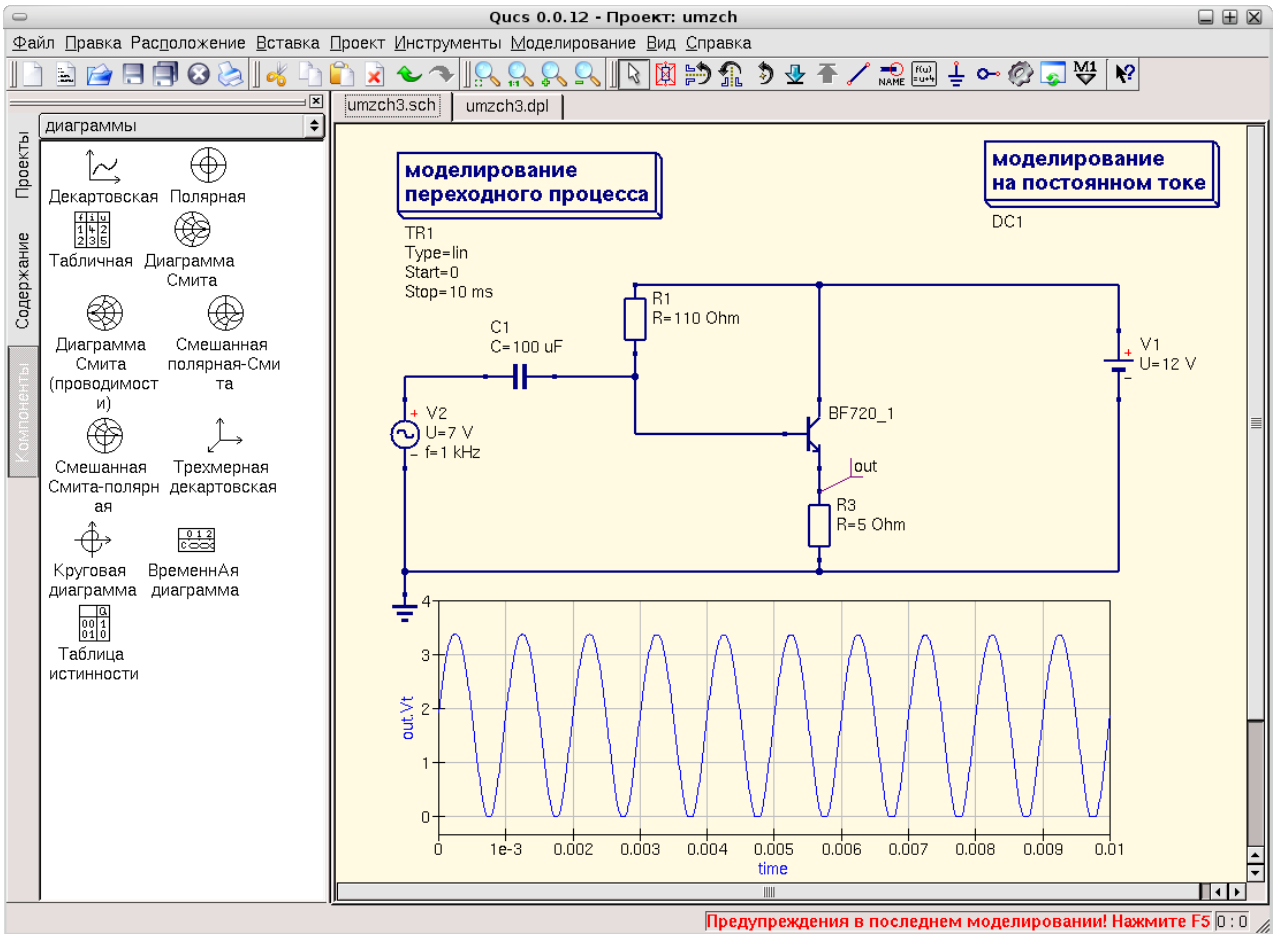


Рис. 6.5. Включение нагрузки в цепь эмиттера выходного транзистора

Все. Моя фантазия иссякла. Что придумать еще я не знаю. Кстати, подобная схема включения транзистора называется включением транзистора с общим коллектором. Дело в том, что сопротивление источника питания сигналу от генератора V2 очень мало. Можно считать, что по отношению к сигналу общий провод, к которому подключен второй вывод генератора, эквивалентен плюсу источника питания, как если бы генератор был вторым концом подключен к коллектору транзистора.

Выходное сопротивление каскада с общим коллектором понижается, входное сопротивление увеличивается (по сравнению с включением транзистора с общим эмиттером), усиление по напряжению меньше единицы. Вполне оправдано было мое решение применить эту схему включения, поскольку нагрузка транзистора низкоомная, а для согласования выходного сопротивления усилителя и нагрузки выходное сопротивление следовало уменьшить.

Есть еще одна неприятность, которую я старательно «замалчиваю». Я даже не стал измерять ток, а рассчитал его при определении мощности. Именно ток меня и смущает. Есть сигнал, нет сигнала, ток через громкоговоритель будет протекать. А проводник с током, помещенный в магнитное поле, будет двигаться, пока его не уравновесят другие силы. В применении к громкоговорителю эта неприятность выглядит так: диффузор громкоговорителя, соединенный с катушкой в поле постоянного магнита, будет втянут (или выдвинется) насколько хватит возможностей его подвески. И больше он перемещаться в эту сторону почти не сможет, не сможет повторять синусоидальное изменение сигнала, то есть, мы получим нелинейные искажения, но уже по звуковому давлению. Есть решение этой

проблемы, как отделить постоянный ток мы знаем – поставить конденсатор. Это так. Но нам придется оставить резистор нагрузки транзистора и к нему через конденсатор подключить громкоговоритель. В этом случае для сигнала, а именно он нас интересует, сопротивление нагрузки выходного каскада уменьшится вдовое (параллельно включены резистор нагрузки и громкоговоритель по 5 Ом). Можно пытаться увеличивать сопротивление нагрузки, чтобы как-то поправить положение, но, боюсь, все, чего мы достигли, может оказаться «за бортом». Вот о такой неприятности я старался не думать.

Может быть зря старался? В конце-концов, я могу и ошибаться в своих рассуждениях. Я вообще могу все понимать неверно. Есть единственный судья, который может решить все сомнения – эксперимент. Добавим к схеме еще один резистор в качестве громкоговорителя и конденсатор большой емкости. Это не долго.

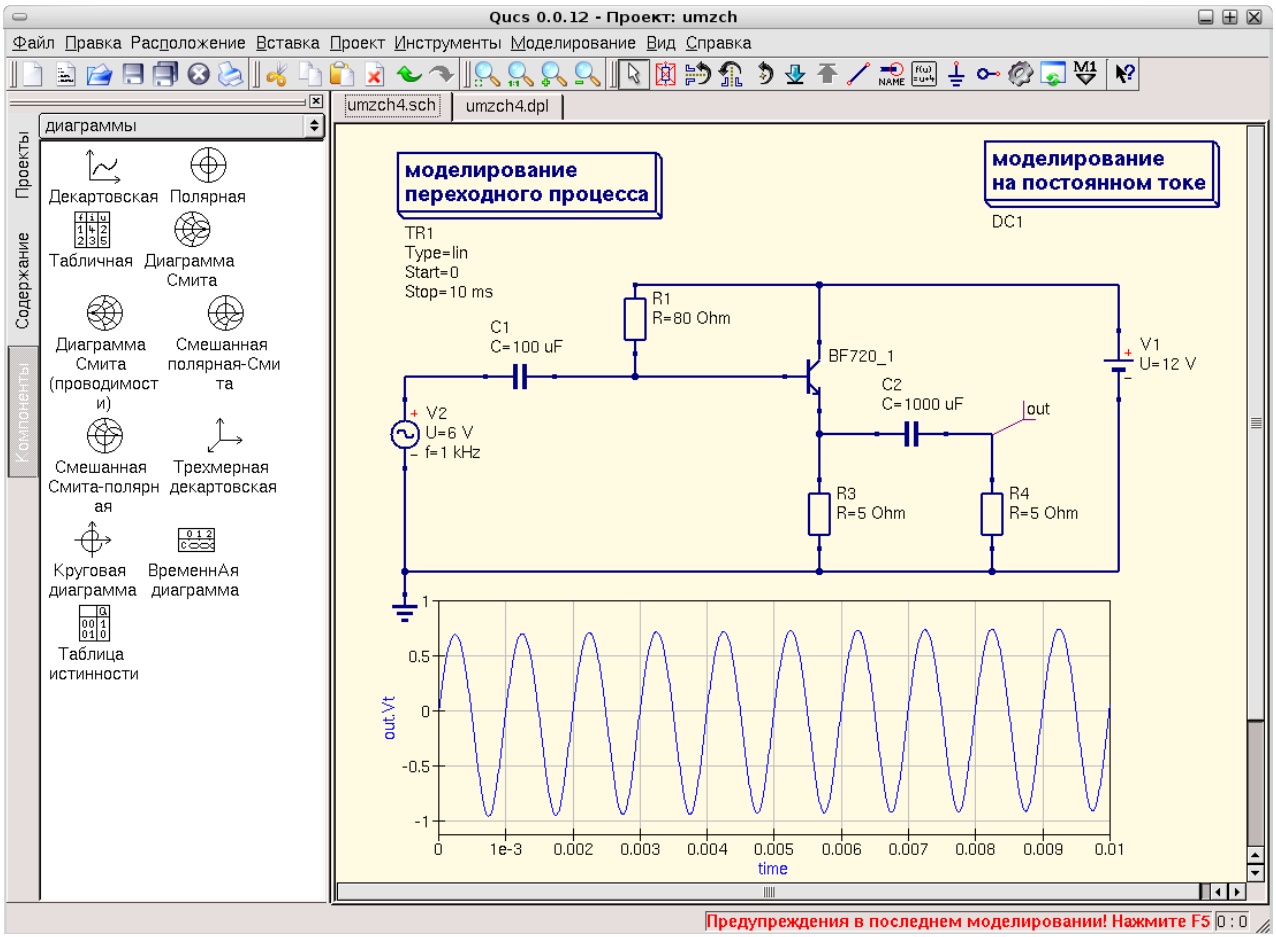


Рис. 6.6. Устранение влияния постоянного тока на громкоговоритель

Что ж, может быть я неправильно все понимаю, может быть я ошибался, но большего мне получить не удалось. Увы!

А, впрочем, может и не напрасно я проводил последний эксперимент. Он натолкнул меня на мысль попробовать заменить пассивный обогреватель воздуха в лице резистора R3 на что-то более полезное. Идея такова: транзистор и обогреватель воздуха R3 образуют, в сущности, делитель напряжения источника питания. При этом транзистор можно рассматривать как резистор, меняющий свое сопротивление в соответствии с законом изменения напряжения сигнала. А резистор R3, хотя он и активное сопротивление, но по отношению к сигналу ведет себя абсолютно пассивно. Что если заставить и его меняться по закону сигнала! То есть, если

вместо резистора R3 включить еще один транзистор? Отчего бы не попробовать?!

Схему включения транзистора я менять не буду, оставлю включение с общим коллектором. А транзистор для симметрии возьму другого типа, p-n-p. В итоге должно получиться что-нибудь похожее на следующее:

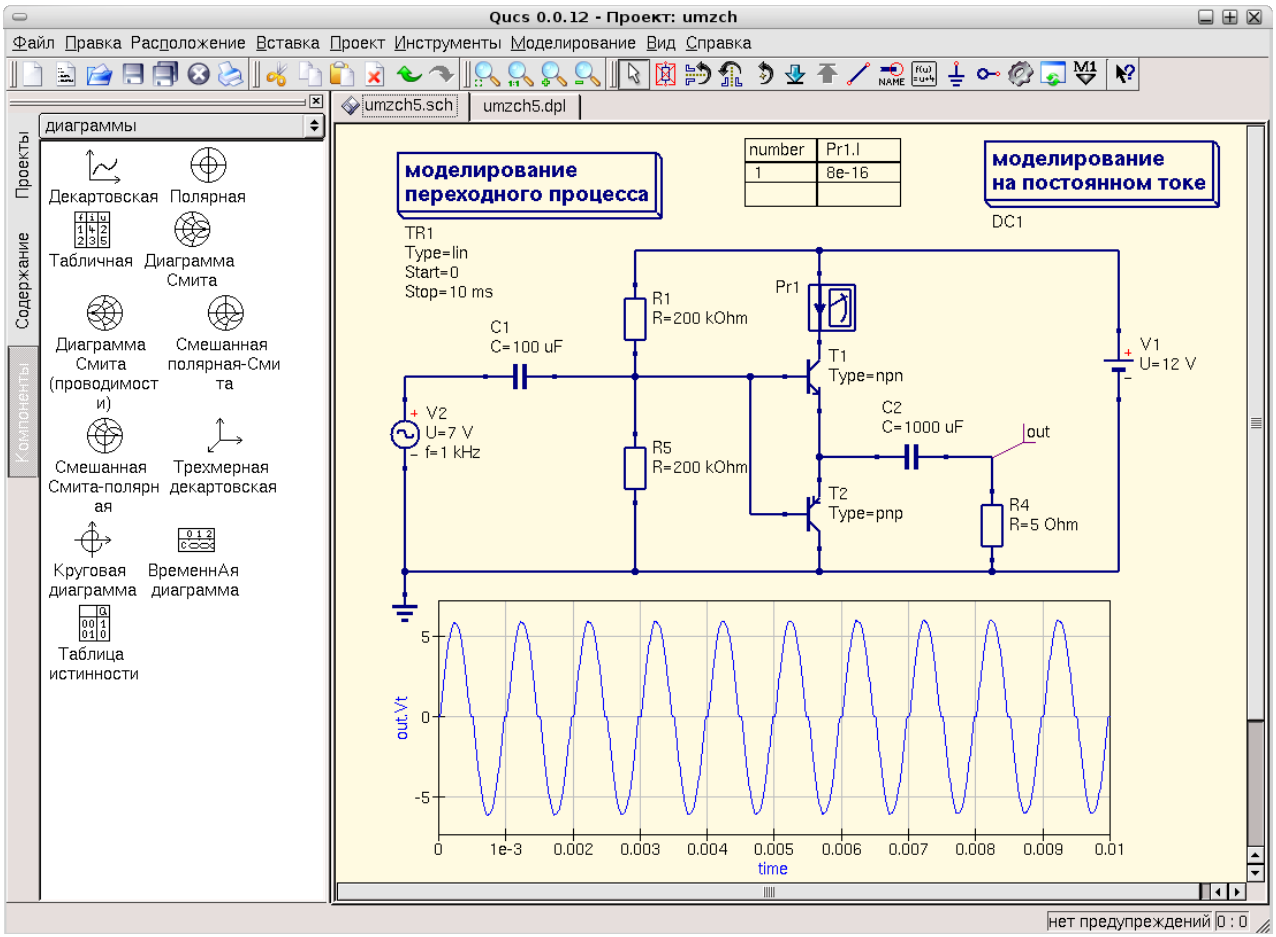


Рис. 6.7. Схема усилителя мощности с транзисторами разного типа

Вот теперь можно считать мощность. Есть что посчитать. И, конечно, я не придумал эту схему, я ее знал еще со студенческих лет, а может и раньше. Когда-то такое включение двух активных элементов в выходном каскаде называли схемой «пуш-пул», тяни-толкай (в прямом переводе толкай-тяни, но мне запомнилось, как я написал). Если не обращать внимание на небольшие искажения вершин, можно несколько уменьшить входной сигнал, то интерес в данном случае следует проявить к току через транзисторы в отсутствии сигнала. Включите амперметр в коллектор транзистора T1 или T2, хотя бы ради любопытства, попробуйте менять величину резисторов R1R5, результаты весьма впечатляют. Наши транзисторы могут в отсутствии сигнала потреблять очень маленький ток. Значит, они не будут греться на «холостом ходу», и не будут потреблять лишний ток, например, от батарейки!

Обычно выходной каскад настраивают с помощью делителя R1R5 так, чтобы в отсутствии сигнала напряжение на эмиттерах транзисторов было равно половине питающего напряжения. Амплитуда выходного сигнала не может превысить это значение – транзисторы не генерируют питающее напряжение, они образуют управляемый сигналом делитель напряжения питания от источника V1.

Есть еще одна особенность, на которую следует обратить внимание. На отметке нуля вольт

выходной сигнал, пересекая эту нулевую границу, имеет характерное искажение очень точно называемое «ступенькой». Это искажение сигнала обусловлено тем, что оба транзистора в отсутствии сигнала (сигнал имеет напряжение равное нулю) не имеют смещения на базе. Чтобы устранить ступеньку в реальных схемах используют два решения. Первое, там где качество воспроизведения звука не очень актуально, применяют глубокую отрицательную обратную связь всего усилителя мощности. Второе решение – задание некоторого начального смещения (напряжения база-эмиттера) за счет введения дополнительных элементов в схему.

Я пока не буду этого делать. Хотя результат весьма неплох, но изначально вопрос звучал иначе. Напомню, задача получить мощность более 5 Вт при напряжении питания 12 В. Пока мощность, которая может быть получена от усилителя даже меньше 5 Вт. Есть по меньшей мере два решения, которые я знаю. Их может быть и больше, но и двух, я думаю пока хватит. Первое решение – это использовать преобразователь (конвертер) для превращения 12 В в, например, 24 В. Но о конвертере я хочу поговорить позже, это связано и с программой PSIM, а сейчас только «нарисую» второе решение, которое проще в выполнении.

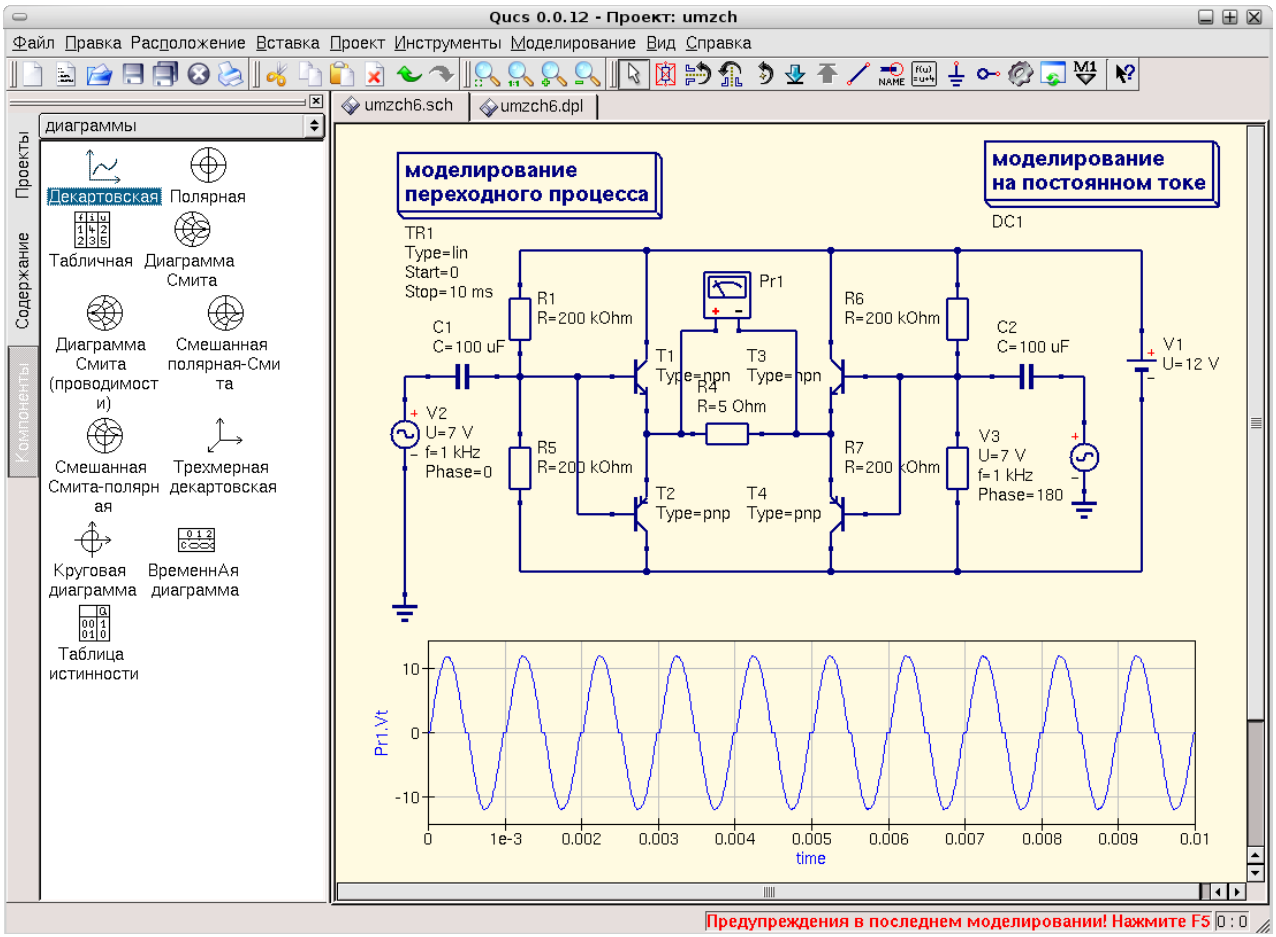


Рис. 6.8. Мостовое включение нагрузки в УМЗЧ

Амплитуда сигнала, как видно из диаграммы, увеличилась почти до 12 В. Можно предположить, что действующее напряжение на нагрузке имеет порядок 8 В. А это соответствует мощности около 12 Вт. Один ответ, похоже, найден. Я использовал два источника сигнала, один из которых V3 имеет начальную фазу 180 градусов, то есть, включен противофазно первому V2. В реальной схеме, конечно, один источник сигнала, а противофазное включение достигается использованием соответствующих входов УМЗЧ, прямого и инверсного, или сигнал инвертируется для получения нужного эффекта.

За счет чего удалось увеличить амплитуду сигнала в нагрузке? Яснее всего это было бы видно, если бы на схеме рисунка 6.7 мы разделили источник питания на два по 6 В, удалили бы конденсатор С2, а громкоговоритель подключили бы одним концом к эмиттерам транзисторов, а вторым концом к точке соединения источников питания по 6 В. При поочередном полном включении транзисторов наша нагрузка поочередно оказывалась бы подключенной то к одному источнику 6 В, то к другому. Это и давало бы максимально возможную амплитуду сигнала, равную половине питающего напряжения. При мостовом включении попеременное полное включение имеют пары транзисторов: Т1 и Т4, Т2 и Т3. Каждый раз при этом нагрузка оказывается подключена полностью к источнику питания. Возможная амплитуда увеличивается до 12 В.

На практике достаточно хороший результат можно получить, используя микросхемы усилителей мощности серии TDA или отечественные. Две микросхемы позволят реализовать мостовое включение и получить большую мощность, чем одна. Что и требовалось доказать! Усилители интересный предмет для разговора, и я хочу отвести этому следующую главу. Там же немного поговорить об усилении звука в целом. А пока пора вернуться к теме первого проекта. И это будет...

Светофор

Если у вас есть младший брат или сын – владелец гаража с несколькими полу-разломанными, но любимыми машинками, ему доставит удовольствие поиграть, день-два (или час-другой), в регулировщика. И даже если этого не произойдет, то вам доставит удовольствие использовать этот проект для модификации в другой, например, для управления новогодними гирляндами. Задачи очень схожи, а, решая их, можно получить «море удовольствия», поскольку решений очень много, и есть в отдельных решениях такие подводные камни, что любой любитель «экстрима» вам позавидует.

Светофор имеет три сигнальные лампы: зеленую, желтую, красную. Мы возьмем три светодиода: зеленый, желтый, красный. Светодиод – разновидность обычного диода, p-n переход которого при протекании через диод тока излучает свет. Собственно, если я не ошибаюсь, то этим занимаются электроны, которые при торможении или при переходе с одного уровня в атоме на другой излучают избыток энергии в виде фотонов (электромагнитную волну в спектре света). Почему они так делают, я не знаю. Уж такова, видимо, их природа, или так им приходится жить, всяк живет, как может.

Многие светодиоды при проверке их с помощью мультиметра, который следует переключить в режим проверки диодов (в режиме измерения сопротивления современные мультиметры могут иметь напряжение на щупах порядка 0.5 В, а это мало для того, чтобы открыть кремниевый, например, диод даже в прямом направлении), светятся. Но не все. Ток через светодиод не может превышать тот, что указан в его паспортных данных. Для многих типов светодиодов этот ток порядка 15-30 мА. Чтобы ограничить ток через диод, последовательно с ним включают резистор. Величину резистора можно подсчитать, зная рабочий ток светодиода, для определенности возьмем 10 мА, и напряжение источника питания. Падение напряжения на светодиоде составляет 1.5-2 вольта, остаток напряжения источника питания будет падать на добавочном резисторе. Разделив его на рабочий ток, великий закон Ома!, мы получим величину добавочного сопротивления.

Схему самого простого светофора я даже не буду рисовать. Три светодиода с тремя добавочными резисторами соединяем катодами (к анодам у нас подпаяны резисторы) и подключаем к батарейке. Три оставшихся конца (резисторов) подпаиваем к переключателю, который в данном случае будет называться переключателем (например, галетным) на четыре положения и одно направление. Токосъемный (подвижной, общий) вывод переключателя мы

подпаиваем к батарейке. Почему четыре положения? В четвертом положении светофор отключается. Все. Щелкаем переключателем, зажигаем огни светофора. Проще не бывает.

Но это не интересно. Хорошо бы, чтобы светофор включал зеленый свет, горел несколько секунд, затем включал бы на секунду желтый, и следом красный на несколько секунд. А потом через желтый свет переходил бы к зеленому. Как реализовать такой светофор с помощью микроконтроллера, я знаю. Но микроконтроллеры мы «еще не проходили». А как еще можно? Не знаю. Нужно подумать.

Есть одно решение. Когда я рассказывал о триггерах, то упоминал D-триггер (рисунок 5.29). Если взять три таких триггера, подключить к ним наши светодиоды с добавочным резисторами, то, переключая эти триггеры, можно получить искомое. Я даже знаю, что можно использовать в качестве источника данных для подачи этих сигналов на входы данных триггеров. Есть такая микросхема, которая называется ПЗУ. Нужные нам данные мы можем запрограммировать в этой микросхеме, добавить ко всему тактовый генератор... но ПЗУ мы тоже «не проходили». Подумаем еще раз.

Поскольку сразу решение я не нахожу, попробую разбить задачу на несколько подзадач. Первая подзадача выглядит так: есть три светодиода с добавочными сопротивлениями, их нужно последовательно переключать. Попробуем решить такую, более простую задачу.

Что-то включать и выключать можно с помощью реле. Используем это.

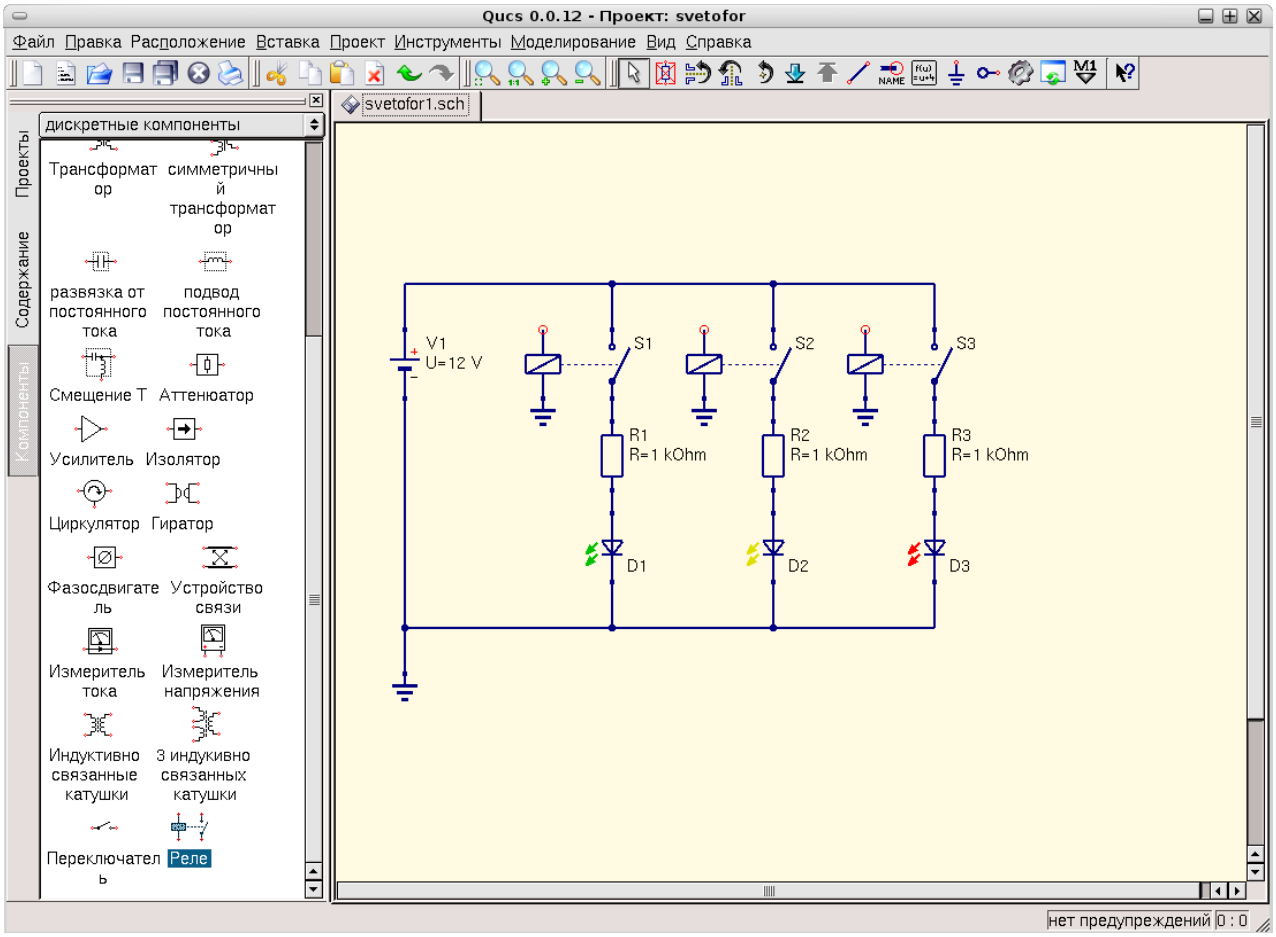


Рис. 6.9. Первая схема проекта «Светофор»

Но в таком виде обмотки реле нужно подключать к переключателю, а это значит «плодить» лишние сущности – реле явно ничего нового не делает. Такой светофор нам не

нужен. В программе Qucs светодиоды можно найти в разделе «Библиотека компонентов», а реле хотя бы с двумя группами контактов я не нахожу. Поскольку то, как включаются светодиоды уже ясно, можно удалить их из схемы, когда они понадобятся, мы знаем, как их включить, а с реле... Можно на время отказаться от программной проверки схемы и выполнить монтаж на макетной плате. Однако реле мне придется покупать, а реле достаточно дорогостоящий компонент — технология его изготовления требует множества вспомогательных процессов, и, мне кажется, много ручной работы. Реле следует применять там, где без него не обойтись. С другой стороны, реле достаточно интересный компонент электрических схем, о котором хотелось бы рассказать побольше. В первую очередь оно имеет контакты, которые не соединены ни с каким источником тока и напряжения, контакты, которые могут коммутировать и большие токи, и достаточно большие напряжения. При замыкании контактов их сопротивление (переходное сопротивление) очень мало, а при размыкании – ОЧЕНЬ велико. Реле, как и другие компоненты электрических схем, существуют разные, и по конструкции, и по принципу работы, и по параметрам.

Я, пожалуй, немного расскажу о параметрах реле постоянного тока, и постараюсь в программе Qucs показать, как бы можно было подступиться к решению задачи проекта «Светофор», а для самого проекта, все-таки поищу более дешевое решение.

Один из основных параметров реле – рабочее напряжение, то есть, такое напряжение, при подаче которого реле «срабатывает» и может находится во включенном состоянии неограниченно долго. Его контакты, или группы контактов, меняют свое состояние на противоположное, те, что были разомкнуты, замыкаются, а замкнутые, размыкаются. Вместе с тем, любое реле включается при напряжении несколько меньшем, чем рабочее напряжение. Такое напряжение называется напряжением включения. А выключается реле при еще меньшем напряжении, напряжении выключения. Эффект разности между напряжением выключения и включения, отношение тока выключения к току включения, называют *коэффициентом возврата*. Дополнительно замечу, что после подачи напряжения на обмотку реле, контакты переключаются не сразу, а спустя некоторое время – время срабатывания реле. Обмотка реле имеет определенное сопротивление, по величине которого можно определить токи срабатывания, выключения и рабочий. Я думаю, что этих параметров нам вполне хватит для понимания происходящего.

Итак. Реле в программе Qucs есть, но это, скорее, абстрактная модель, хотя имеющая напряжение отпускания и «гистерезис». Оно имеет одну группу контактов, которую я определил бы как «нормально разомкнутую» по умолчанию. У реле могут быть группы контактов, работающие на размыкание, нормально замкнутые, работающие на замыкание, нормально разомкнутые, и на переключение, когда один из трех контактов в группе общий и с одним из контактов группы образует нормально замкнутую пару, а с другим нормально разомкнутую.

Поскольку полную схему я не планирую проверять в программе, а выбор решения я пока сделал в пользу реле, попробую решить вторую подзадачу. Ее я определю следующим образом: после подачи питающего напряжения на схему включается первое реле, которое нормально разомкнутыми контактами подключает зеленый светодиод с его добавочным резистором к плюсу источника питания; одновременно второй группой нормально разомкнутых контактов это реле подключает второе, которое должно включиться через 1-2 секунды после подачи на него напряжения питания.

Как я хочу реализовать такое решение? Я включу второе реле через дополнительный резистор так, чтобы падение напряжения на нем было равно разности питающего напряжения и напряжения включения второго реле. Это достигается подбором величины дополнительного сопротивления и/или напряжения питания. Зная величину сопротивления

обмотки реле мы можем выполнить расчеты с помощью закона Ома. А вся хитрость в том, что касается «*которое должно включиться через 1-2 секунды*», что я параллельно обмотке реле включу конденсатор большой емкости. Сейчас я покажу, а затем расскажу, что из этого может получиться.

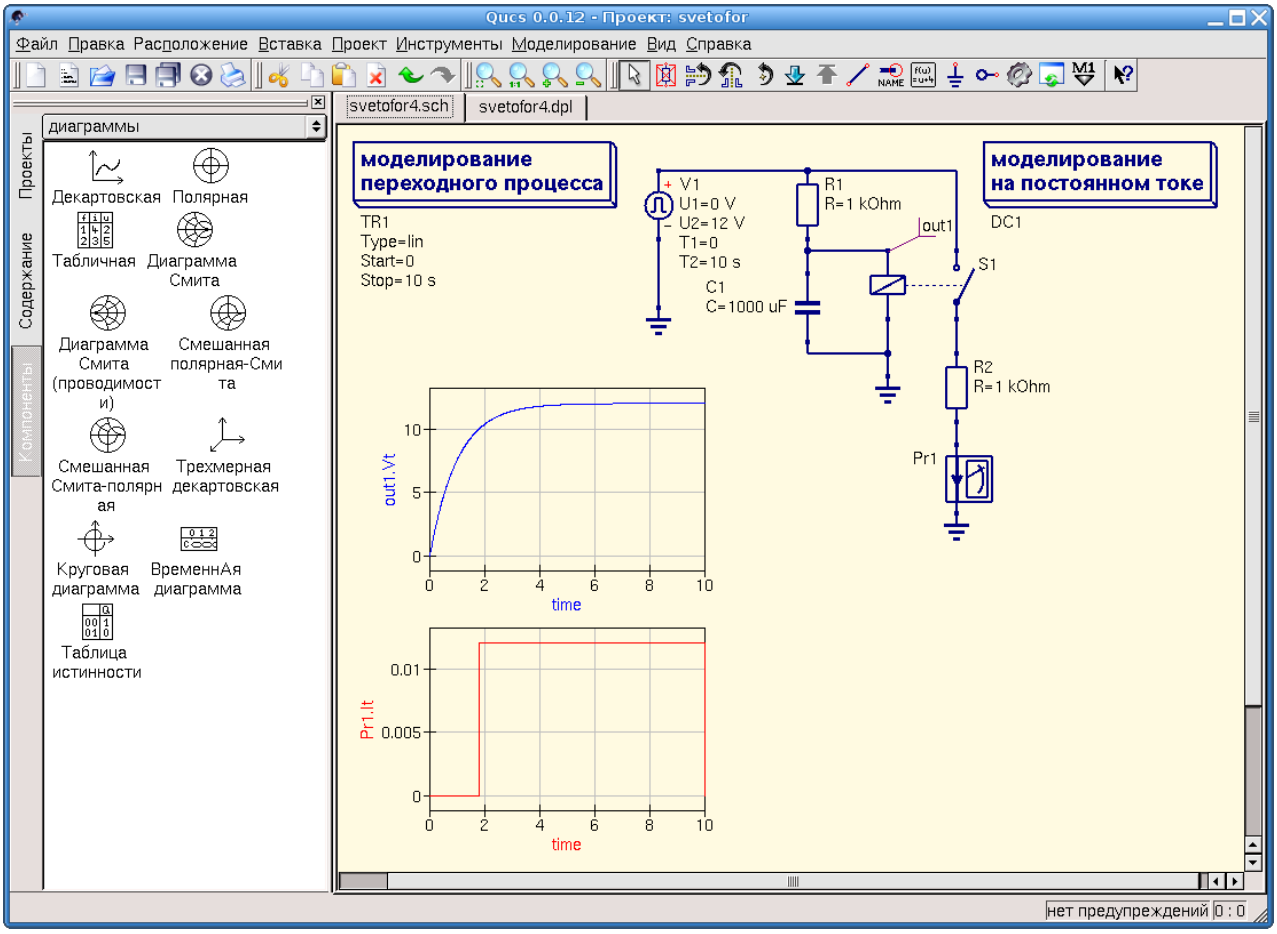


Рис. 6.10. Включение реле постоянного тока с «задержкой».

Чтобы не включать «рубильник», подключающий батарейку к реле, я использовал, я этим часто пользуюсь, источник импульсного напряжения V1. Он в момент времени t = 0 имеет нулевое напряжение, затем его напряжение становится равно 12 В и остается таким в течение 10 секунд (все эти параметры меняются в свойствах источника). Дополнительное сопротивление (последовательно с обмоткой реле) R1. А конденсатор C1, совместно с резистором R1, образуют время-задающую цепь. При появлении импульса величиной 12 В напряжение на конденсаторе равно нулю, как это видно из верхней диаграммы, затем конденсатор заряжается через резистор R1 (по экспоненте, так называется закон изменения напряжения на конденсаторе), и, спустя какое-то время (около 2 секунд на верхней диаграмме), напряжение на нем достигает 10 В. Именно такое напряжение я выбрал в качестве напряжения срабатывания реле в свойствах второго реле S1. При этом напряжении реле включается, своими контактами включая резистор R2. На второй, нижней диаграмме, показан ток, который сразу возрастает до величины, определяемой (по закону Ома) сопротивлением R2 = 1 кОм. Таким образом, реле включится не сразу после включения питающего напряжения, а спустя 2 секунды.

Если это, второе, реле мы подключим к источнику напряжения 12 В второй парой контактов первого реле, включающего своими нормально разомкнутыми контактами зеленый

светодиод, и, если это второе реле включает желтый светодиод, то он загорится через 1-2 секунды после зажигания зеленого. Используем нормально разомкнутую пару контактов второго реле параллельно включающей паре контактов первого реле, а нормально замкнутую пару контактов второго реле для подключения питания первого реле, и тогда второе реле будет подключено (своими замкнувшимися контактами) к источнику питания, а первое реле будет от этого источника отключено контактами второго реле...

Я уже немного запутался в этих контактах, проще будет нарисовать.

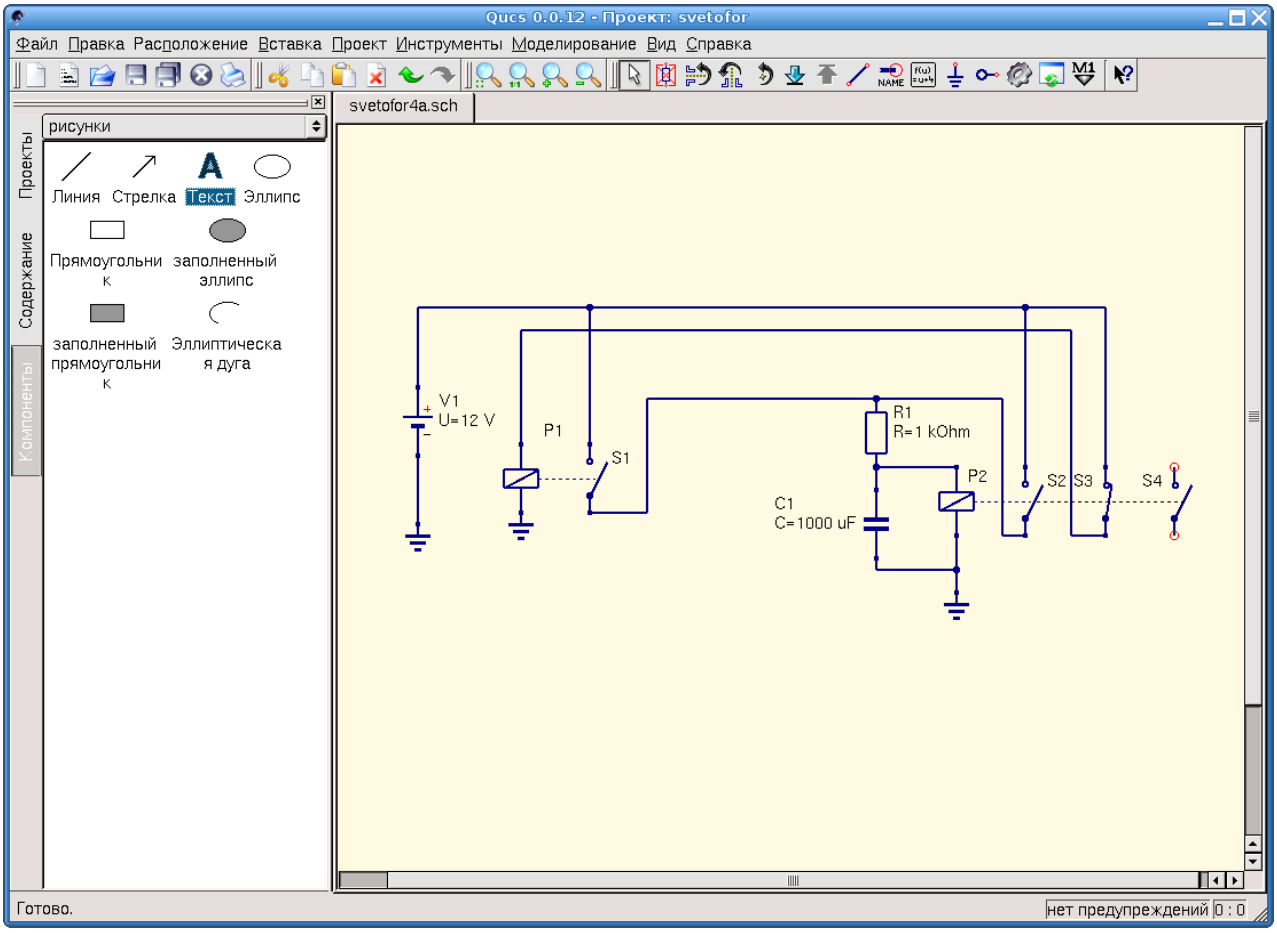


Рис. 6.11. Использование контактов реле для переключения двух реле

На схеме два реле P1 и P2. Первое имеет одну группу нормально разомкнутых контактов S1, а второе имеет три группы контактов S2, S3 и S4. Нормально замкнутые контакты S3 реле P2 подключают первое реле к питающему напряжению. После включения батарейки V1 реле P1 включается и своими контактами S1 подключает реле P2 через время-задающую цепочку R1C1. Когда конденсатор зарядится до напряжения 10 В (через 1-2 секунды), реле P2 включится. Своими контактами S2 оно подключит себя к источнику напряжения и останется включенным. Контактами S3 реле P2 разорвет цепь питания реле P1, и реле выключится, размыкая контакты S1 (но реле P2 остается включенным благодаря своим контактам S2, включенным параллельно). Контактами S3 реле P2 может включить желтый светодиод через 1-2 секунды после включения зеленого светодиода (контактами реле S1, которые не нарисованы), а зеленый светодиод погаснет. Вот таким образом реле последовательно переключатся.

Можно промоделировать временные соотношения этого процесса, поэкспериментировать с изменением емкости конденсаторов, резисторов. Следует только помнить, что реле в

программе, скорее, абстрактное. Его сопротивление обмотки не определено. Как это скажется при замене абстрактного реле конкретным... а, вот, не скажу! Вы вполне можете сами, добавляя к любой из схем для моделирования, приведенных выше или ниже, параллельно обмотке реле включить резистор, который будет изображать сопротивление обмотки. Попробуйте, это интересно. И постарайтесь не забыть, что переключение контактов реле происходит не мгновенно, а требует некоторого, пусть и небольшого, времени. При коммутации возникает ситуация, когда одна пара контактов еще не включена, а другая уже выключена. Если об этом забыть, то схема может оказаться не работоспособна.

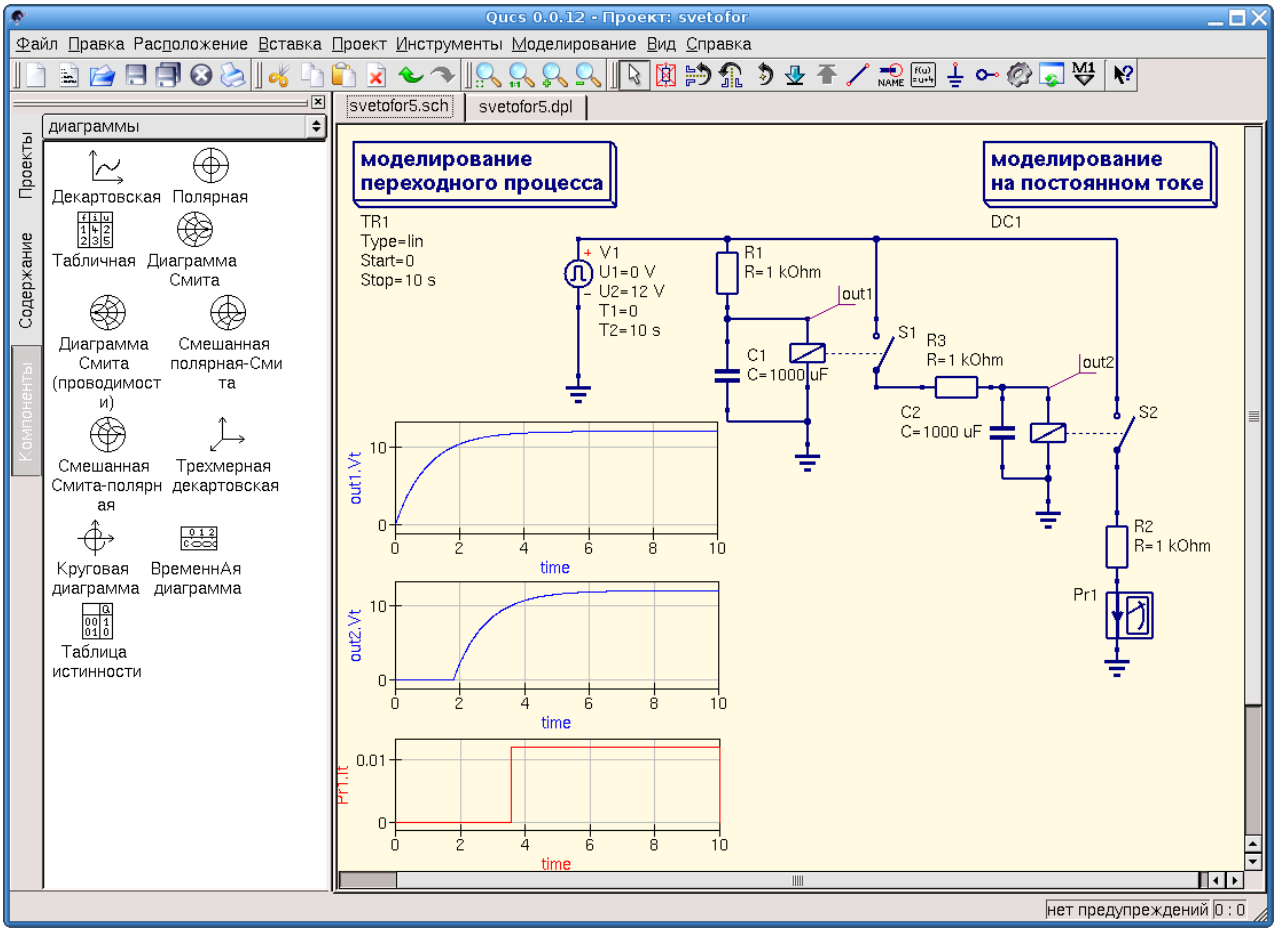


Рис. 6.12. Последовательное включение реле с задержкой

После того, как стало понятно, как можно последовательно включать реле с задержкой, можно продолжить схему, добавляя еще одно реле, которое будет выключать и первое и второе, и подумать, как с помощью дополнительных контактов или дополнительных реле заставить «светофор» переключаться в обратном направлении. Я честно сознаюсь, что не знаю решения. Но не уверен, что его нет. Если у вас нет желания и интереса продолжать построение схемы светофора с помощью реле, то присоединяйтесь ко мне. Я собираюсь поискать более дешевое решение, используя другие компоненты. Цель моего рассказа о поиске решения с помощью реле была в том, чтобы напомнить вам, что реле, выходящее из моды, остается очень хорошим и удобным элементом электрических схем. Иногда применение реле сильно упрощает схему, иногда удешевляет решение, и о реле не следует забывать. Не так давно реле использовалось очень широко, было множество полезных схем, построенных с применением реле, и в частности разновидность реле – шаговое реле – была основой работы всех автоматических телефонных станций.

Кроме того, мне хотелось еще раз напомнить, что, хотя конденсатор не проводит постоянный ток, он довольно часто используется в цепях постоянного тока, благодаря переходному процессу (заряд конденсатора по экспоненциальному закону), совместно с резисторами для формирования временных интервалов. Да и вообще, мне реле нравится.

Но и схему светофора надо придумывать.

Пока я рассказывал о реле, я лихорадочно вспоминал, не приходилось ли мне решать подобную задачу прежде? И раздумывал, как еще можно подступиться к решению, хотя бы в принципе. Ничего не припомнив, ничего не придумав, я, тем не менее, обратил внимание на характер работы светофора. Он, как счетовод, посчитал, перекинул костяшку на счетах, посчитал, перекинул другую. Вначале он складывает, потом вычитает... Что-то в этом роде.

И исходной точкой в задаче становится счет, вернее, отсчет времени или подсчет импульсов тактового генератора. А среди цифровых микросхем есть такая микросхема, которая называется счетчик. Как она работает? Самое простое описание работы счетчика – с приходом каждого импульса от тактового генератора число на выходе микросхемы увеличивается (или уменьшается при обратном счете) на единицу. Число на выходе счетчика задается несколькими выводами, на которых устанавливается либо высокий, либо низкий уровень. Каждый вывод, например, позиция двоичного числа, и по состоянию выходов легко можно записать двоичное число: 1001, если выводов четыре, соответствует десятичному числу 9. Такое позиционирование двоичных чисел еще называют записью в коде 1-2-4-8, хотя правильнее это выглядит в написании 8-4-2-1. В позиции 8 записывается наличие восьмерки, как в примере с числом 1001, а позиции в 1 – наличие единицы. Если есть восемь и один, значит число, несомненно, девять.

Я не знаю из скольких транзисторов сделана микросхема, я не знаю, как эти транзисторы соединяются в электрические цепи в микросхеме, но я знаю, что, подав на ее входы все необходимые сигналы, а это не только сигнал от тактового генератора, я могу считать импульсы тактового генератора. Вот это свойство цифрового счетчика мне кажется подходящим для реализации проекта «Светофор». Может статься, что я ошибаюсь, это покажет время, но я могу попробовать создать схему на основе счетчика.

Прежде, чем сделать первую попытку в реализации схемы, я чуть-чуть расскажу о счетчике. К моему огорчению в программе Qucs нет готового компонента, который называется счетчик. Но я знаю, если ничего не путаю, что работу любой цифровой микросхемы можно описать с помощью математического уравнения. Для сложных микросхем такое уравнение тоже оказывается сложным, но... его можно упростить. Упростить до вида, когда уравнение описывается с помощью достаточно простых функций вида «И», «ИЛИ», «НЕТ», образующих базовый набор функций. А это означает, что любую сложную цифровую микросхему можно построить из более простых. Не буду пугать вас, рисуя подобные схемы, но как это может быть сделано, покажу.

Начнем с простой схемы триггера, который называется RS-триггер. Напомню, что триггер – это устройство, которое может неограниченно долго (пока включено питающее напряжение) находится в одном из двух устойчивых состояний. В RS-триггере есть два входа, обозначенных как R и S. Если на вход S подать активный уровень, обычно, но не всегда, это уровень логической единицы, то выход триггера принимает активное значение. Часто триггер имеет два выхода. Тогда на первом устанавливается высокий уровень, а второй принимает значение низкого уровня (логического нуля). Если же на вход R триггера подать активный уровень, то все происходит наоборот: основной выход триггера «сбрасывается». Обозначение входов триггера происходит от английских слов *Set* и *Reset* (установка и сброс). Входы установки и сброса, или один из таких входов, как правило, имеют многие цифровые микросхемы. В сложных микросхемах бывает необходимость начинать работу с некоторого

определенного состояния. Например, процессор, начиная свою работу, обращается к программе, где записана последовательность всех его дальнейших действий. Если адрес с которого процессор должен начинать считывание программы не определен, то совершенно не определено и поведение процессора, поскольку в памяти могут хранится и команды, и данные. Но и те, и другие – только двоичные числа. Можете себе представить, что будет вытворять ваш компьютер, если начнет с чтения данных, принимая их за команды! Поэтому начальное состояние любой цифровой схемы желательно иметь вполне определенным, устанавливая выходы микросхем в наперед заданное положение. Кстати, я не зря заговорил о процессоре, начиная рассказ о цифровых микросхемах. Почти все популярные цифровые серии микросхем позволяют создать процессор. И когда-то так процессоры и строили. Это теперь появились микропроцессоры, которые прячут в своих глубинах и счетчики, и триггеры и прочую цифровую премудрость. А раньше все было... впрочем, было, да давно прошло.

Итак. RS-триггер.

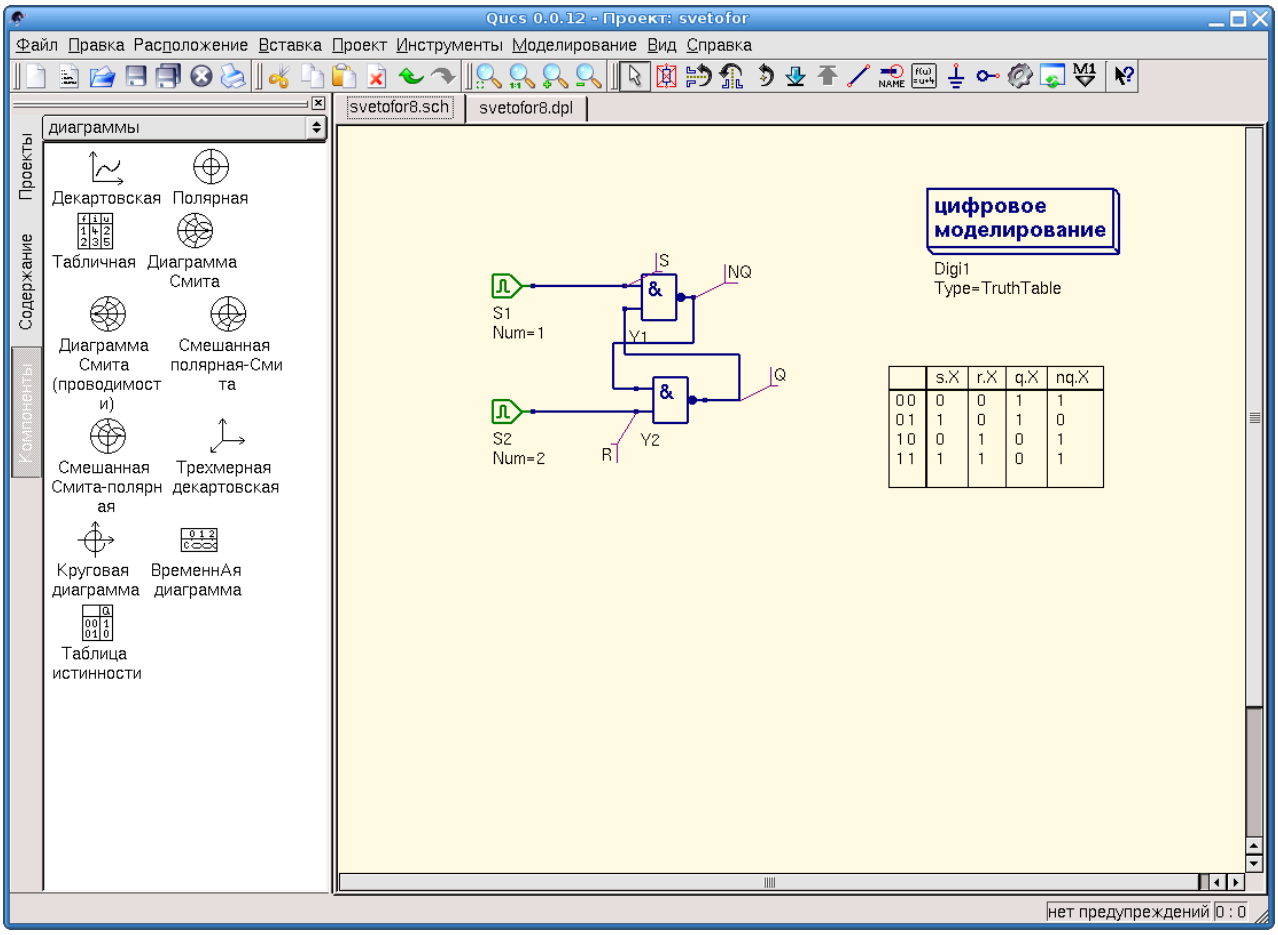


Рис. 6.13. RS-триггер на элементах И-НЕ

В таблице истинности значения входов и выходов приведены в соответствие с описанием выше. Программа Qucs имеет в своем составе SR-триггер, но при моделировании у меня процессор загружается полностью, и программа молча висит на экране. Проще оказалось использовать другой триггер, имеющий SR-триггер составной частью. А понадобилось мне это, чтобы показать таблицу истинности. Вот что получилось в итоге.

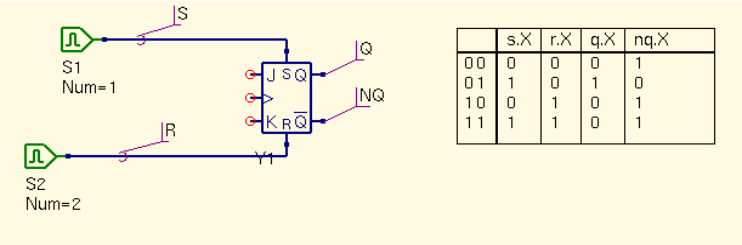


Рис. 6.14. Таблица истинности RS-триггера

Таблицы истинности почти полностью совпадают, то есть, поведение схемы на элементах И-НЕ будет соответствовать поведению SR-триггера. Усложняя схему, можно получить другие виды триггеров, а из триггеров построить другие элементы, в частности счетчики. Очень просто получается счетчик из D-триггеров.

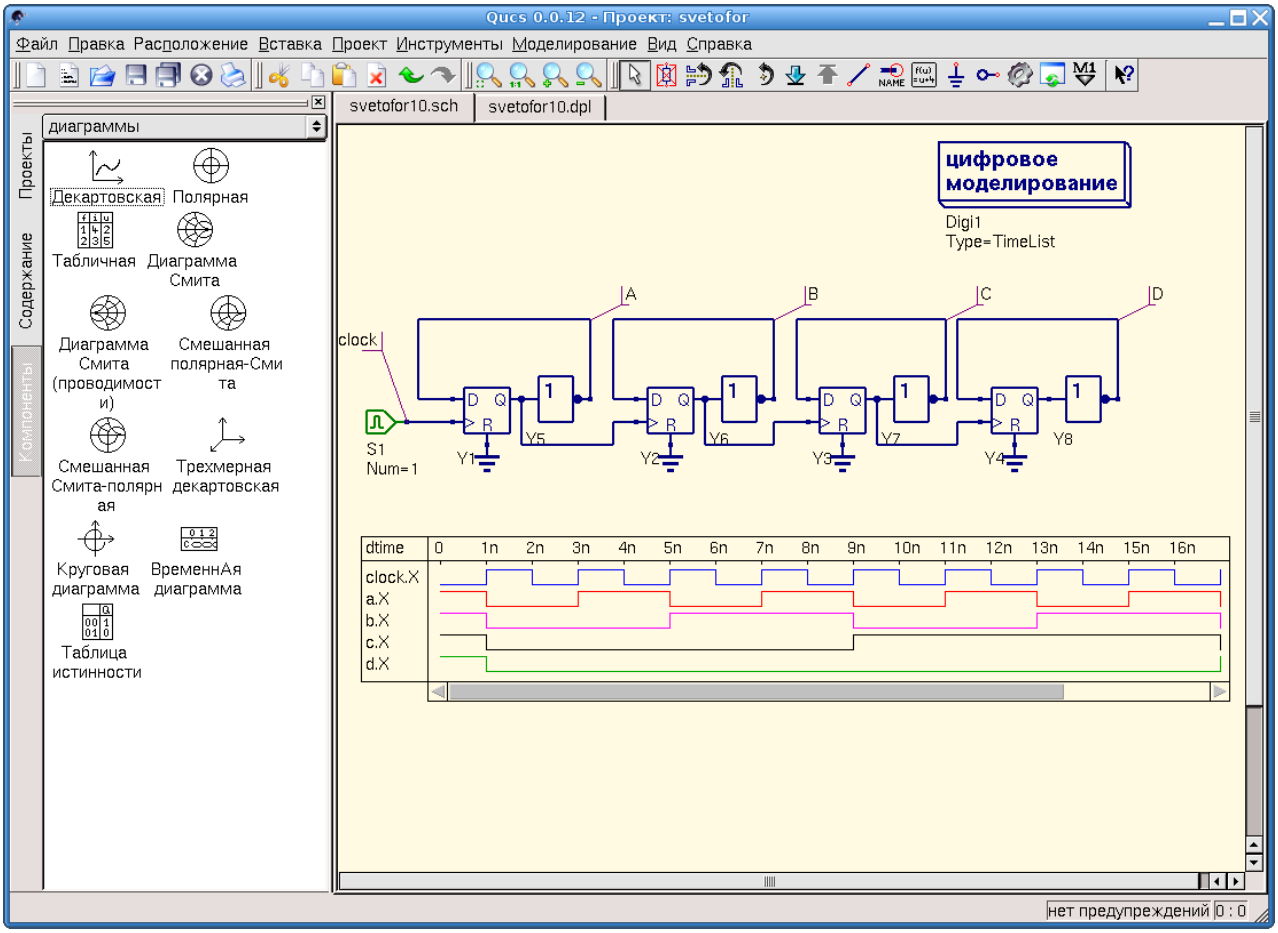


Рис. 6.15. Счетчик на D-триггерах

На рисунке показана не таблица истинности, а временная диаграмма, из которой можно понять еще одно частое применение счетчиков – в качестве делителей частоты. Частота на выходе A счетчика вдвое ниже тактовой (*clock*), на выходе B вдвое ниже, чем на выходе A, и т.д. И можно посмотреть, как считает счетчик: после прихода тактового импульса в момент времени 3n (3 наносекунды) на выходе A устанавливается единица, в момент времени 5n единица на выходе A сбрасывается, но устанавливается на выходе B, а в момент времени 7n устанавливается и на выходе A, и на выходе B, что соответствует количеству полученных импульсов к этому времени равному 3 (двоичное 11).

Это свойство счетчика отсчитывать время, задаваемое тактовым генератором, я хочу использовать для схемной реализации проекта, конечно, используя микросхемы удобные для такого построения. В первую очередь я хочу использовать счетчик, который может считать в прямом и обратном направлении, использовать дешифратор для преобразования двоичного числа на выходе счетчика в более для меня удобное десятичное число. Далее я рассуждаю так: пока счетчик считает от 1 до 4 я с помощью четырехвходовой схемы ИЛИ буду удерживать включенным зеленый светодиод. При счете от 5 до 6 – желтый, а потом красный. Когда счетчик досчитает до 10 мне нужно будет запустить его на обратный отсчет. Для этой цели я использую D-триггер...

Пока это только соображения и фантазии, но их можно проверить.

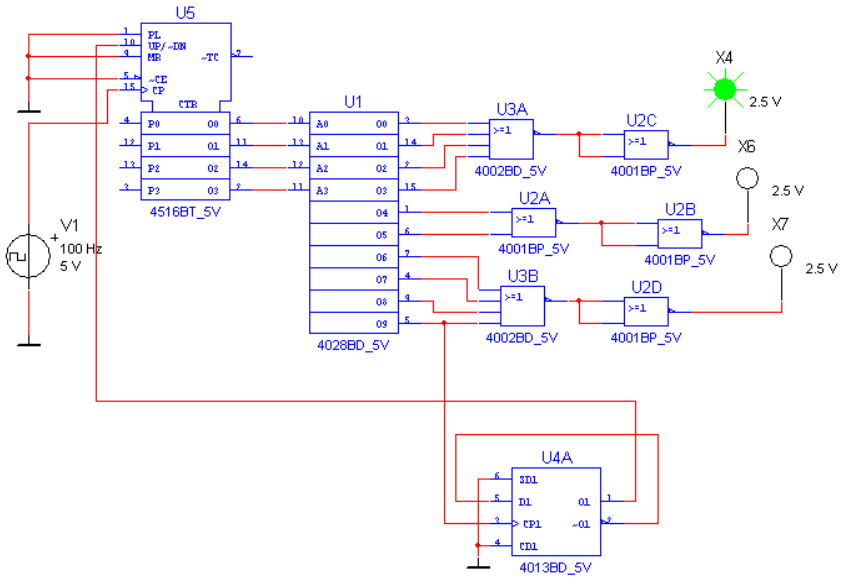


Рис. 6.16. Схема проекта «Светофор» на цифровых микросхемах

Здесь U5 – счетчик с прямым и обратным счетом, U1 – дешифратор, а U4A – D-триггер, который устанавливает на входе счетчика направление счета. К сожалению программа Qucs не имеет подходящих готовых цифровых компонент, и чтобы не пугать вас сложными схемами, я нарисовал схему в другой программе. При таком построении схема, с моей точки зрения, должна работать, но проверять ее работу на макетной плате я не вижу смысла, поскольку разумнее, что и не дает мне покоя, реализовать проект на базе микроконтроллера, но если вам захочется познакомиться с работой цифровых счетчиков, можно использовать любой из серий К155 или К561. В качестве тактового генератора можно использовать типовую схему на элементах «И-НЕ» или инверторах. Работая с частотой 1 Гц такой генератор позволит увидеть, если на четыре вывода счетчика включить четыре светодиода с токоограничительными резисторами, последовательность включения. Но это, если интересно. Если нет, то дождемся главы о микроконтроллерах, где и завершим проект. И пока не забыл, очень полезной, если у вас ее нет, будет книга «Популярные цифровые микросхемы», автор Шило В.Л. К сожалению ее нет в моей библиотеке, но я точно помню, что книга мне очень понравилась.

А сейчас, если вы не против, можно затеять еще один проект. До сих пор я предполагал, что вы подключаете к макетной плате батарейку, или используете программу для проведения экспериментов. Но можно использовать программу, чтобы реализовать проект сетевого блока питания. Питание очень важный фактор для работы любой электронной схемы.

Блок питания

Самый простой блок питания, если говорить о питании от сети, получается из понижающего трансформатора, мостового выпрямителя и электролитического конденсатора. Понижающий трансформатор я советую купить готовый. Дело не в том, что его сложно рассчитывать, есть простые расчетные формулы, с которыми можно справиться, и дело не в том, что его сложно наматывать, было бы желание и терпение, но при самостоятельном изготовлении трансформатора можно легко повредить обмоточный провод и испортить не только результаты своего нелегкого труда, но и вызвать неполадки в силовой сети. Я бы посоветовал даже больше – использовать готовый сетевой адаптер или блок питания без стабилизации. От многих устройств остаются такие неприкаянные сетевые блоки питания. Само устройство давно выброшено или разобрано на детали, а блок питания лежит где-нибудь, заваленный другими не менее забытыми бытовыми приборами. Подобный блок питания можно аккуратно разобрать, хорошо изолировать изоляционной лентой все, что связано с сетью, и провести ряд исследований, которые помогли бы понять работу и устройство обычных блоков питания. Можно бы и не совсем обычных, а таких, как импульсные блоки питания, но исследование их работы в реальных условиях сопряжено с опасностью поражения электрическим током, что, как мне известно, не входит в круг ваших интересов. Как выглядит схема обычного блока питания?

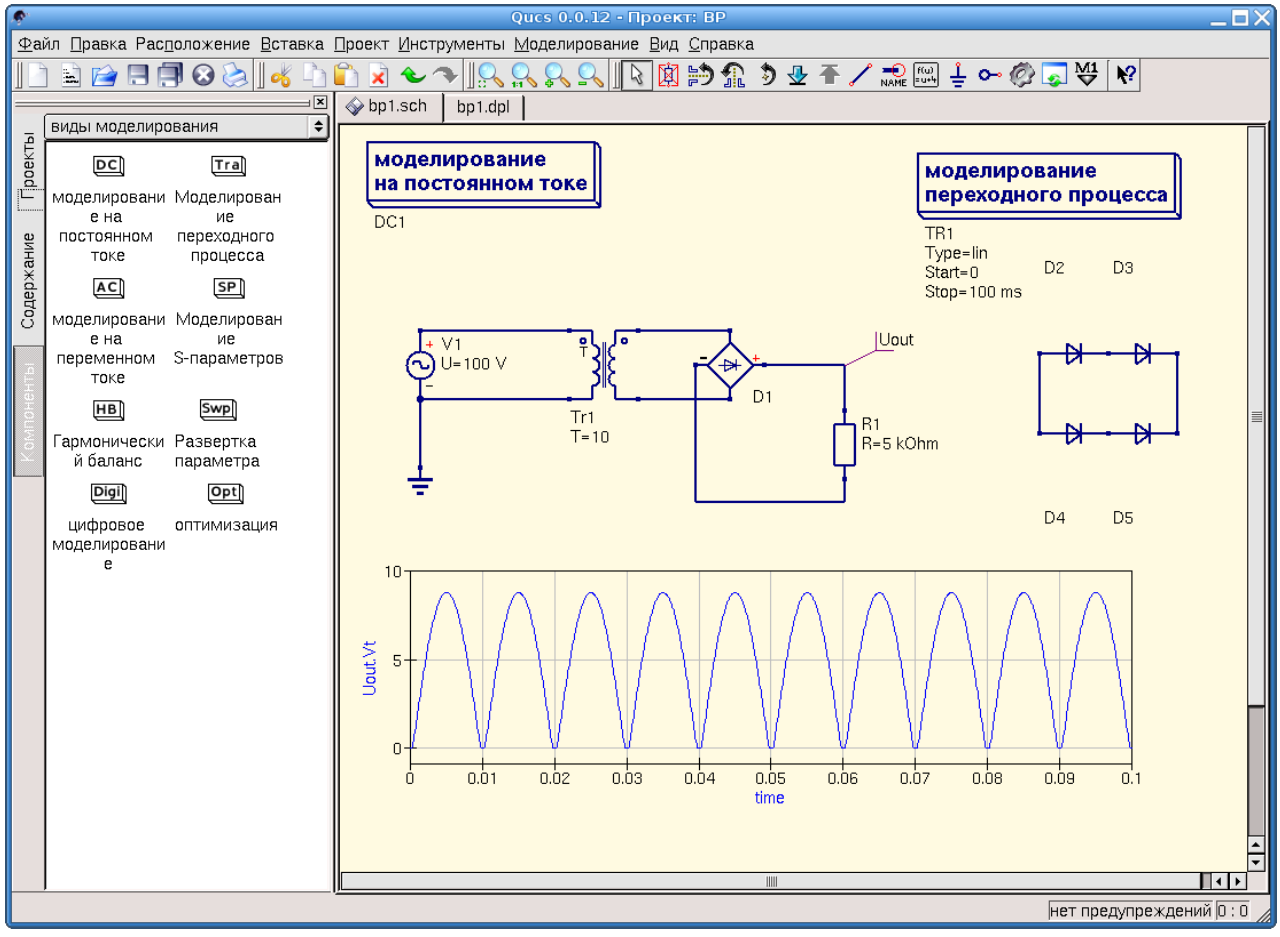


Рис. 6.17. Схема простейшего блока питания

Для простоты я использовал источник V1 с напряжением 100 В, трансформатор с коэффициентом трансформации 10 (такой коэффициент я задаю в программе), готовый диодный мост, который взял из *Библиотеки компонентов* в разделе меню *Инструменты*, но,

подумав нарисовал схему выпрямительного моста рядом, использовав отдельные диоды, и сопротивление нагрузки R1. Конденсатор в схему я пока не добавил, чтобы получить диаграмму выпрямленного напряжения. О трансформаторе я упоминал выше, когда говорил об индуктивности. Могу добавить только, что сердечник трансформатора в данном случае изготавливается из специальных сортов стали, изготавливается из тонких пластин, которые покрывают лаком, и которые затем собирают в пакет. Такой сердечник позволяет максимально избежать потерь. Трансформатор имеет две обмотки, обычно их располагают на одном или двух каркасах из диэлектрика, у понижающего трансформатора первичная имеет много больше витков и выполнена из более тонкого провода, чем вторичная. Идея с витками и проводом достаточно ясна, если вспомнить, что для переменного тока индуктивность представляет собой некоторое сопротивление, зависящее от частоты. Чтобы получить достаточно большое сопротивление на частоте силовой сети 50 Гц, приходится увеличивать индуктивность, которая тем больше, чем больше витков. При этом по первичной обмотке протекает ток, а следовательно мы можем определить мощность в первичной обмотке, умножив этот ток на напряжение. А поскольку почти вся мощность первичной обмотки передается во вторичную, напряжение которой меньше, ток этой обмотки должен быть больше, а, значит, провод должен быть толще. Для нормальной долговременной работы трансформатора толщину обмоточного провода выбирают по допустимой плотности тока в обмотке, то есть, по отношению тока в обмотке к сечению провода. По выбранному сечению и определяется диаметр (толщина) провода. Но это не то, о чем мне хотелось бы сейчас сказать. О том, как переменное напряжение «выпрямляется», я говорил в связи с работой полупроводникового диода. Мостовой выпрямитель, состоящий из четырех диодов, выпрямляет обе полу-волны переменного напряжения, что и показывает диаграмма. Но после выпрямления мы еще не получаем постоянного напряжения. Оно остается переменным. Оно не меняется по направлению, но меняется по величине! Чтобы исправить это и включают на выходе выпрямителя конденсатор.

Конденсатор в сочетании с сопротивлением выпрямителя, состоящим из активного сопротивления вторичной обмотки и сопротивления двух открытых диодов, можно рассматривать как фильтр, срезающий все высокие частоты, представленные в выпрямленном, но «переменном» напряжении. Фильтр «отфильтровывает» все пульсации с частотой 100 Гц и выше, из этих соображений и выбирают величину конденсатора. Почему 100 Гц? Исходно на вторичной обмотке трансформатора переменное напряжение 50 Гц. В течение одного полу-периода на ней плюс там, где обмотка помечена точкой. В этом случае ток проходит через диоды D3 и D4 (рисунок моста на рисунке 6.17). В другой полу-период на вторичной обмотке, помеченной точкой, минус, а ток проходит через диоды D5 и D2. Оба полу-периода равнозначны, значит повторение сигнала на выходе выпрямителя происходит в два раза чаще, чем в сети переменного напряжения, то есть, с частотой 100 Гц.

Действие конденсатора на выпрямленное напряжение можно рассмотреть проще. Конденсатор заряжается через активное сопротивление вторичной обмотки и диодов, а разряжается через сопротивление нагрузки. Время заряда (и разряда) зависит от величины сопротивлений и емкости конденсатора. Если емкость конденсатора выбрать достаточно большой, сопротивление нагрузки достаточно большим, а сопротивление обмотки и диодов всегда достаточно маленькое, то конденсатор будет гораздо быстрее заряжаться, чем разряжаться. Сам же заряженный конденсатор вполне можно рассматривать, как источник напряжения. Без конденсатора выпрямленное напряжение будет «пульсирующим», эти-то пульсации и сглаживает конденсатор.

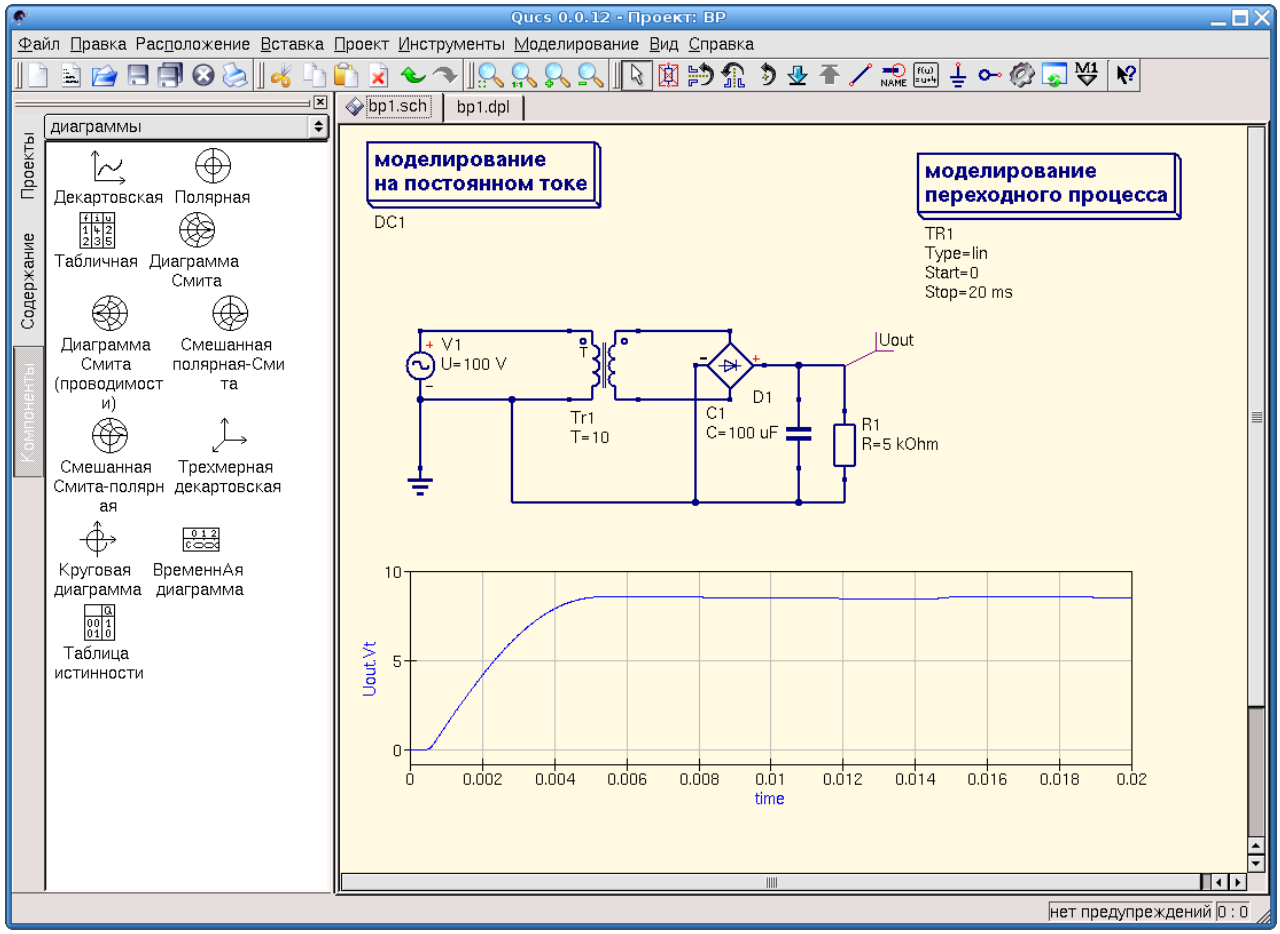


Рис. 6.18. Выпрямленное напряжение с добавленным в схему конденсатором

Такое напряжение уже вполне можно называть постоянным. Величину конденсатора выбирают по току потребления, или иначе можно сказать, что емкость конденсатора зависит от сопротивления нагрузки. Рабочее напряжение конденсатора выбирают равным или большим, чем напряжение на выходе выпрямителя «на холостом ходу», то есть, без сопротивления нагрузки, эквивалентного сопротивлению схемы, для которой мы делаем блок питания. Уменьшим сопротивление нагрузки, что эквивалентно увеличению потребляемого тока, и посмотрим на диаграмму.

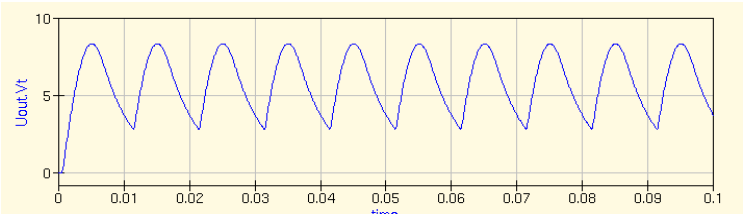


Рис. 6.19. Влияние нагрузки на форму выпрямленного напряжения

Я уменьшил величину сопротивления нагрузки в предыдущей схеме до 50 Ом и увеличил время наблюдения до 100 мС. Напряжение бывшее раньше постоянным вновь становится переменным (по величине) и существенно. Увеличим емкость конденсатора С1 до 1000 мкФ.

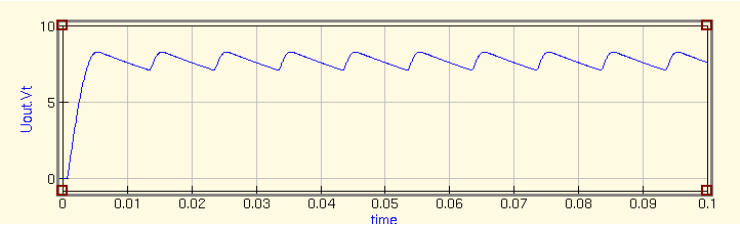


Рис. 6.20. Влияние емкости конденсатора на выходе выпрямителя

Положение стало лучше. Пульсации всегда присутствует на выходе выпрямителя, амплитуда пульсаций входит в расчеты, связанные с созданием блоков питания и не должна превышать некоторой величины, обеспечивающей нормальную работу схемы, для которой блок питания предназначен. Для одних схем допустимы большие пульсации, для других меньшие.

Пока я не убежал слишком далеко в своем рвении рассказать о блоках питания «побольше», хочу обратить ваше внимание еще на один факт, показанный на диаграммах. Если вернуться к самому первому рисунку, то можно отметить, что амплитуда напряжения меньше десяти вольт. Амплитуда генератора 100 вольт. Трансформатор уменьшает ее в десять раз. Почему амплитуда выпрямленного напряжения меньше 10 вольт? Я думаю, что программа учитывает то, что два диода в мостовой схеме выпрямления, а в этом случае для каждой полу-волны в прямом направлении включено два диода, что два диода требуют, чтобы на их p-n переходах было напряжение 0.5-0.7 В. Это напряжение необходимо диоду, чтобы он открывался и пропускал ток в прямом направлении. Реально на диодах падает напряжение даже и несколько больше. Это напряжение, похоже, и учтено программой, это напряжение будет падать и на реальных выпрямительных диодах. Кроме того, после подключения нагрузки, когда в нагрузке протекает ток, на активном сопротивлении вторичной обмотки тоже будет падение напряжения. Это падение напряжения обычно учитывается при расчете трансформатора. Но это напряжение зависит от тока, потребляемого нагрузкой, а, значит, на «холостом ходу» на выходе выпрямителя может быть большее напряжение, чем в рабочем режиме. И это увеличение напряжения следует учитывать при выборе конденсатора, при выборе допустимого напряжения на конденсаторе.

Кроме сглаживающего конденсатора на выходе выпрямителя можно использовать более эффективные фильтры. Но об этом позже. А сейчас немного о том, какое напряжение по величине будет получаться на выходе простейшего блока питания. Или, иначе, что мы можем измерить вольтметром на выходе блока питания?

Рассмотрим схему с конденсатором 100 мкФ и сопротивлением нагрузки 50 кОм. К схеме добавим измеритель напряжения.

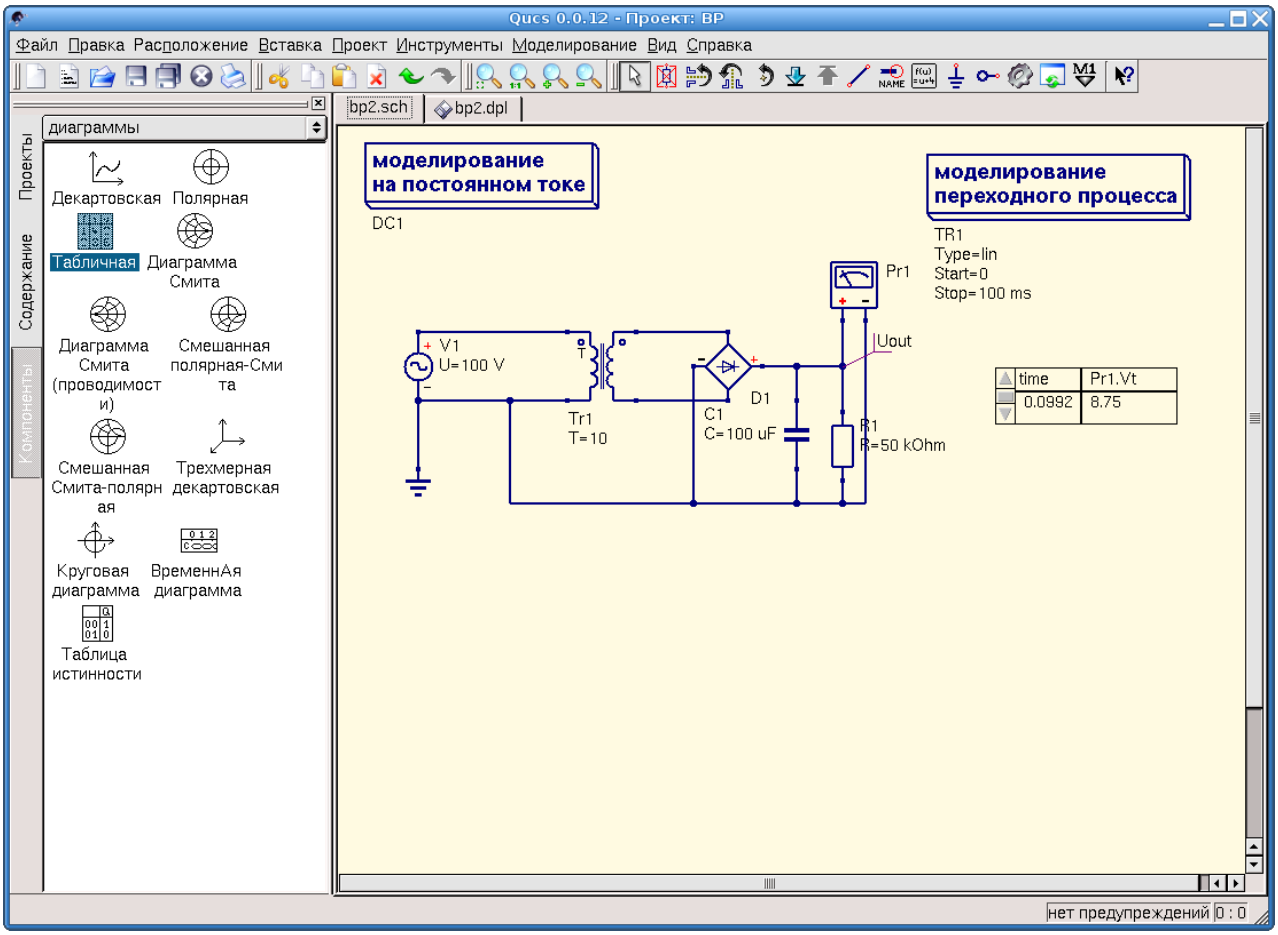


Рис. 6.21. Схема для экспериментов с измерением напряжения

Диаграмму, как вы заметили, я выбрал в табличной форме, а в диалоговом окне выбрал зависимое от времени напряжение измерителя Pr1. В длинном списке напряжений есть те, что получаются через 100 мС после начала измерений. Это напряжение, равное 8.75 В и показано на рисунке. Теперь уменьшим сопротивление нагрузки до 5 кОм. Измеритель покажет напряжение 8.55 В. Уменьшим его до 500 Ом. Напряжение становится меньше. А уменьшив нагрузку до 50 Ом, мы получим 4.4 В. Но если теперь просмотреть все напряжения, записанные в таблице, то будут и меньшие, чем это значение, и большие. А что покажет мультиметр? Некоторое среднее значение между этими напряжениями. Мультиметр в режиме измерения постоянного напряжения показывает среднее значение. Вот на это мне хотелось обратить ваше внимание. Иногда это может приводить к недоразумениям.

Но вернемся к проекту. Когда мы знаем, какое напряжение нам нужно, какова нагрузка – сопротивление нагрузки или потребляемый ток, мы можем подобрать все элементы блока питания надлежащим образом. Но мы-то хотели сделать блок питания, который можно применять во многих случаях, пусть даже при одинаковом напряжении. А напряжение, как выясняется, будет зависеть от нагрузки, и я промолчу о качестве этого напряжения. Как можно улучшить ситуацию?

Конечно, используя стабилизацию напряжения. Рассмотрим два подхода к построению стабилизированных блоков питания, которые могут быть полезны в плане понимания процесса стабилизации напряжения, а затем решим, как сделать блок питания для экспериментов с разными схемами.

Рассказывая о параметрах транзисторов, я говорил, что важно не превышать предельно

допустимого напряжения на коллекторе, иначе ток может резко и неуправляемо возрасти. Но это касается не только транзисторов, а и диодов. Когда обратное напряжение на диоде превышает его предельно допустимое напряжение, ток через закрытый диод может резко возрастать. Диоды, специально выполненные так, чтобы этот эффект стал более «выпуклым», чтобы он был ярче выражен, такие диоды называют стабилитронами (или диодами Зенера). Самый простой стабилизатор – это стабилитрон и резистор, ограничивающий ток через него.

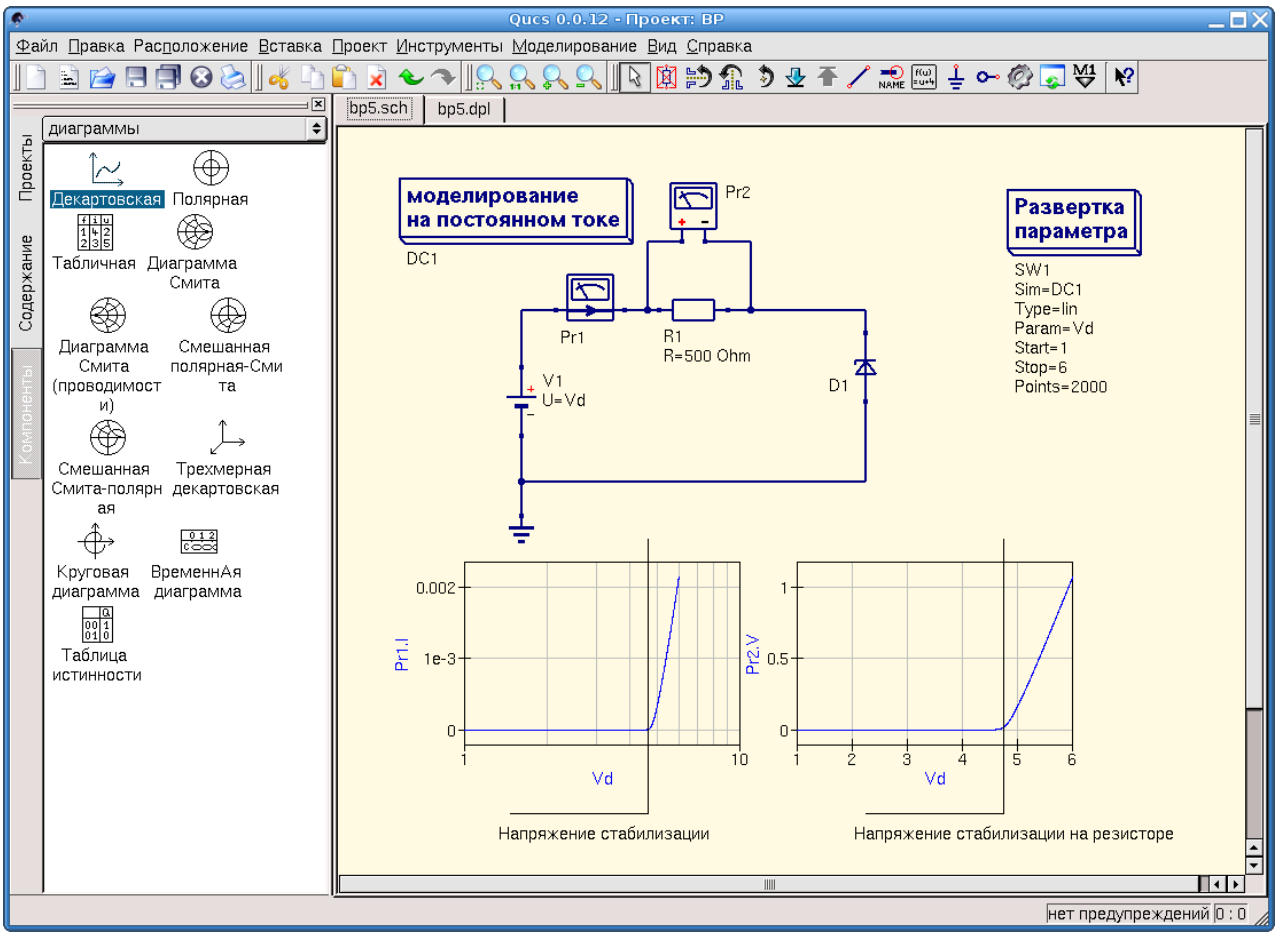


Рис. 6.22. Включение стабилитрона

На первой диаграмме показан ток через стабилитрон при изменении напряжения источника питания V1 от 1 до 6 В. Когда напряжение на стабилитроне достигает величины напряжения стабилизации, ток начинает быстро возрастать. А это приводит к резкому увеличению напряжения на резисторе R1, как это показано на второй диаграмме, в итоге на напряжение на стабилитроне, равное разности напряжений источника и падения напряжения на резисторе, уменьшается и затем поддерживается достаточно постоянным в широком диапазоне изменений напряжения источника питания. Кроме того, если ток в нагрузке, она не показана на схеме, возрастает, то это тоже приводит к увеличению падения напряжения на резисторе R1 и уменьшению напряжения на стабилитроне. Но уменьшение напряжения на стабилитроне приводит к быстрому уменьшению тока через стабилитрон, уменьшению падения напряжения на резисторе R1, то есть, напряжение на стабилитроне и в этом случае будет стремиться к некоторому динамическому равновесию около напряжения стабилизации. Обратите внимание, что пара резистор-стабилитрон, тоже образуют делитель напряжения. Однако напряжение на стабилитроне остается постоянным, а на резисторе R1 будет разница

между напряжением от источника питания и этим постоянным напряжением.

Стабилитрон, с его свойством стабилизировать напряжение, используется в простейшем стабилизированном блоке питания (его называют параметрическим), как это показано ниже.

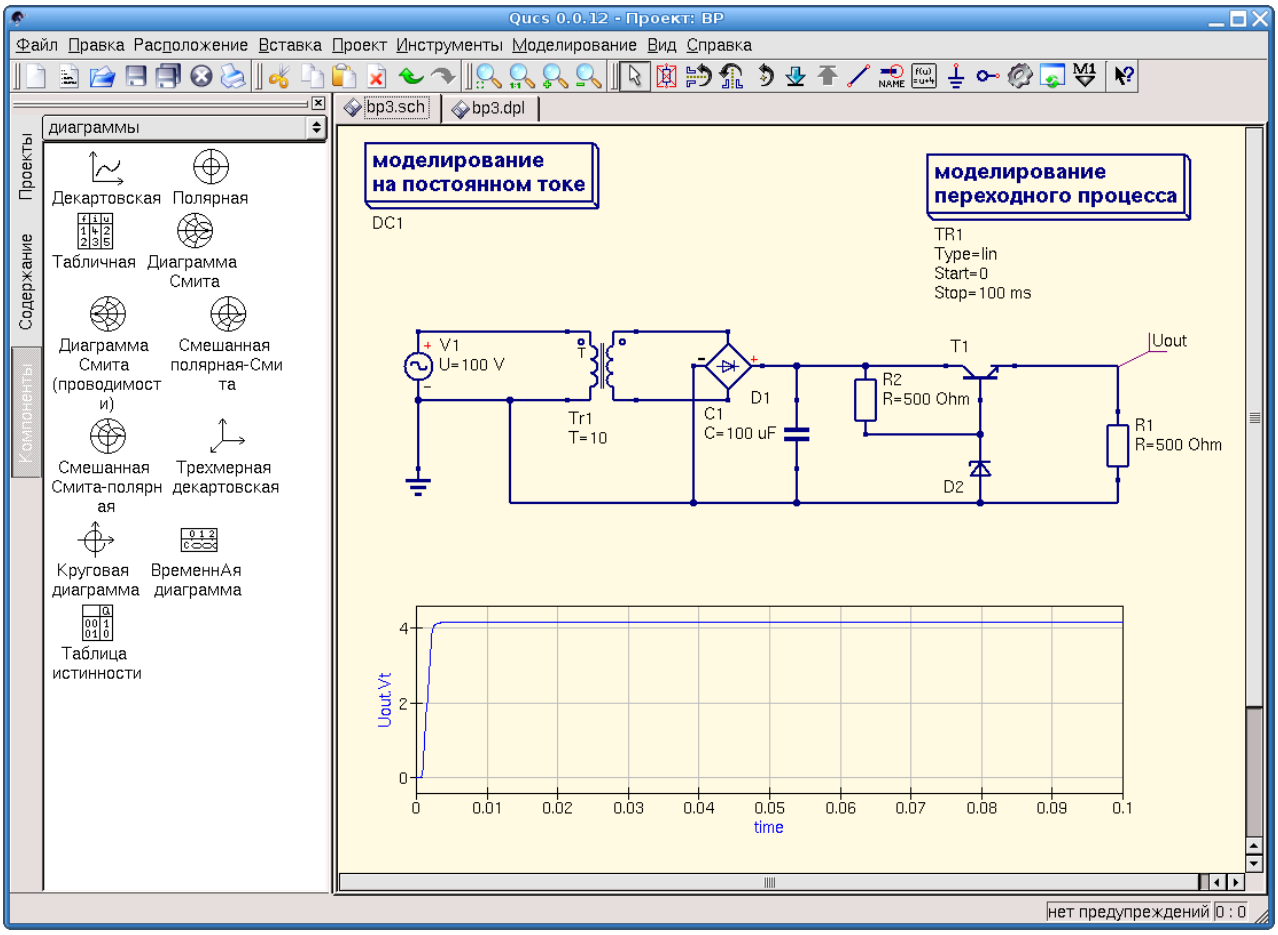


Рис. 6.23. Простой стабилизированный блок питания

Благодаря стабилитрону D2 с напряжением стабилизации 5 В и дополнительным резистором R2, напряжение на базе транзистора T1 поддерживается постоянным. Но, вспомним законы Кирхгофа и Ома, напряжение между базой транзистора и общим проводом должно быть равно сумме напряжения база-эмиттер транзистора и напряжения в нагрузке R1. Напряжение база-эмиттер транзистора составляет 0.7-1 В и меняется очень мало, так устроен транзистор, в итоге и напряжение на сопротивлении нагрузки R1 меняется тоже очень мало. Изменяя сопротивление нагрузки от 5 кОм до 500 Ом, мы можем наблюдать, что выходное напряжение остается, практически, постоянным, тогда как ток, потребляемый от блока питания существенно изменяется. На диаграмме ниже показано напряжение на выходе выпрямителя (на конденсаторе C1) и на выходе блока питания.

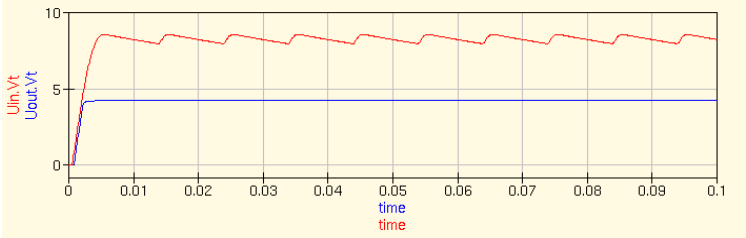


Рис. 6.24. Напряжения на входе и на выходе стабилизатора

Напряжение сети, напомню 100 В, а трансформатор понижает его в 10 раз. Предположим,

что напряжение в сети возросло до 150 В. Как поведет себя стабилизатор?

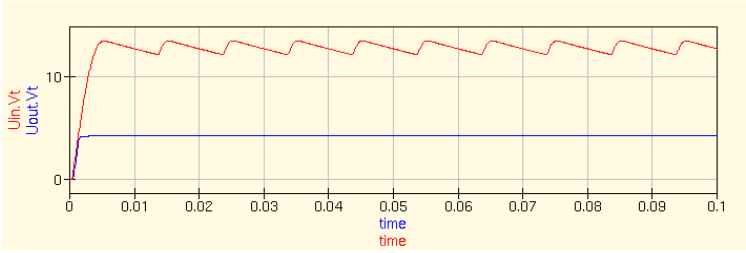


Рис. 6.25. Напряжение на входе и выходе стабилизатора при увеличении сетевого

Напряжение на входе стабилизатора увеличилось с 8 до 12 В, тогда как напряжение на выходе стабилизатора осталось на прежнем уровне. И обратите внимание на то, что пульсации на выходе выпрямителя (на конденсаторе C1), заметные невооруженным глазом, практически отсутствуют на выходе стабилизатора. Кроме поддержания напряжения он еще и повышает качество выпрямленного напряжения.

Еще одна схема стабилизированного блока питания будет полезна для понимания того, как работают микросхемы стабилизаторов напряжения.

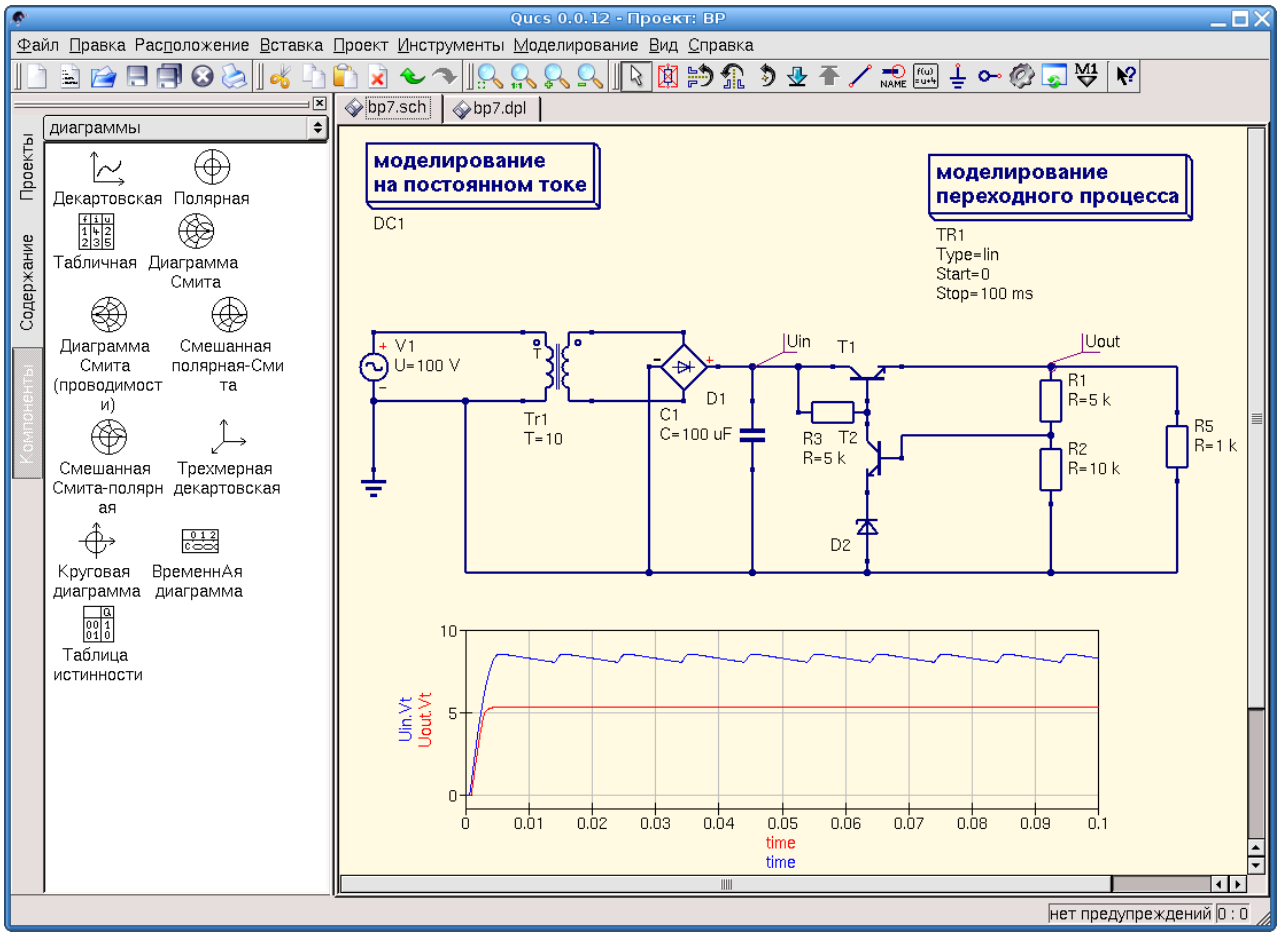


Рис. 6.26. Компенсационный стабилизатор напряжения

Напряжение на выходе стабилизатора напряжения (*Uout*) определяет ток в базовой цепи транзистора T2, чем больше это напряжение, тем больше базовый ток. Коллекторный ток

этого транзистора, связанный с базовым коэффициентом усиления по току, создавая падение напряжения на резисторе R3, определяет напряжение базы транзистора T1 относительно общего провода. А это напряжение, как и в предыдущей схеме, должно быть равно падению напряжения на переходе база-эмиттер T1 плюс выходное напряжение стабилизатора. С ростом базового тока транзистора T2 уменьшается эта сумма, а напряжение перехода у транзистора меняется мало, значит уменьшится напряжение на выходе стабилизатора. А это, как мы определили в начале, уменьшит базовый ток транзистора T2, устанавливая, в конечном счете, динамическое равновесие около напряжения стабилизации. Это напряжение задается делителем напряжения на резисторах R1 и R2. Можете, изменяя величину резистора R2, посмотреть, как изменится выходное напряжение. Стабилитрон D2 в эмиттерной цепи транзистора T2 существенно улучшает работу стабилизатора. Для сравнения замените его сопротивлением 500 Ом.

На диаграмме показано напряжение на выходе выпрямителя, верхний график, и напряжение на выходе стабилизатора. Как и в предыдущем случае стабилизированное напряжение получатся с меньшими пульсациями. Схема микросхем серии К142 устроена сложнее, но обе схемы работают схожим образом.

За счет чего во всех приведенных схемах достигается стабилизация напряжения? Конечно за счет того, что напряжение на выходе выпрямителя больше, чем выходное напряжение стабилизатора. Этот избыток напряжения и используется для компенсации выходного напряжения, если оно уменьшается, или в избыток (запас) напряжения отводится часть выходного, когда оно становится больше заданного. Транзистор T1 и сопротивление нагрузки (все сопротивления на выходе стабилизатора) образуют делитель напряжения, где сопротивление одного из элементов регулируется для получения соответствия с заданным значением напряжения деления. Избыток напряжения, как видно из схемы, падает на транзисторе, а, следовательно, на транзисторе выделяется мощность, зависящая от этого напряжения и тока, потребляемого нагрузкой блока питания. С одной стороны, чем больше запас напряжения, тем больше возможности по поддержанию стабильного напряжения, но тем больше и рассеиваемая транзистором мощность. На диаграмме видно, что избыток напряжения равен 3 В. При токе в 1 А на транзисторе выделяется мощность 3 Вт. Ни один транзистор, как правило, не может избавиться от тепла при такой выделяемой мощности. Тепло рассеивается корпусом транзистора за счет его контакта с воздухом. Эффективность рассеивания тепла определяется площадью поверхности корпуса и, конечно, температурой окружающего воздуха. Для отвода тепла транзистор в подобном случае крепят на специальном теплоотводе – радиаторе. Радиатор для увеличения поверхности при заданных габаритах делают ребристым или игольчатым. Чем больше поверхность, тем лучше отводится тепло. Но... до некоторого предела. Тепло от источника тепла, транзистора, растекается по теплоотводу, и если он очень большой, то тепло не будет успевать «добраться до окраин». Избыточная поверхность радиатора не принимает участия в теплообмене. Чтобы этого не происходило, радиатор для каждого конкретного случая рассчитывается. Среди параметров транзисторов есть ряд параметров, называемых тепловыми сопротивлениями, которые и входят в эти расчеты. Как проводник оказывает сопротивление протеканию тока, так и материалы корпуса транзистора и радиатора сопротивляются протеканию тепла, ухудшая отвод тепла от кристалла, из которого сделан транзистор. При креплении корпуса транзистора к радиатору стараются сделать его таким, чтобы не образовывалось воздушного промежутка. Для этой цели применяют специальные не высыхающие пасты с хорошей теплопроводностью. В компьютерной технике широко применяют принудительное охлаждение с помощью вентиляторов. Точный расчет радиатора достаточно сложен, хотя и не сложнее расчета самого блока питания, трансформатора, выпрямителя, стабилизатора. Этим расчетам следует посвятить отдельное место, а пока, просто не будем забывать о том, что все

элементы схемы могут нагреваться, и их следует охлаждать при сильном нагреве.

Конечно, если у вас еще нет блока питания для дальнейших экспериментов с электронными устройствами, и есть намерение самостоятельно изготовить его, наилучшее решение применить в качестве стабилизатора микросхему. Достаточно микросхемы серии КР142ЕН с фиксированным напряжением, скажем, 5 В. Такая микросхема имеет три вывода: вход, выход и общий. Как и транзистор, микросхему нужно установить на радиатор, если вы хотите получать на выходе стабилизатора токи порядка ампера и больше, а при выборе трансформатора можно принять, что выпрямленное напряжение станет равным амплитудному значению. Избыточного напряжения в 2-3 В будет достаточно для работы стабилизатора. При токе 0.5 А и избыточном напряжении 2 В на микросхеме рассеивается мощность 1 Вт. Микросхемы серии КР142ЕН могут рассеять такую мощность без дополнительного теплоотвода. Так что, снижая требования к выходным параметрам блока питания, можно существенно облегчить его конструкцию. Из этих же соображений можно сделать два блока питания, положим, один на 5 В, а второй на 12 В. Прежде, чем приступать к их изготовлению, можно в программе Qucs провести необходимые эксперименты, измерить все токи и напряжения, на которые и ориентироваться. Мощность, потребляемая нагрузкой от выпрямителя (сумма мощности в нагрузке стабилизатора и рассеиваемой транзистором), определит необходимую мощность трансформатора, но необходимо учитывать, что коэффициент полезного действия трансформатора не сто процентов, а имеет порядок 60-70%. Такой трансформатор и следует покупать. Лучше взять трансформатор большей мощности. Еще полезнее будет купить трансформатор большей мощности с несколькими выходными обмотками на разное напряжение. Впоследствии этот трансформатор можно использовать для улучшенных версий блока питания. Но самым надежным вариантом для вас может стать покупка готового блока питания с регулируемыми напряжениями. А создание своего блока питания можно превратить в интересный проект, где исследование свойств устройства – главная и единственная задача, которую можно решать с помощью программы, впоследствии переносится макетную плату и проверяется. Обретя опыт, вы сможете строить современные, например, импульсные блоки питания с заданными параметрами: нагрузочными, габаритными и т.д.

Чтобы легче было во всем этом разобраться, давайте вернемся к устройству, которое мне кажется одним из самых основных, позволяющих понять очень многое из жизни электроники – к усилителю.

Глава 7. Усилитель

Современный усилитель звуковой частоты широкого применения – это микросхема. Она почти не потребует дополнительных элементов, при правильном подключении будет работать сразу, не требуя наладки и многочисленных проверок. Но рассказ об усилителе я, вопреки всему, хочу начать с рассказа о каскаде усиления на радиолампе. Лампы, сегодня почти не используемые, служили нам верой и правдой долгие годы, и хотя бы по этой причине заслуживают, чтобы о них не забывали.

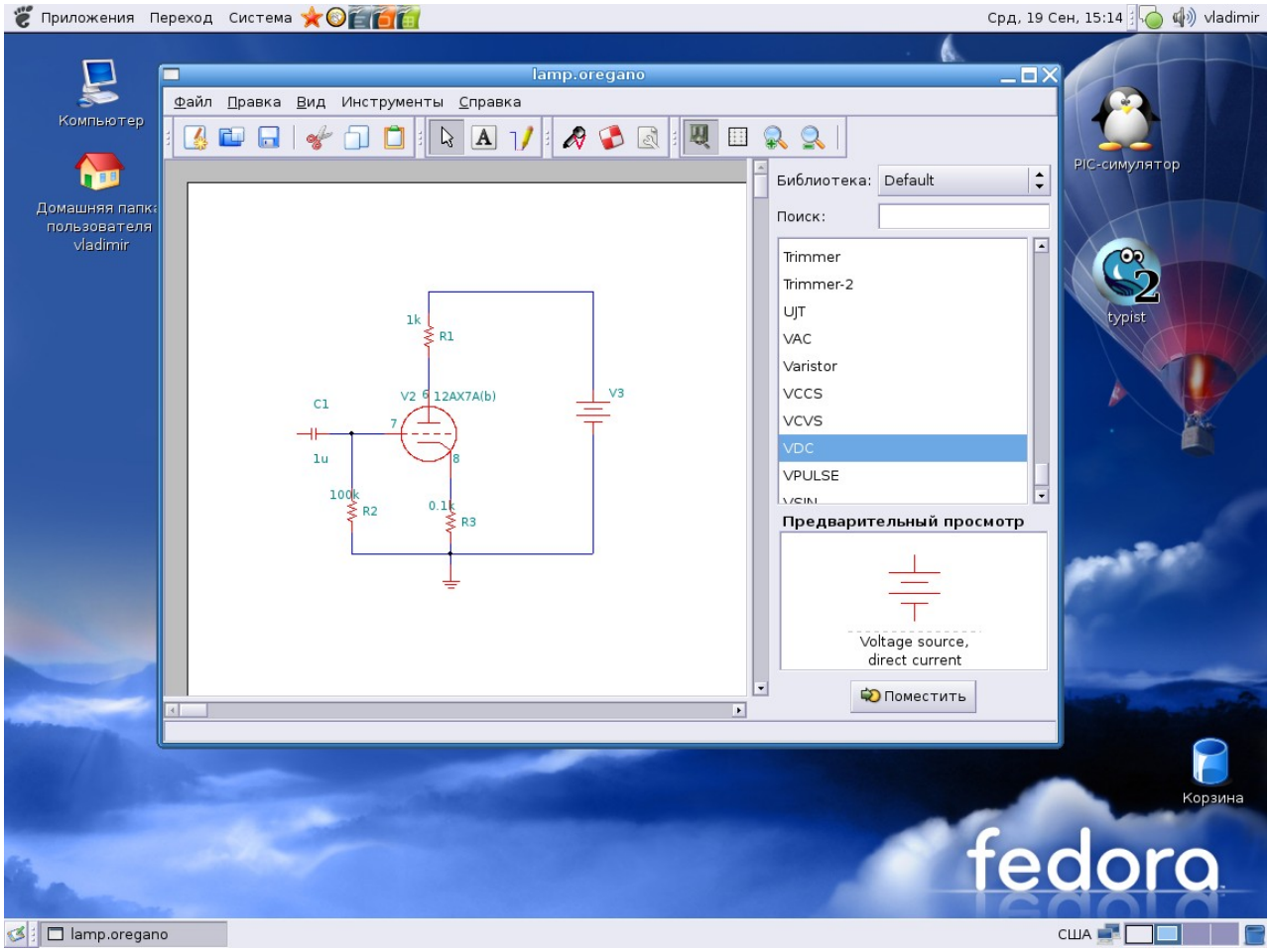


Рис. 7.1. Схема лампового каскада усиления

Схему я рисовал по памяти, мог и ошибиться. Радиолампа, чаще всего, имела цепь накала, но я выбрал рисунок лампы без нее, хотя с нити накала начинается работа лампы. Под действием приложенного напряжения нить накала разогревается настолько, что разогревает и катод, излучающий электроны в пространство между катодом (вывод 8) и анодом (вывод 6). Когда между катодом и анодом включается источник питания (V3), как правило достаточно большого напряжения в десятки и сотни вольт, то создаваемое им электрическое поле заставляет поток электронов от катода устремляться к аноду, создавая ток, создавая падение напряжения на резисторах R1 и R3. Напряжение на резисторе R1 – выходное напряжение каскада. А напряжение на резисторе R3, которое плюсом приложено к катоду, а минусом через резистор R2 к сетке, будет создавать электрическое поле, уменьшающее поток электронов к аноду. Напряжение сигнала, приходящее от предыдущего каскада через конденсатор C1, меняет это «запирающее» напряжение, формируя в анодной цепи лампы

усиленный сигнал. Считается, что понять работу лампы проще, чем транзистора. Возможно, если вам понятно, как электроны покидают свои «насиженные места» в материале катода, как умудряются путешествовать между катодом и анодом, и почему сетка умудряется ими управлять. Хотя, я согласен, если отвлечься от этих «тонкостей», то поток электронов, подобно водному потоку или потоку слов, которые я умудряюсь извлечь из клавиатуры, легче укладываются в понятие электрический ток. А аналогия с сообщающимися сосудами, когда один из них поднимается на вышку, а второй остается на земле, легче поясняет, отчего ток больше при увеличении напряжения. Возможно.

С другой стороны, мы не видим ни потока, ни электрического тока. Все, что нам дано, это включить амперметр и измерить ток. Или подключить вольтметр, чтобы измерить напряжение. И достаточно, измеряя ток через любое сопротивление, менять напряжение источника питания, чтобы заметить, что с увеличением напряжения увеличивается ток. А как он «бегает» по проводам... не изгибать же провода синусоидой для переменного тока!

Физика интереснейший предмет, очень нужно изучать ее, чтобы знать сегодняшние модели физических процессов, но заниматься этим, возможно, я очень ошибаюсь, следует тогда, когда интересна сама физика, или когда в твоей области интересов тебе не удается решить свою задачу без знания каких-то физических особенностей процесса.

Вспомнить о радиолампе меня заставили сомнения – рассказ продвинулся далеко вперед, а все ли ясно с такими важными, но не всегда понятными вещами, как электрический ток, напряжение, я не говорю об электромагнитном поле. Понятно ли, как следует обращаться с транзистором? Мне немного не по себе отого, что я «пожадничал» и не рассказал о методах контурных токов и узловых потенциалов, когда рассказывал о законах Кирхгофа, и чтобы немного исправить свою оплошность, я, пожалуй, расскажу побольше о транзисторах. Для рассказа я постараюсь активнее использовать книгу Qucs Workbook, где авторы проекта показывают, как много интересного можно выполнить при работе с этой программой.

Включение транзистора с общей базой

Начнем с того, как транзистор включается в схему? У него три вывода: эмиттер, база и коллектор. Эмиттер, подобно катоду лампы, поставляет носители тока в таинственные глубины транзистора. База, не так как сетка лампы, но как умеет, формирует ток коллектора. А коллектор, отдувается за всех, формируя усиленный сигнал на нагрузке транзистора, чаще всего в виде сопротивления. Если у транзистора три вывода, то мы можем любой из них использовать в качестве общего вывода для входного и выходного сигналов (напряжений или токов).

Самый распространенный способ включения транзистора – с общим эмиттером. Реже применяется включение транзистора с общим коллектором. Еще реже – с общей базой. С этой редкости и начнем.

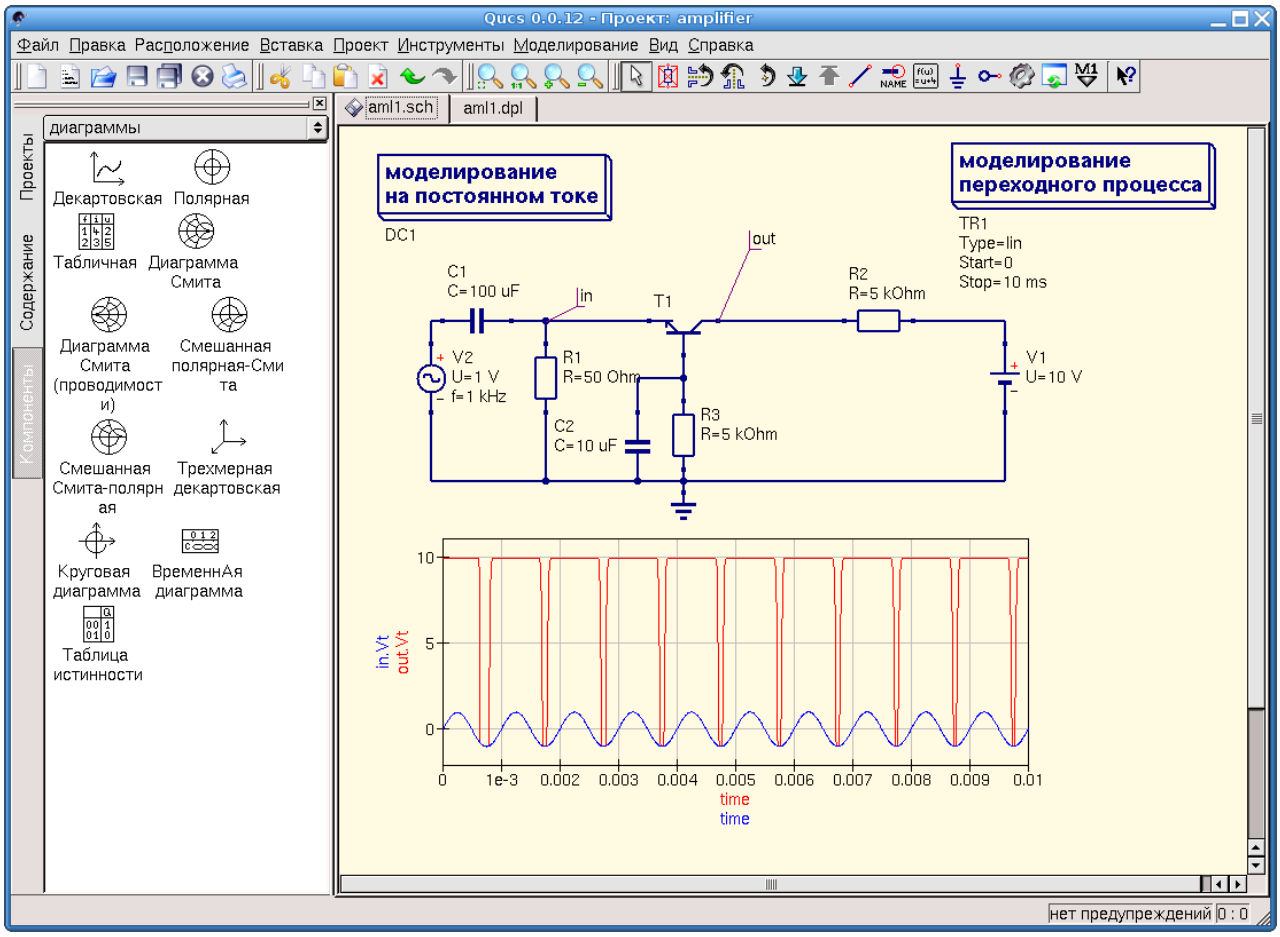


Рис. 7.2. Включение транзистора с общей базой

Я давно не пользовался схемами, где транзистор включается с общей базой, и давно забыл, как это выглядит, о чем мне напоминает диаграмма. В нижней ее части сигнал от генератора V2, красивая синусоида на входе усилителя, а в верхней части диаграммы сигнал на выходе, полное безобразие, которое я устроил, сделав ошибку, которой пока не вижу...

О-о! Как говорит мой сын: «Как все запущено!». Я столько говорил о необходимости базового тока для нормальной работы транзистора в усилительном режиме, а сам?! Вы, уже, верно, заметили, как я оплошал. Убрать, что ли этот рисунок? С другой стороны, жалко, красивый рисунок...

Я забыл задать ток базы, определяющий режим работы транзистора по постоянному току, увлекшись рассказом о том, как транзистор включается по отношению к сигналу. Для переменного напряжения с частотой 1 кГц, именно такую частоту генератора я задаю, сопротивления конденсаторов C1 и C2 невелико. Можно считать, что источник сигнала включен между эмиттером и базой транзистора. А выходной сигнал мы снимаем с базы-коллектора транзистора. То есть, база – общая точка для входного и выходного напряжений. Но для режима усиления транзистору необходимо задать правильный режим работы по постоянному току. На диаграмме видно, что в моменты, когда генератор создает ток базы, транзистор начинает работать в режиме усиления, а когда ток базы отсутствует, транзистор закрывается и напряжение на коллекторе становится равно напряжению источника питания V1. Сейчас я изменю положение элементов на рисунке и удалю источник переменного напряжения (сигнал). Все станет еще понятнее.

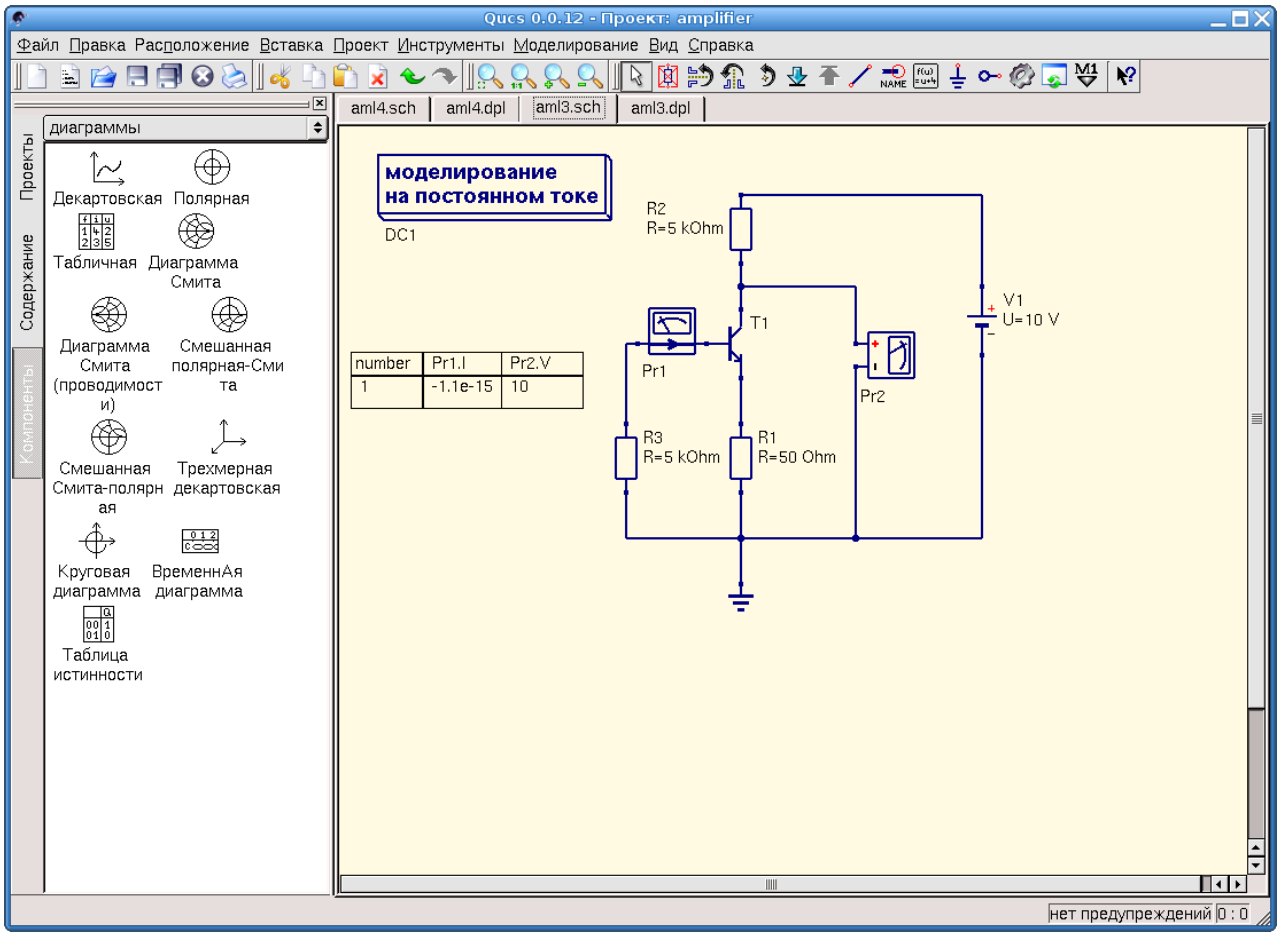


Рис. 7.3. Включение транзистора предыдущей схемы по постоянному току

Я добавил еще и амперметр в цепь базы, и вольтметр на выходе каскада, чтобы в таблице вы могли посмотреть, ток базы очень и очень мал, и этот ток – обратный ток (значение отрицательно) перехода коллектор-база. Любой p-n переход имеет очень маленький ток, но он есть, при обратном смещении, пока напряжение на переходе не превысит некоторого, предельно допустимого, значения. Вольтметр показывает напряжение источника питания, то есть, транзистор закрыт. По отношению к постоянному току обе схемы идентичны, отличаются только расположением элементов. И про резистор, задающий ток базы, я забыл. Отсюда и плачевный результат при подключении источника сигнала, источника переменного напряжения с частотой 1 кГц. Чтобы исправить свою ошибку, я добавлю резистор между коллектором и базой транзистора. Можно включить этот резистор и между базой и плюсом источника питания. В первом случае резистор будет выполнять и функцию элемента обратной (отрицательной) связи, во втором этого не произойдет. Но в обоих случаях ток базы, задающий режим работы транзистора, появится. Проверим это.

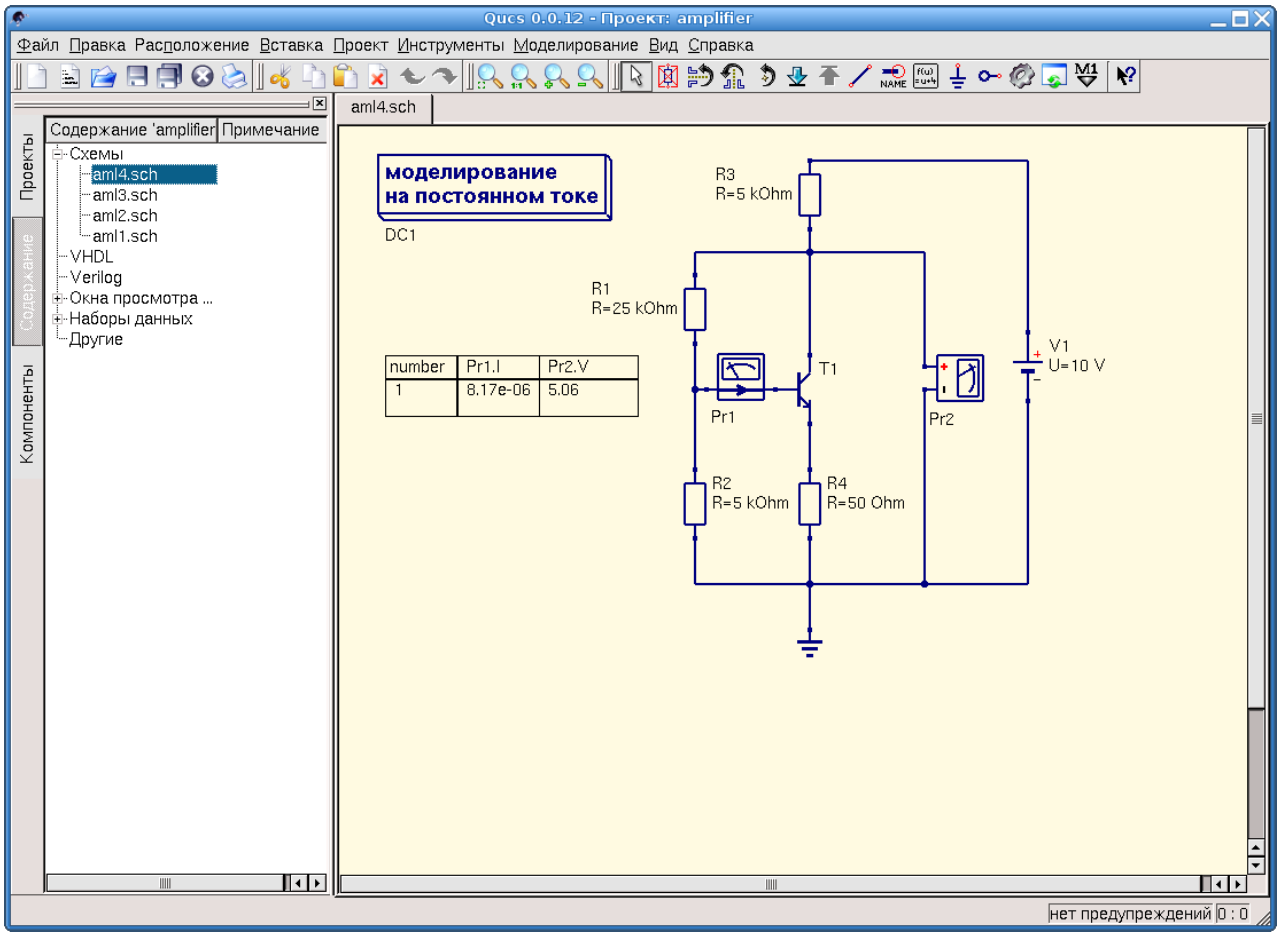


Рис. 7.4. Правильное включение транзистора по постоянному току

Базовый ток, как видно из таблицы, увеличился на несколько порядков, что привело к появлению коллекторного тока, а резистор R1 я подобрал так, чтобы напряжение на коллекторе транзистора было равно половине питающего напряжения, что тоже отражено в таблице (Pr2.V).

Да, дал я маху, и самое время сделать вид, что все так и задумано (если бы), самое время продемонстрировать мои глубокие познания в области работы с токами и напряжениями. С этой целью я, я уже обронил фразу об обратной связи, хочу показать, как при таком включении резистора R1 образуется обратная связь по постоянному и переменному току. Резистор R1 – активное сопротивление, для которого в широком диапазоне частот справедливо то, что его сопротивление не зависит от частоты. Поэтому обратная связь, задаваемая с помощью резистора, начинает работать с нулевой частоты, или с постоянного тока. Нам потребуется вспомнить только то, что говорилось о законах Ома и Кирхгофа, что говорилось о делителе напряжения.

Рассмотрим выходное напряжение каскада, которое на рисунке измеряется вольтметром. К этому напряжению подключено несколько цепей: R1R2, транзистор плюс R4. Ток, протекающий от плюса источника питания (за положительное направление тока в технике, напомню, принято направление от плюса к минусу) через резистор R3, разветвится на небольшой ток к базовой цепи (через резистор R1) и основной ток коллектора, все в соответствии с законом Кирхгофа. Делитель напряжения, образованный резистором R1 и параллельно включенными резистором R2 и цепочкой перехода база-эмиттер плюс R4, делит выходное напряжение каскада. Та часть напряжения, которая снимается с резистора R2, и

создает напряжение обратной связи.

Попробуем, не используя ничего кроме известных нам законов, посчитать, какую часть выходного напряжения мы прикладываем ко входу каскада для создания обратной связи. Вначале определим сопротивление цепи, параллельной резистору R2. Базовый переход транзистора (база-эмиттер) – это включенный в прямом направлении p-n переход, напряжение на котором для полупроводника на основе германия 0.2-0.5 В, а на основе кремния 0.5-0.7 В. Разделим напряжение 0.5 В на ток базы (амперметр Pr1 показывает около 8 мкА) и получим сопротивление по закону Ома 62.5 кОм. Такое сопротивление, включенное параллельно резистору R2 = 5 кОм, напомню, что складывать следует обратные величины (1/Rобщ = 1/5 + 1/62.5), такое сопротивление мало изменит результат, даже после того, как к 62.5 кОм (62500 Ом) мы добавим R4= 50 Ом. Не делая большой ошибки мы можем считать, что делитель напряжения образован резисторами R1 и R2. А выходное напряжение делится в соотношении (R1 + R2)/R2, то есть, в шесть раз. В итоге этих мудреных подсчетов, мы получим, что на вход каскада на транзисторе T1 мы подаем 5/6 В (0.83 В). Включите вольтметр параллельно резистору R2 и убедитесь, что расчеты не слишком сильно отличаются от измерений. Осталось сообразить, какая это обратная связь, положительная или отрицательная? При увеличении тока базы (ток растет), вызванного увеличением напряжения на входе, растет ток коллектора, связанный коэффициентом усиления по току с этим током (свойство транзистора), что увеличивает падение напряжения на резисторе нагрузки R3. В этом случае напряжение на коллекторе транзистора, выходное напряжение, уменьшается: по закону Кирхгофа сумма падений напряжений должна быть равна напряжению источника питания – если одно падение напряжения увеличивается, другое должно уменьшиться. В итоге увеличение напряжения на *входе* вызывает уменьшение напряжения на *выходе*, что свидетельствует в пользу того, что эти напряжения находятся в противофазах, а обратная связь будет отрицательной.

Если разбирать схему до конца, то резистор R4 тоже участвует в создании обратной связи. Ток эмиттера, протекающий по этому резистору, создает на нем падение напряжения. Поскольку ток эмиттера тоже зависит от входного сигнала, падение напряжения на резисторе R4 будет участвовать во всех процессах, происходящих в усилителе. Но базовый ток транзистора, управляющий этими процессами, определяется напряжением база-эмиттер. Следовательно, напряжения на резисторах R2 и R4 должны быть алгебраически (с учетом знака) сложены. Или иначе – напряжение на резисторе R2 должно быть равно сумме напряжения база-эмиттер и напряжения на R4. Или совсем иначе, напряжение база-эмиттер должно быть равно разности напряжения на R2 и напряжения на R4, которое явно активно участвует в жизни усилителя.

Мне не хочется углубляться в описание подсчетов и расчетов, что лучше делать, используя не слова, а формулы. С формулами не ошибешься, а столь длинное описание на словах, может привести к ошибкам, которые ускользнут от внимания. Мой совет, в программе Qucs, или в любой аналогичной программе, которой вы пользуетесь, или на макетной плате не поленитесь, проведите все эти измерения, они вам не соврут!

Ну, вот. Свои познания я продемонстрировал, можно внести исправления в первоначальную схему, чтобы все-таки увидеть, как сигнал с частотой 1 кГц усиливается каскадом на транзисторе, включенном с общей базой.

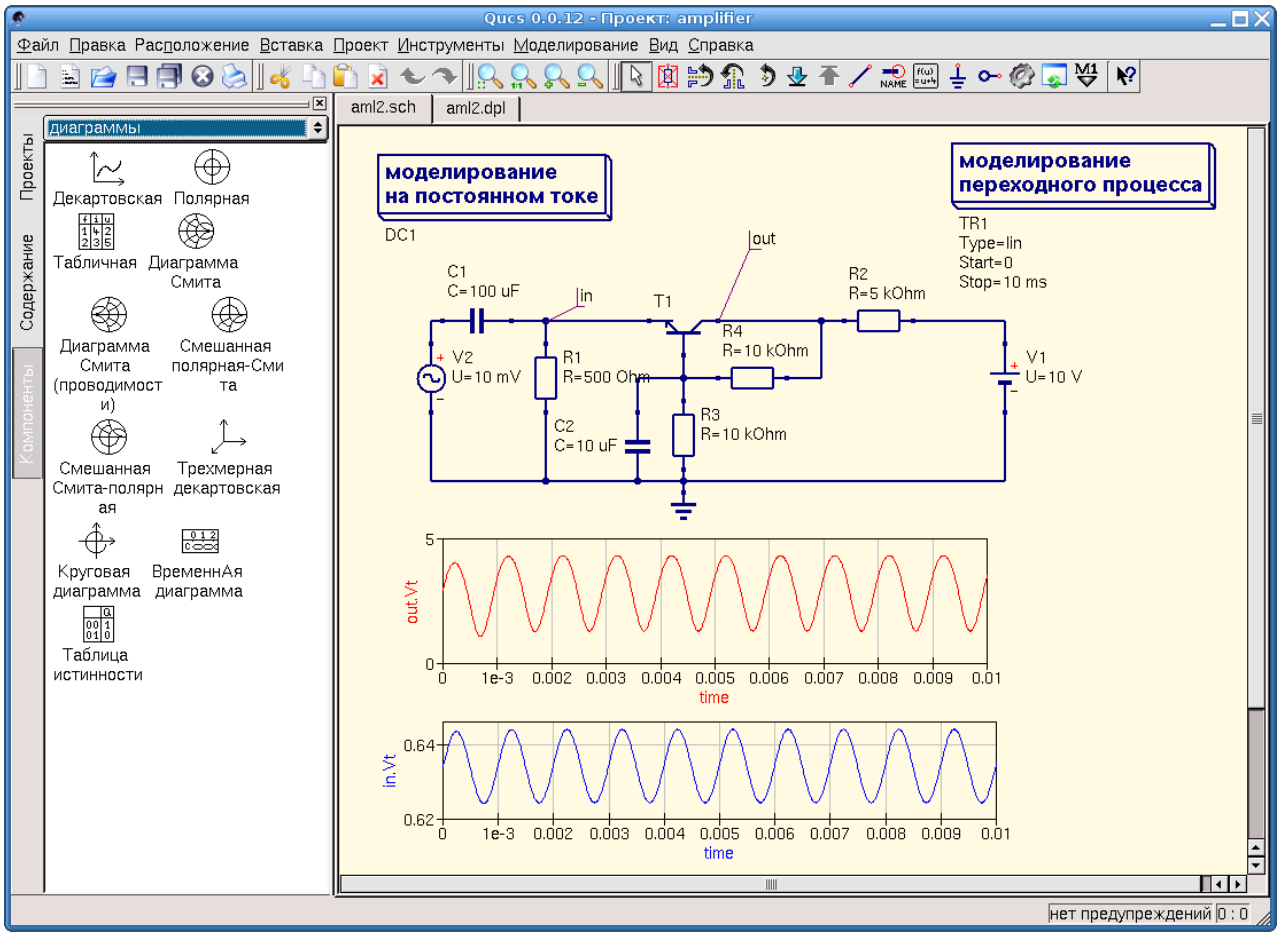


Рис. 7.5. Исправленная схема включения транзистора с общей базой

Надеюсь, что с включением транзистора с общим коллектором у меня получится удачней.

Включение транзистора с общим коллектором

Основным достоинством схемы с общим коллектором является большое входное сопротивление каскада и маленькое выходное. Именно эти свойства используются, например, в усилителе мощности звуковой частоты, когда транзистор работает на низкоомную нагрузку, громкоговоритель с сопротивлением звуковой катушки в несколько ом. Первоначально усилители на транзисторах, подобно ламповым усилителям, строили с выходным трансформатором. В этом есть свои преимущества, но и свои недостатки, к которым можно сразу отнести стоимость трансформатора и его габариты, соответственно, и вес. Бестрансформаторный выходной каскад усилителя мощности, кроме того, позволяет ввести очень глубокую отрицательную обратную связь, улучшающую параметры усилителя пропорционально глубине обратной связи. Правда, при глубокой отрицательной обратной связи транзисторных усилителей разработчики столкнулись с другими проблемами качества звука, но это отдельная история.

Итак, включение транзистора в схему с общим коллектором. То есть, общим для входного и выходного сигналов становится коллектор транзистора. Как это выглядит на схеме?

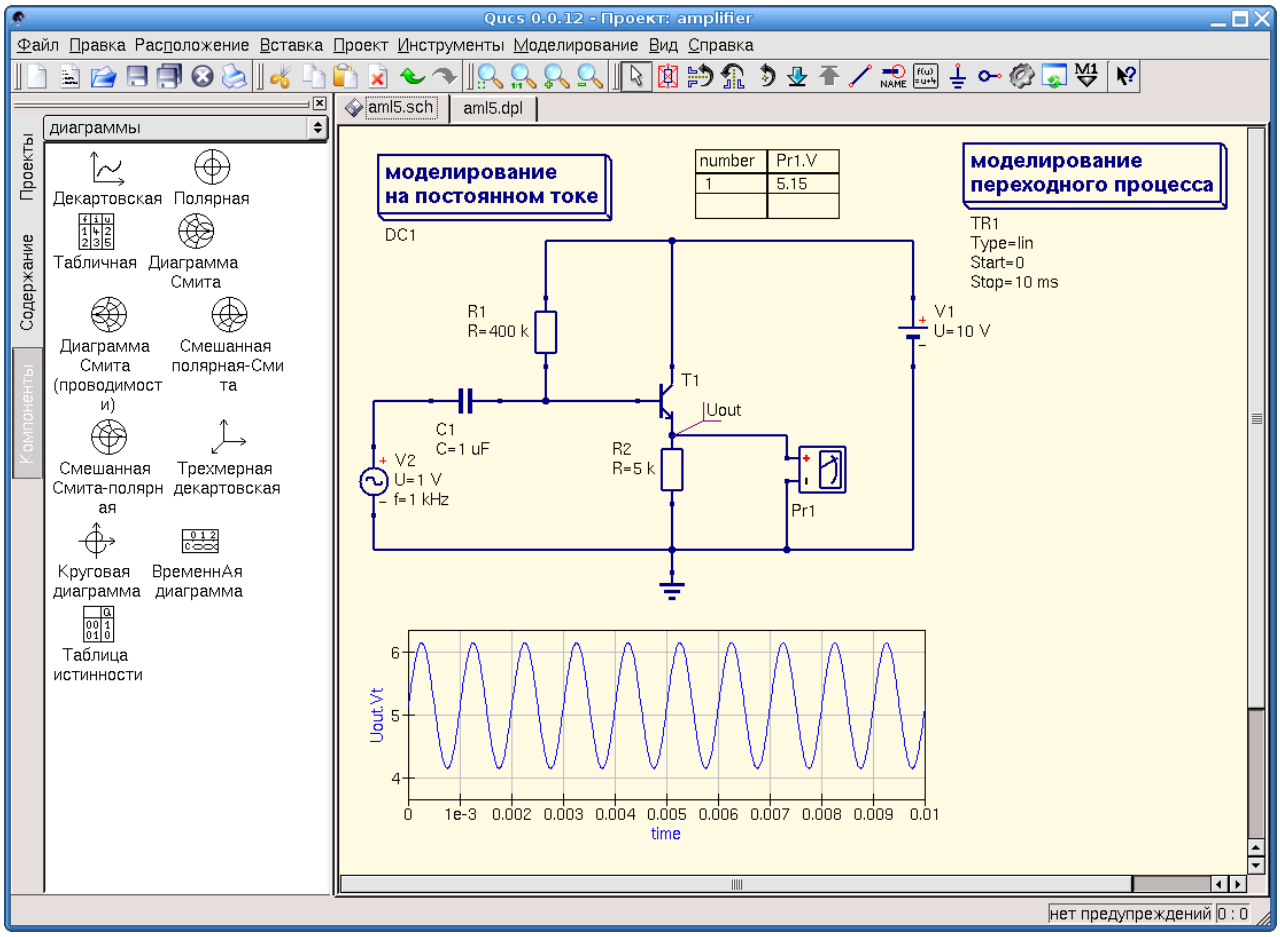


Рис. 7.6. Включение транзистора с общим коллектором

Гораздо лучше, чем в прошлый раз. Напомню, что для сигнала, переменного напряжения с частотой 1 кГц, сопротивление источника питания, практически нулевое, его можно «замкнуть накоротко», и тогда и вход усилителя, и его выход будут иметь общую точку, соединенную с коллектором транзистора, то есть, коллектор будет общим для входного и выходного сигналов. Вольтметр Pr1 показывает (таблица в верхней части), что резистор R1 выбран так, чтобы на постоянном токе напряжение на резисторе нагрузки каскада R2 было равно половине напряжения питания.

Что еще можно сказать, глядя на рисунок? Резистор R2, как и в предыдущей схеме, будет резистором обратной связи. Напряжение генератора (источника переменного напряжения V2) распределится между напряжением на этом резисторе и напряжением база-эмиттер транзистора, которое и будет, меняя базовый ток, управлять усилением сигнала. Обратная связь отрицательная, а напряжения, как видно из диаграммы, где амплитуда сигнала на резисторе R2 почти равна 1 В, скорее всего будут вычитаться. При включении транзистора с общим коллектором его сопротивление нагрузки, одновременно являясь сопротивлением отрицательной обратной связи, вводит глубокую обратную связь, вследствие чего усиление по напряжению такого каскада будет меньше единицы. А как же с усилением? Усиление по току будет существенным. Это и позволяет использовать, например, низкоомную нагрузку. И входное сопротивление, которое в предыдущей схеме было не больше 500 Ом, в этой может иметь порядок сравнимый с сопротивлением R1 = 400 кОм. Значит, такое включение особенно полезно, когда нужно получить большое входное сопротивление каскада усиления.

Включение транзистора с общим эмиттером

Этот способ включения транзистора наиболее распространен. В схеме с общим эмиттером транзистор имеет коэффициенты усиления по току и по напряжению значительно больше единицы. Что и позволяет говорить о схеме, как схеме усилителя, без каких-либо оговорок.

Я даже начну эту часть рассказа не со схемы, а с рассказа о способах задания рабочего режима по постоянному току.

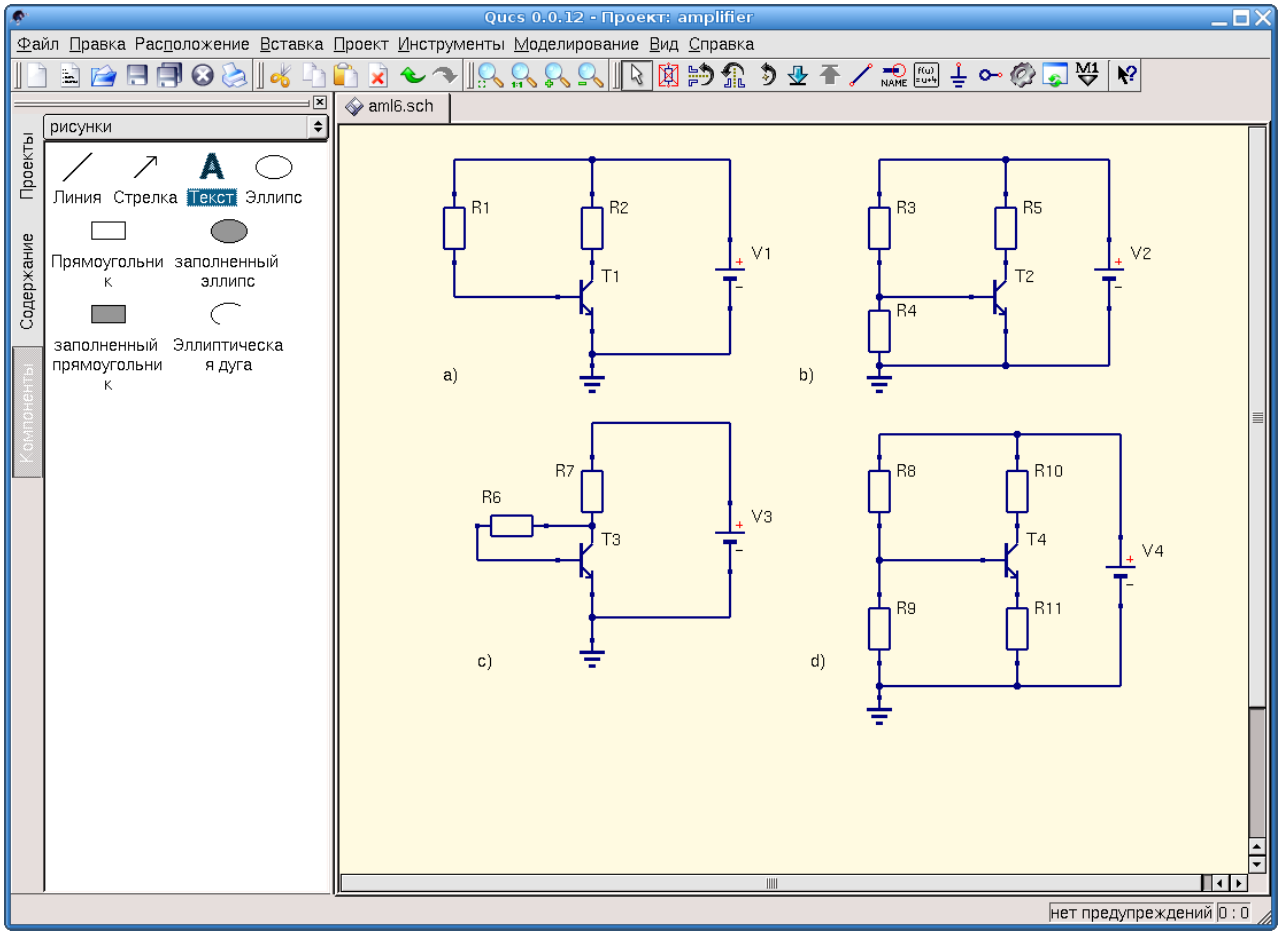


Рис. 7.7. Способы задания рабочего режима транзистора

Самый простой способ показан на рисунке 7.7.а. В зависимости от назначения каскада усиления напряжение на коллекторе транзистора может выбираться в пределах от долей напряжения источника питания до полного его значения. Однако часто это напряжение выбирается равным половине напряжения источника питания. В этом случае сигнал может менять это напряжение от близкого к нулю до почти равного напряжению питания, что очень важно при усилении симметричных сигналов, таких как синусоидальные. Реально сигнал не может достигать на выходе значения равного нулю. Некоторое напряжение, от долей вольта до 1-2 вольт, всегда остается на коллекторе транзистора сколько бы мы ни увеличивали ток базы. Это напряжение еще называют напряжением насыщения транзистора.

Резистор R2 является нагрузкой каскада на транзисторе. Его величину определяют, исходя из разных соображений. Часто величину нагрузки определяет каскад, следующий за данным каскадом усиления. Точнее его входное сопротивление. Если выходное сопротивление данного каскада будет больше, чем входное сопротивление следующего каскада, то можно ожидать искажений сигнала, связанных с перегрузкой. Для сигнала резистор R2,

динамическое сопротивление транзистора и входное сопротивление следующего каскада образуют параллельно включенные сопротивления. Если входное сопротивление следующего каскада значительно больше, чем первые два компонента общего сопротивления, то мы можем рассматривать параметры данного каскада независимо от следующего, тот не внесет больших изменений в работу.

Для определенности положим, что входное сопротивление следующего каскада 10 кОм. Тогда сопротивление нагрузки каскада на транзисторе T1 равное 2 кОм вполне можно считать подходящим. А если мы определили, что будем работать с синусоидальным сигналом, то нам нужно получить напряжение на этом резисторе равное половине напряжения питания, которое для определенности положи равным 10 В. При падении напряжения на резисторе R2 5 В и его сопротивлении 2 кОм мы получим (по закону Ома) ток через этот резистор 2.5 мА. Это ни что иное, как ток коллектора транзистора T1. Если транзистор имеет статический коэффициент усиления по току равный 100, то базовый ток однозначно должен быть равен 25 мкА. Именно этот ток определит величину резистора R1. Падение напряжения на этом резисторе практически равно (за вычетом напряжения 0.5-0.7 В на переходе база-эмиттер) напряжению питания. Разделив 10 В на 25 мкА мы получим значение сопротивления R1 = 400 кОм (если я правильно посчитал). Вот мы и определили все элементы нашего простейшего каскада с общим эмиттером. Осталось проверить их.

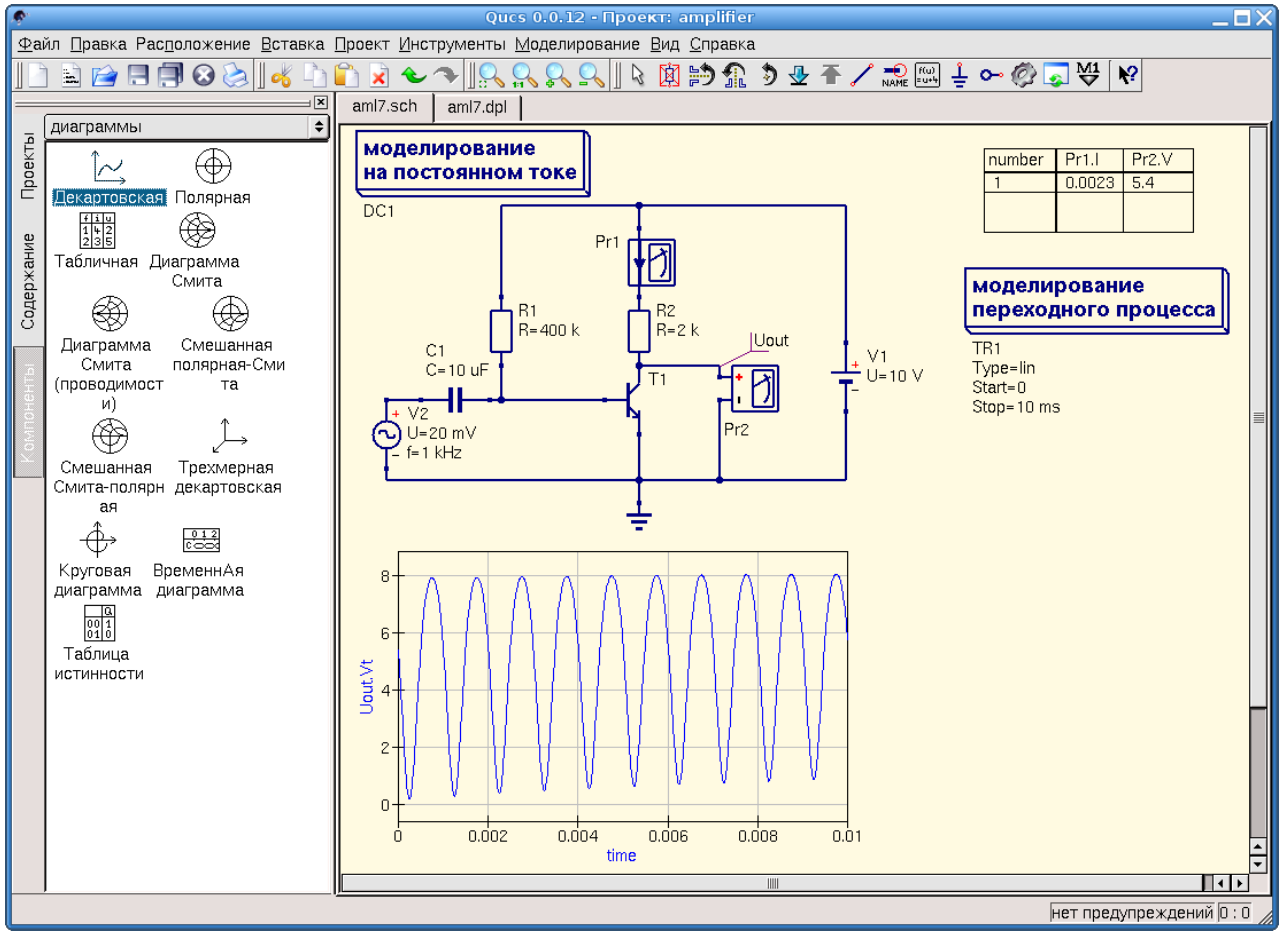


Рис. 7.8. Проверка расчетов простейшего усилителя

Расчеты были простейшие, но как видно на графике выходной сигнал искажен не очень сильно, а его амплитуда близка к половине напряжения питания. В таблице в верхней части рисунка можно увидеть ток коллектора (амперметр Pr1) равный 2.3 мА и напряжение на

коллекторе (вольтметр Pr2) равное 5.4 В. Расчетные данные близки к результату полученному в программе Qucs, хотя я уверен, что формулы, использованные для моделирования работы транзистора в программе, были значительно более точными. И еще одно. Я в первых главах говорил об источнике тока. Если мы рассмотрим источник напряжения V1 в сочетании с резистором R1 как источник тока, а сопротивление постоянному току база-эмиттер транзистора значительно меньше, чем 400 кОм, то базовый ток не должен зависеть от замены одного транзистора другим. Проверим это предположение. Перенесем амперметр Pr1 в цепь базы, измерим ток базы – 2.3e-05. После замены транзистора ток базы и напряжение на коллекторе будут:

number	Pr1.I	Pr2.V
1	2.34e-05	1.63

Рис. 7.9. Ток базы и напряжение на коллекторе 2N2222A

number	Pr1.I	Pr2.V
1	2.33e-05	0.982

Рис. 7.10. Ток базы и напряжение на коллекторе 2N4401

Как можно заметить, напряжения на коллекторе разных транзисторов различны, а токи базы схожи. А это может означать, что либо предположение было верно, либо величина сопротивления постоянному току цепи базы у разных транзисторов почти одинакова. Как бы это проверить?

На рисунке 7.7.b сопротивление база-эмиттер шунтируется резистором R4. Если исследуемое нами сопротивление значительно ниже 400 кОм, то ток после добавления резистора R4 через резистор R1 не должен сильно отличаться от полученных ранее значений... и не отличается.

Однако изменился ток базы. Часть тока, ранее уходившего в базу транзистора, ответвляется в добавленное нами сопротивление. Изменилось и напряжение на базе транзистора. Делитель напряжения, прежде образованный резистором R1 и сопротивлением перехода база-эмиттер транзистора, теперь формируется в нижнем плече параллельно включенными переходом и добавленным резистором. А от напряжения база-эмиттер зависит ток базы.

После замены резистора R1 на сопротивление 90 кОм ток базы можно вернуть к исходному значению, как на рисунке ниже.

Теперь рабочий режим транзистора задается делителем напряжения в его базовой цепи. Если выбрать резистор R3 (на рисунке 7.11) сопротивлением заведомо ниже сопротивления перехода база-эмиттер на постоянном токе, то именно напряжение делителя будет полностью определять режим работы.

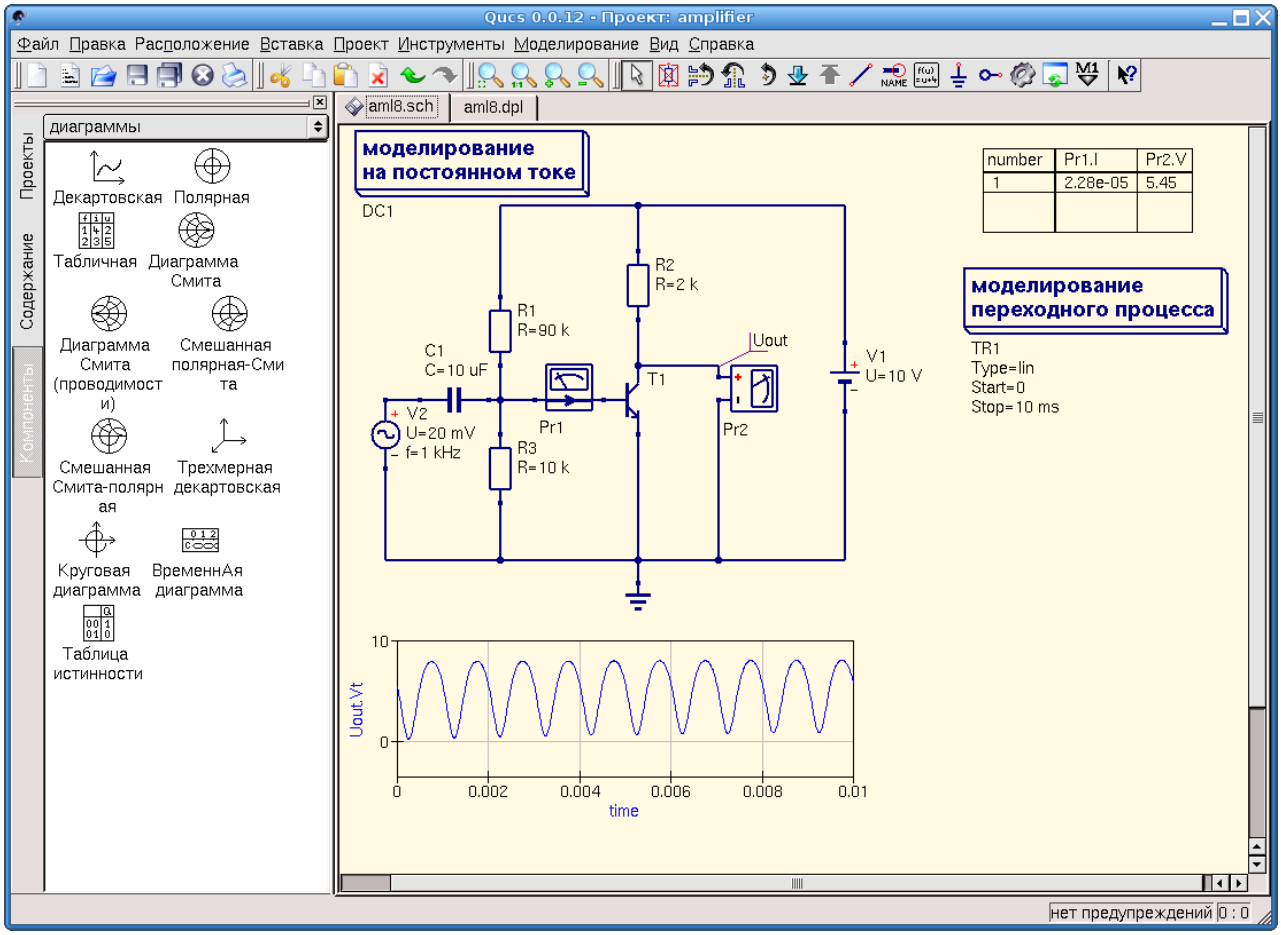
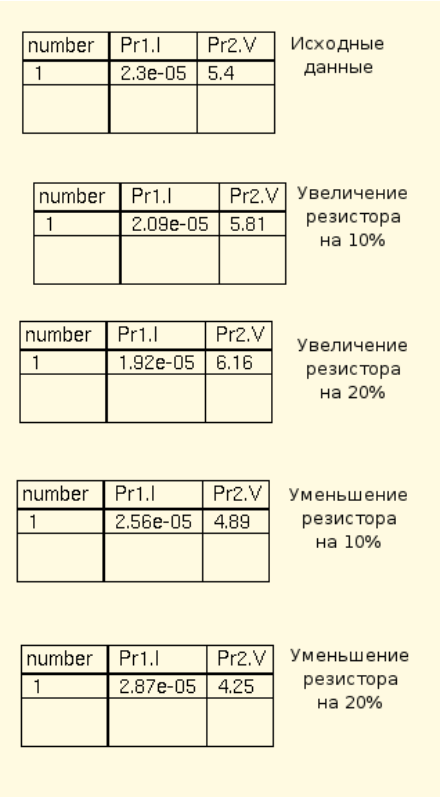


Рис. 7.11. Токи и напряжения в схеме с делителем напряжения на входе

Расчет такой схемы несколько сложнее. Для его проведения необходимо знать зависимость тока базы от напряжения база-эмиттер. Такие данные можно получить из документации к полупроводниковому прибору или снять зависимость самостоятельно. Резисторы R1 и R3 образуют делитель напряжения, а падение напряжения на R3 определит ток базы, а соответственно и ток коллектора. В остальном порядок расчета одинаков. При замене транзистора на другой, похожего, но другого типа, параметры схемы следует подбирать заново, то же происходит при изменении напряжения питания. И возникает вопрос – а зачем усложнять схему, если это не дает заметного выигрыша?

Первую схему я бы назвал схемой с фиксированным током базы. Вторую с фиксированным напряжением база-эмиттер. Мы говорили выше, что при изменении температуры величина сопротивления меняется. В первой схеме ток базы, определяющий все параметры схемы, зависит от величины сопротивления одного резистора, который будет менять базовый ток под действием температуры. Во второй схеме при таком же действии температуры на резисторы оба изменят сопротивление, но напряжение на резисторе R3 останется неизменным. Программа Qucs позволяет вам проверить мои рассуждения, возможно, ошибочные. Вы можете построить зависимость тока базы от напряжения, можете изменить значения резисторов. Вот таблицы, которые иллюстрируют сказанное.



number	Pr1.I	Pr2.V
1	2.3e-05	5.4

Исходные данные

number	Pr1.I	Pr2.V
1	2.09e-05	5.81

Увеличение резистора на 10%

number	Pr1.I	Pr2.V
1	1.92e-05	6.16

Увеличение резистора на 20%

number	Pr1.I	Pr2.V
1	2.56e-05	4.89

Уменьшение резистора на 10%

number	Pr1.I	Pr2.V
1	2.87e-05	4.25

Уменьшение резистора на 20%

Рис. 7.12. Таблицы испытаний схемы рисунка 7.8 при изменении R1

Сопротивление нагрузки оставалось неизменным. В качестве нагрузки мог оказаться колебательный контур или наушники, которые при изменении температуры ведут себя иначе. Таблицы, представленные выше, я собрал по результатам экспериментов с помощью графического редактора GIMP. Но Qucs предоставляет возможность провести этот эксперимент иначе, с помощью развертки параметра.

Для этого необходимый мне параметр, в данном случае сопротивление R1, я обозначаю как Rvar. После выбора вида моделирования в свойствах *Развертки параметра* достаточно выбрать в качестве моделирования DC1 (моделирование на постоянном токе предварительно добавлено), в качестве параметра для развертки Rvar, начало задать как 320 кОм, конечное значение параметра 480 кОм, и шаг 40 кОм. Пять точек измерения должны соответствовать прежде проделанным измерениям.

Запуск моделирования и выбор вида диаграммы в виде таблицы не отличается от обычных завершающих операций, они производятся после выполнения моделирования, когда появляется рабочее поле для сохранения результатов измерений в файле с расширением dpl. В диалоговом окне выбора переменных для отображения на диаграмме соответственно выбраны Pr1 и Pr2 — показания амперметра в цепи базы и показания вольтметра, подключенного к коллектору транзистора.

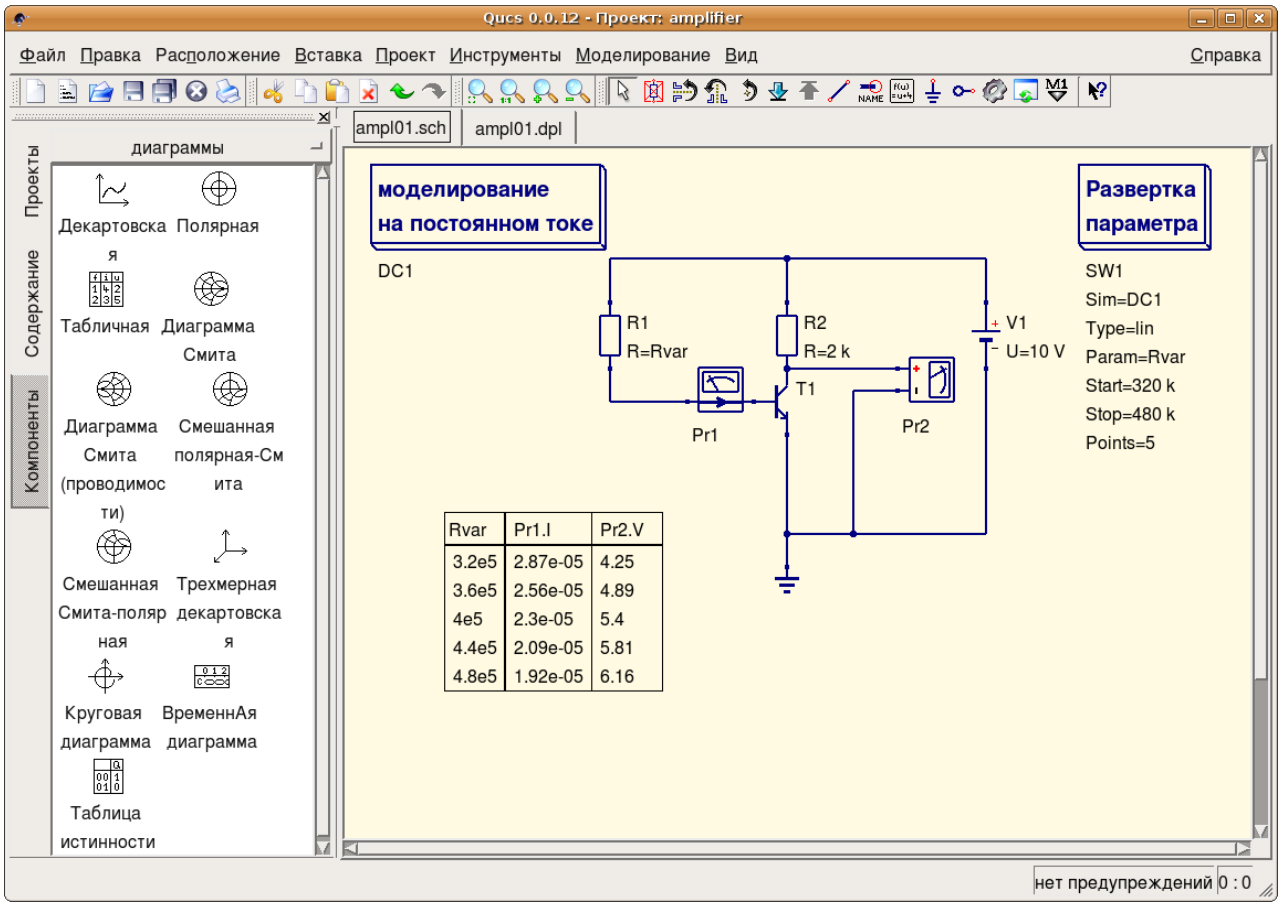


Рис. 7.13. Применение моделирования в виде развертки параметра

Согласитесь, удобно.

А теперь я хотел бы повторить изменения в тех же пределах резисторов R1 и R3 схемы на рисунке 7.11, но с начальными значениями 11.3 кОм и 1 кОм. Повторить развертку параметра я не могу, поскольку параметров два, а я не знаю, как в этом случае поступить, придется проделать несколько повторных запусков моделирования и собрать результаты в графическом редакторе.

number	Pr1.I	Pr2.V	
1	2.13e-05	5.74	Исходное значение резисторов

number	Pr1.I	Pr2.V	
1	2.13e-05	5.75	Увеличение резистора на 10%

number	Pr1.I	Pr2.V	
1	2.11e-05	5.79	Увеличение резистора на 20%

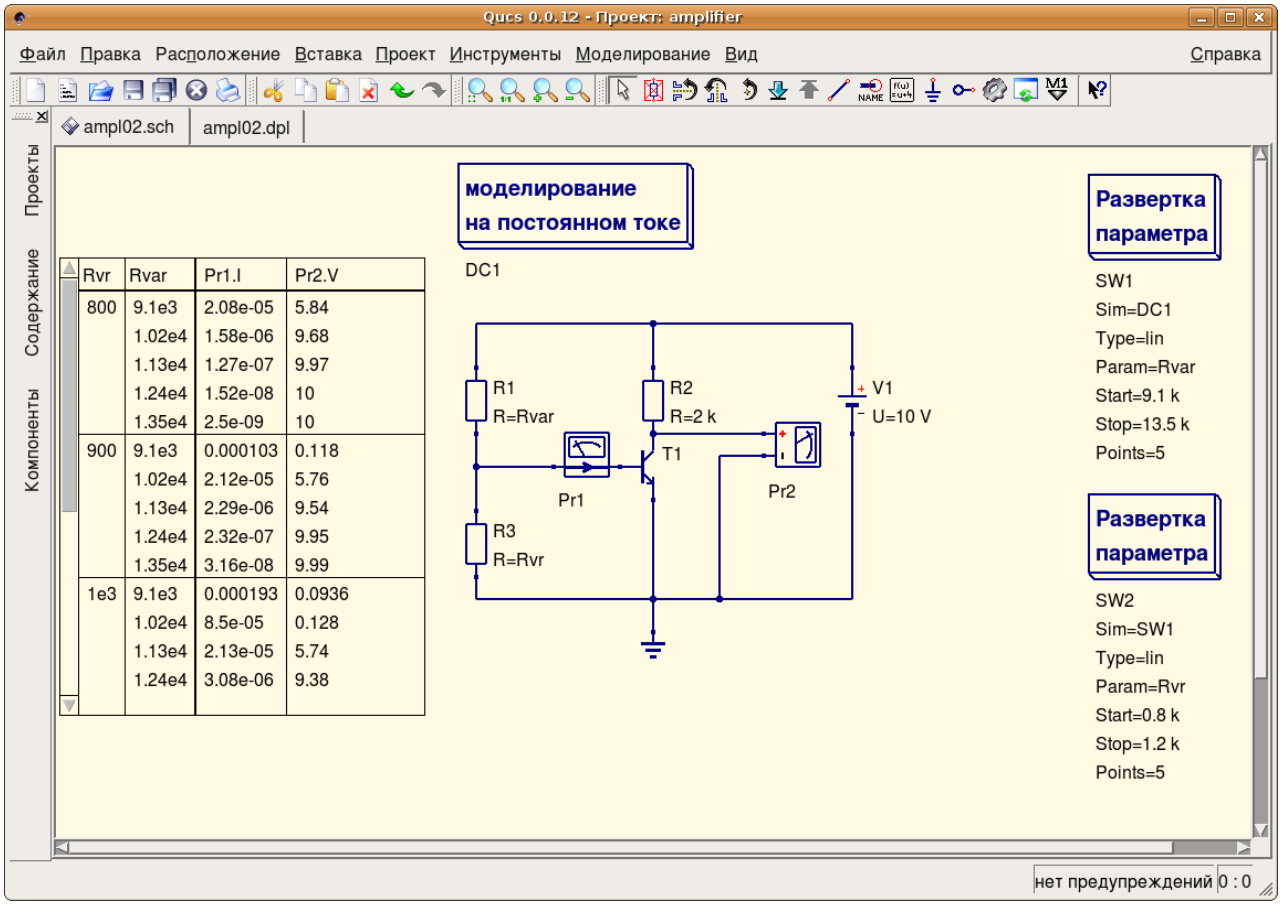
number	Pr1.I	Pr2.V	
1	2.12e-05	5.76	Уменьшение резистора на 10%

number	Pr1.I	Pr2.V	
1	2.08e-05	5.84	Уменьшение резистора на 20%

Рис. 7.14. Результаты измерений параметров схемы рисунка 7.11 при вариации R1 и R3

Как видно из таблиц, результаты лучше, чем предыдущие. И эксперимент не столь сложен. Изменить величину двух резисторов, запустить моделирование, получить результат... Но я

подумал, может быть, лучше было так:



Rvr	Rvar	Pr1.I	Pr2.V
800	9.1e3	2.08e-05	5.84
	1.02e4	1.58e-06	9.68
	1.13e4	1.27e-07	9.97
	1.24e4	1.52e-08	10
	1.35e4	2.5e-09	10
900	9.1e3	0.000103	0.118
	1.02e4	2.12e-05	5.76
	1.13e4	2.29e-06	9.54
	1.24e4	2.32e-07	9.95
	1.35e4	3.16e-08	9.99
1e3	9.1e3	0.000193	0.0936
	1.02e4	8.5e-05	0.128
	1.13e4	2.13e-05	5.74
	1.24e4	3.08e-06	9.38

Рис. 7.15. Развертка двух параметров в программе Qucs

Схема установки рабочей точки транзистора с помощью делителя напряжения решает некоторые проблемы, но не все. Посмотрим, что нам даст схема на рисунке 7.7.c. Она похожа на схему 7.7.a, но резистор включен в цепь отрицательной обратной связи. Расчет, думаю, такой же, только напряжение для вычисления величины резистора следует взять равным половине напряжения питания. Я попробую просто уменьшить это значение вдвое.

И, конечно, я воспользуюсь разверткой параметра в качестве вида моделирования. А чтобы легче было сравнить результаты, вставлю таблицу результатов первого эксперимента с помощью графического редактора.

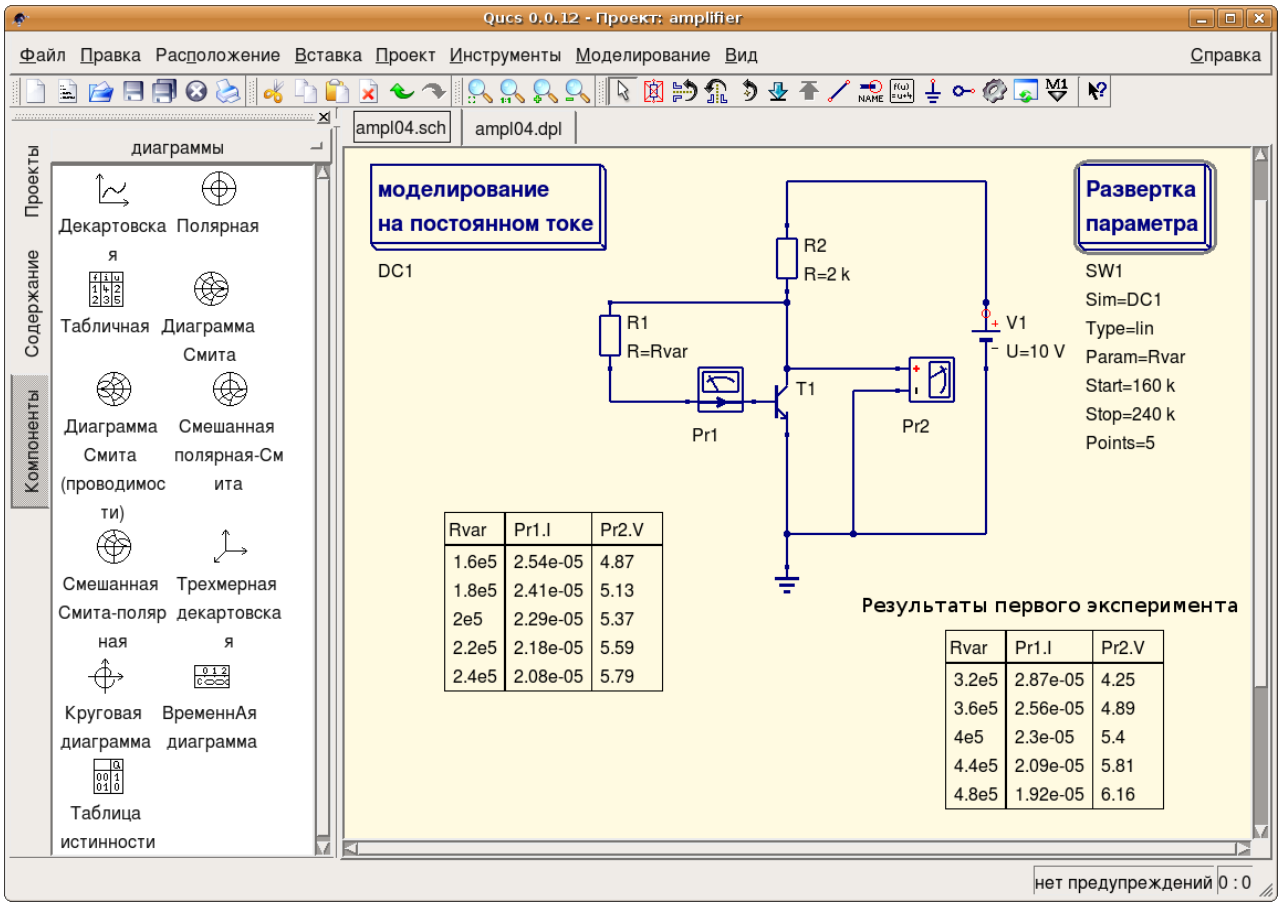
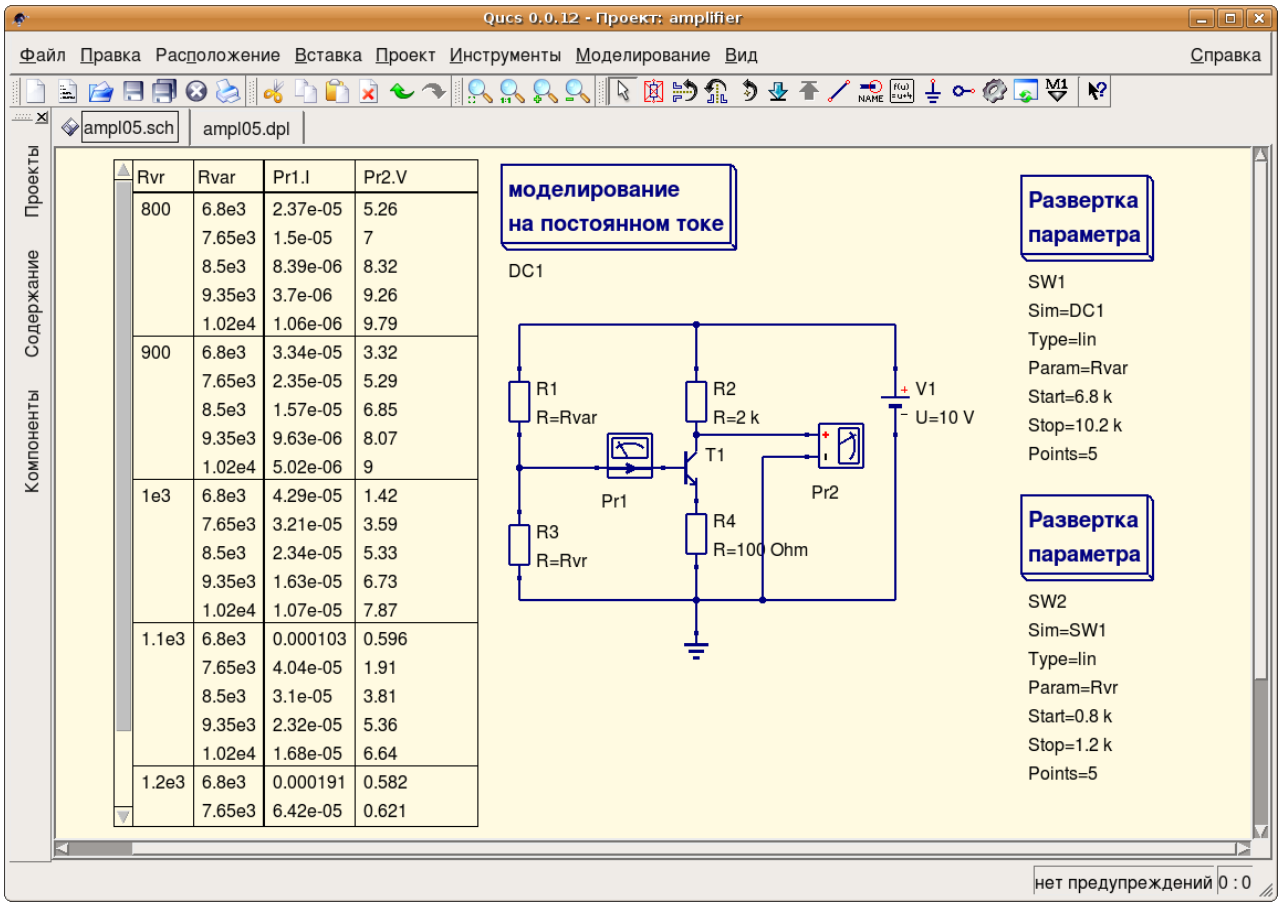


Рис. 7.16. Испытания схемы рисунка 7.7.c

И эти результаты явно лучше, чем результаты первого эксперимента. Остается проверить только схему на рисунке 7.7.d. Резистор, добавляемый в цепь эмиттера транзистора, вы уже знаете это, тоже добавляет обратную связь по постоянному току в схему каскада усиления. Ниже я приведу схему эксперимента, но хочу добавить, что никто и ничто не мешает и в этом случае использовать вторую обратную связь, как в схеме на рисунке 7.7.c. Каскад будет иметь два резистора отрицательной обратной связи. Удовольствие провести эксперименты с этим вариантом я оставлю вам.

Обратите внимание на то, что резистор в эмиттерной цепи вводит последовательную обратную связь, напряжение обратной связи включено последовательно со входным напряжением, что увеличивает входное сопротивление каскада.

Тогда же, когда вы включаете сопротивление между коллектором и базой транзистора, вы вводите параллельную обратную связь. При такой обратной связи входное сопротивление каскада уменьшается. Как в программе Qucs проверить входное сопротивление... об этом я рассказывал, а проверить его на макетной плате... тоже рассказывал, но так же, как и в программе Qucs.



Rvr	Rvar	Pr1.I	Pr2.V
800	6.8e3	2.37e-05	5.26
	7.65e3	1.5e-05	7
	8.5e3	8.39e-06	8.32
	9.35e3	3.7e-06	9.26
	1.02e4	1.06e-06	9.79
900	6.8e3	3.34e-05	3.32
	7.65e3	2.35e-05	5.29
	8.5e3	1.57e-05	6.85
	9.35e3	9.63e-06	8.07
	1.02e4	5.02e-06	9
1e3	6.8e3	4.29e-05	1.42
	7.65e3	3.21e-05	3.59
	8.5e3	2.34e-05	5.33
	9.35e3	1.63e-05	6.73
	1.02e4	1.07e-05	7.87
1.1e3	6.8e3	0.000103	0.596
	7.65e3	4.04e-05	1.91
	8.5e3	3.1e-05	3.81
	9.35e3	2.32e-05	5.36
	1.02e4	1.68e-05	6.64
1.2e3	6.8e3	0.000191	0.582
	7.65e3	6.42e-05	0.621

Рис. 7.17. Испытания схемы рисунка 7.7.d

Определение зависимости тока базы от напряжения

Для расчета режима транзистора по постоянному току с делителем напряжения удобно воспользоваться графиком зависимости тока базы от напряжения база-эмиттер. В старых справочниках по транзисторам можно было встретить такие графики. Но ничто не мешает построить эту зависимость, если это интересно, самостоятельно, либо в программе Qucs, как я собираюсь это сделать, либо с помощью макетной платы, на которой будет установлен транзистор и необходимые источники питания. Схемы проведения экспериментов получатся одинаковыми, но при реальной экспериментальной проверке следует быть очень внимательным, чтобы не испортить транзистор. Если в вашем распоряжении есть функциональный генератор с выходным сигналом треугольной формы, и генератор может работать на очень низких частотах, скажем, 0.1 Гц, то можно включить его на вход схемы и снимать показания, записывая их в таблицу.

По полученной зависимости, если она в табличном виде, легко построить график. Полученная зависимость может, в свою очередь, зависеть от других параметров, например, напряжения на коллекторе транзистора или температуры. Я не думаю, что вы настолько педантичны, что захотите проводить реальные эксперименты по полной программе. Я даже не думаю, что подобный эксперимент будет иметь для вас большее значение, чем первоначальное знакомство с предметом. Но при использовании программы вы имеете возможность не заботится о сохранности транзисторов и приборов, и в этом случае вы можете проводить подобные эксперименты всякий раз, когда у вас возникают сомнения в правильном понимании происходящего.

Итак. Транзистор включен по схеме рисунка 7.7.b. Вместо того, чтобы подбирать значения сопротивлений, мы хотим получить эти значения расчетным путем.

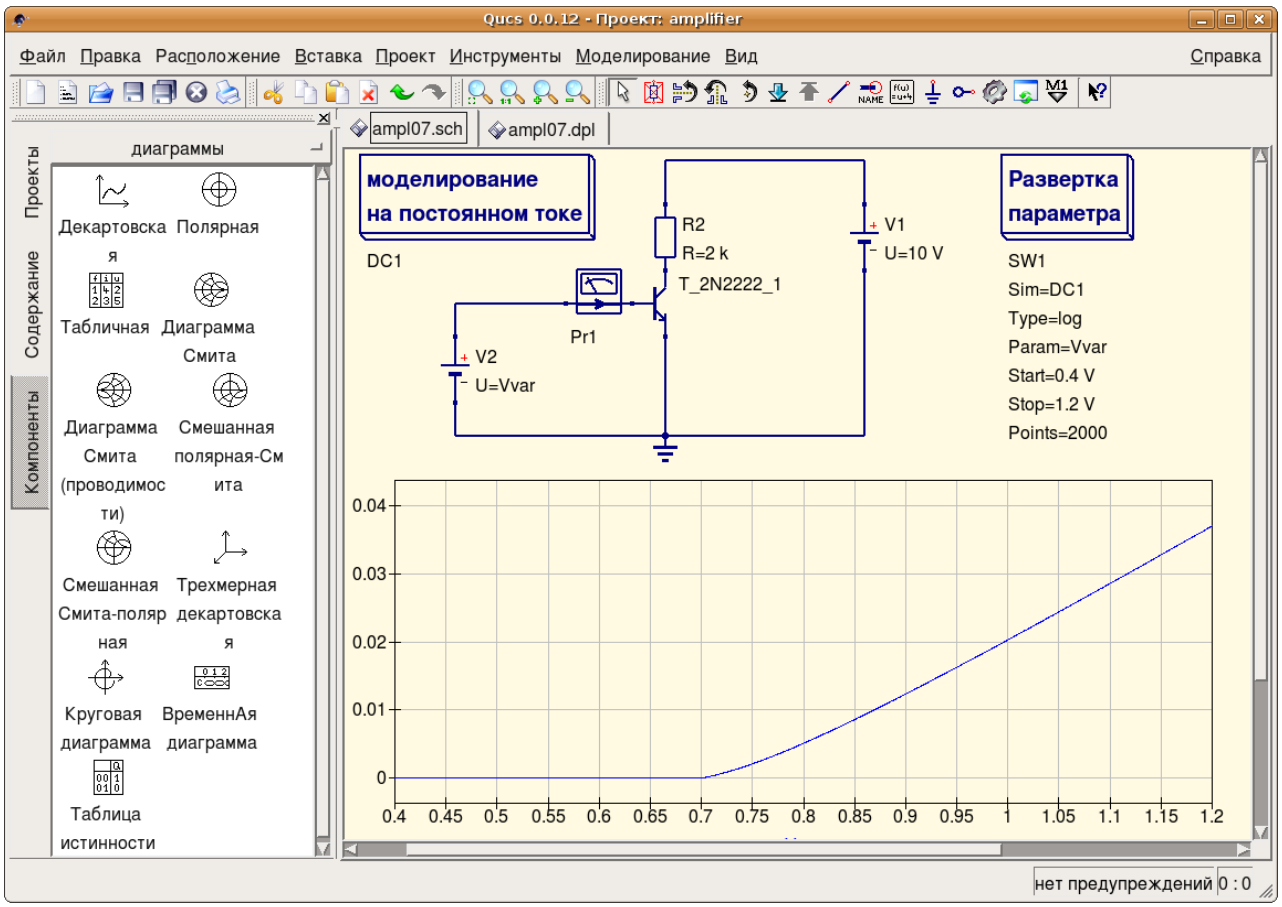


Рис. 7.18. Получение зависимости тока базы от напряжения

И рядом разместим результаты измерений параметров другого транзистора.

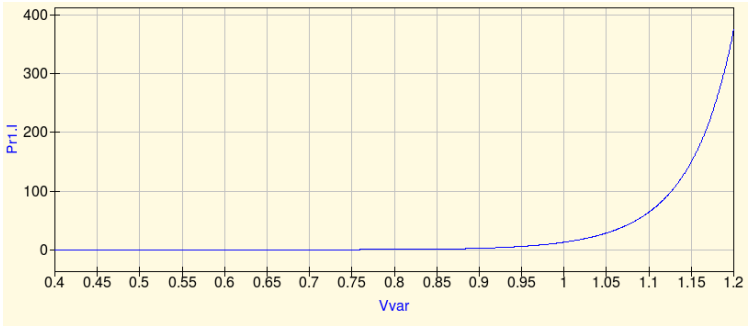


Рис. 7.19. График зависимости тока базы для транзистора 2N3507

Преимущество использования программы в том, что токи порядка 100 мА, если мы будем ориентироваться на предыдущие расчеты, нас будут мало интересовать, а именно такие зависимости вы можете обнаружить в справочнике. В программе достаточно изменить пределы «качания» параметра, то есть, начальную и конечную точку. Возьмем, например, в качестве начальной точки 0.75 В, а в качестве конечной 0.85 В, и повторим предыдущее моделирование в этом диапазоне значений.

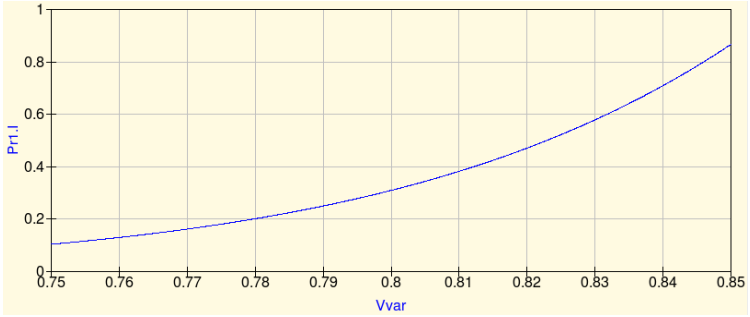


Рис. 7.20. График зависимости тока базы в заданном диапазоне напряжений

Вместе с тем, есть два фактора, могущие повлиять на ваше решение о способе проведения эксперимента. Первый, вы не хотите использовать программу, а второй — отсутствие данных о взаимозаменяемости транзисторов, и отсутствие нужного вам транзистора в библиотеке компонентов программы. В этом случае вы решите провести реальный эксперимент. Думаю, и в этом случае вы уже не должны испытывать затруднений.

Конечно, я, снимая характеристики, мог себе позволить использовать 2000 точек. Не думаю, что это потребуется вам. Достаточно нескольких в заданном диапазоне напряжений, который можно получить тоже с помощью нескольких предварительных замеров.

Не помню, приходилось ли мне проводить подобные испытания, почти уверен, что если и приходилось, то использовался подходящий прибор для получения характеристик, но, сознаюсь, что с большим интересом проделал все, о чем говорил, перебрал с десяток транзисторов из библиотеки компонентов программы Qucs, добавлял вольтметр для измерения напряжения на коллекторе и смотрел, как оно зависит от напряжения база-эмиттер, менял температуру транзистора...

Очень интересно.

Частотные характеристики усилителя

Выше я говорил об амплитудно-частотной характеристике усилителя. Это достаточно важный параметр для любого усилителя. В этом смысле усилители можно разделить на несколько групп: широкополосные усилители, полосовые, избирательные.

Что такое АЧХ? Это зависимость амплитуды выходного сигнала от частоты при неизменной амплитуде входного сигнала. В своих примерах я использую усилители низкой частоты (или звуковые), поскольку с ними любители сталкиваются чаще, но все сказанное можно отнести к любым усилителям. На практике снимать амплитудно-частотную характеристику приходится не так часто. Но при этом, не следует забывать, что это важный параметр успеха или неудачи в вашей работе.

Что может повлиять на амплитудно-частотную характеристику простого транзисторного усилителя на одном транзисторе, включенным с общим эмиттером? Когда я рассказывал о том, что такое транзистор, я говорил, что p-n переход — это нечто подобное конденсатору. Посмотрим, как выглядит амплитудно-частотная характеристика простой RC цепи.

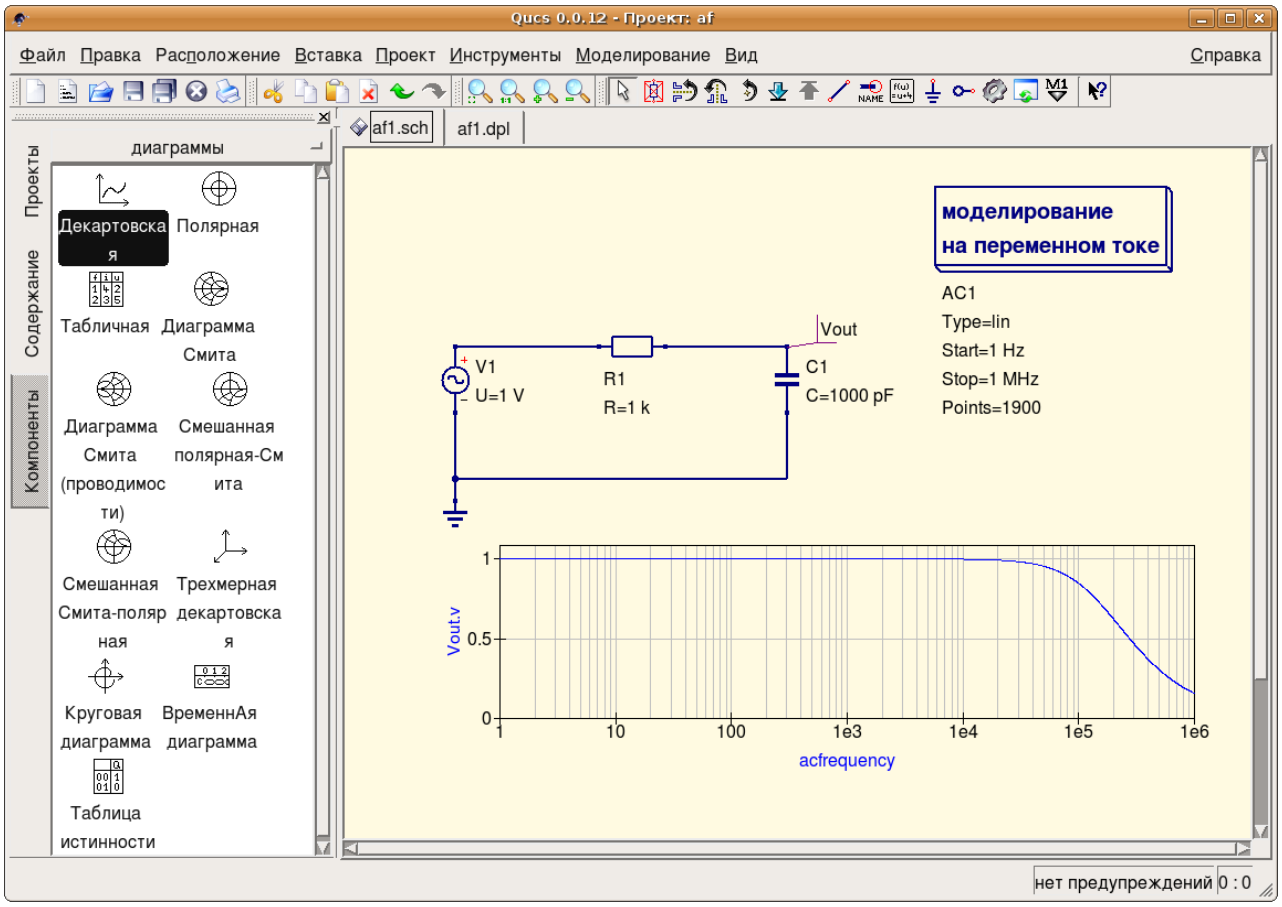


Рис. 7.21. Амплитудно-частотная характеристика RC цепи

И для сравнения я приведу АЧХ простого усилителя в той же полосе частот.

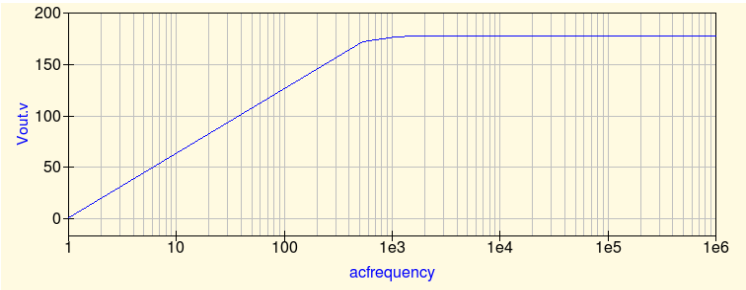


Рис. 7.22. Амплитудно-частотная характеристика каскада на транзисторе

Обе характеристики имеют область частот, где амплитуда сигнала на выходе остается постоянной. И обе имеют области частот, где амплитуда выходного сигнала меняется. Во втором случае это начальная область частот, где изменение амплитуды обусловлено конденсатором 1 мкФ, включенным на входе усилителя. Это конденсатор изменил характер зависимости амплитуды сигнала от частоты. Транзистор, который я использовал для построения каскада усиления, идеальный. И если бы ни конденсатор на входе, АЧХ была бы плоской линией «от края и до края». Что изменится, если я заменю идеальный транзистор реальной моделью?

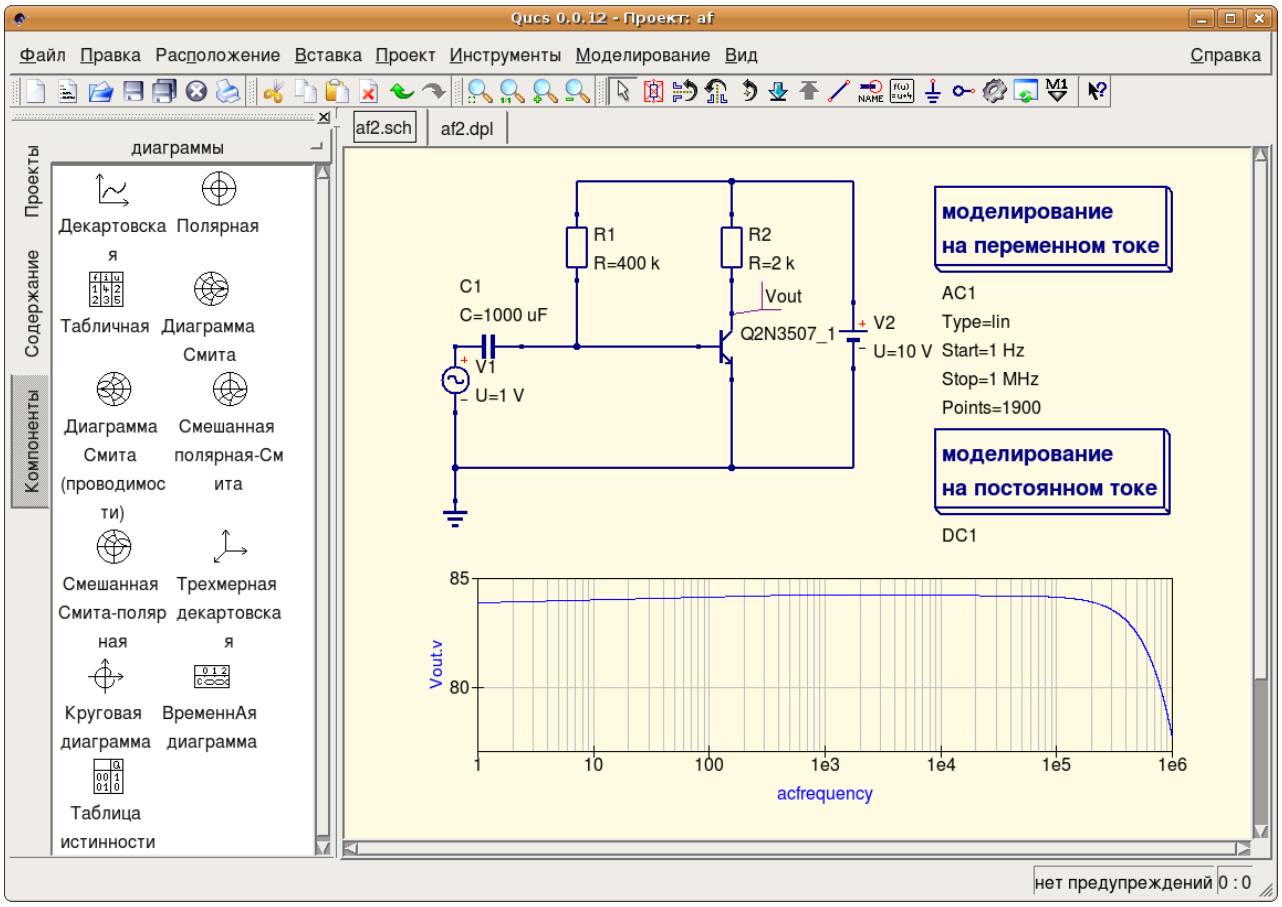


Рис. 7.23. Амплитудно-частотная характеристика с транзистором 2N3507

Я увеличил конденсатор на входе до величины 1000 мкФ, чтобы сделать амплитудно-частотные характеристики RC цепи и транзисторного усилителя похожими на начальном участке характеристики. Теперь они обе в районе частоты 100 кГц (1e5 на графиках) начинают спадать. То есть, реальный транзистор, в отличие от идеального, можно представить как идеальный с добавленной внутри транзистора RC цепью. RC цепь вида, изображенного на рисунке 7.21, называется интегрирующей, а амплитудно-частотная характеристика такой цепи имеет частоту среза. За такую частоту на практике принимают частоту, где амплитудно-частотная характеристика спадает на 3 дБ. Вернемся к графику рисунка 7.2.1, который я несколько переделал: расширил диапазон частот и применил логарифмический вид графика для напряжения, а не только для частоты.

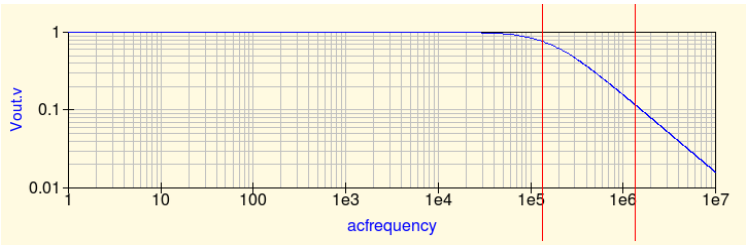


Рис. 7.24. Характер спада АЧХ за частотой среза RC цепи

Двумя вертикальными линиями я отметил участок АЧХ за частотой среза, где частота изменяется в 10 раз. В 10 раз изменяется и отношение напряжений. Если выразить это отношение напряжений в децибелах, для напряжения это 20log(U1/U2), то принято говорить,

что за частотой среза напряжение спадает со скоростью 20 дБ на декаду. Добавим в этом эксперименте (с RC цепью) еще одну RC цепь с другой частотой среза.

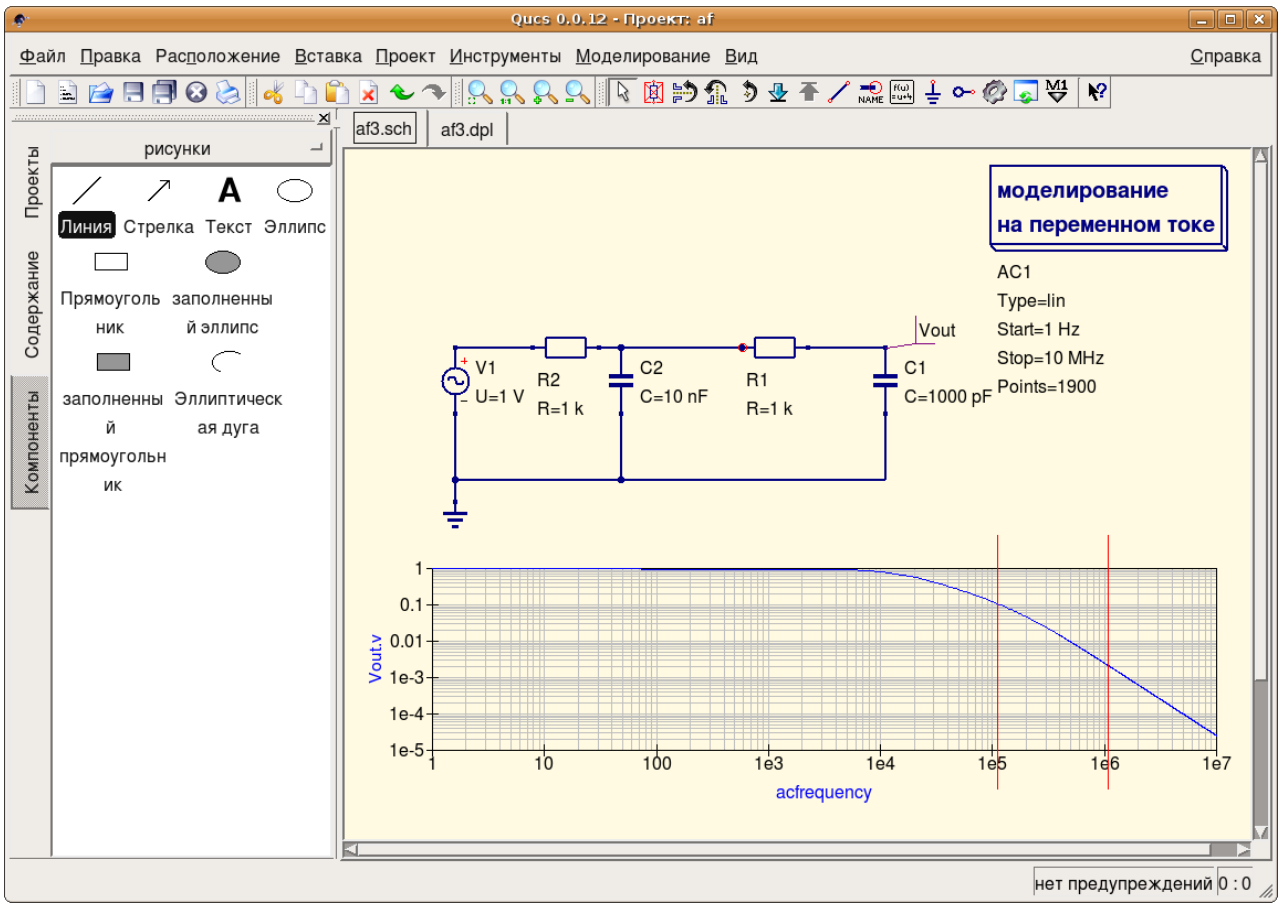


Рис. 7.25. АЧХ двух RC цепей, включенных последовательно

Две вертикальные линии отмечают декаду на оси частот, при этом отношение напряжений, это видно на графике, равно 100 или 40 дБ. Следовательно, две RC цепи с разными частотами среза дают АЧХ, частично спадающую со скоростью 40 дБ на декаду. Кроме изменения амплитуды сигнала на частоте среза и более высоких частотах меняется фаза сигнала относительно входного. График такой зависимости называется фазо-частотной характеристикой. Но как ее получить в программе Qucs, я не знаю. Есть два способа «сжульничать» на этот раз: нарисовать эту кривую вручную, но я плохой рисовальщик, или использовать другую программу, что мне гораздо проще. Я раньше получал подобные графики в программе Multisim с помощью плоттера Боде, думаю, проще отыскать их в архиве и использовать для демонстрации.

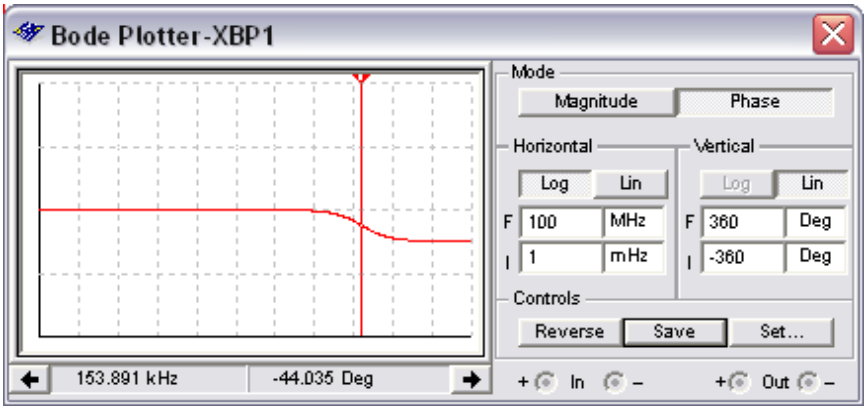


Рис. 7.26. Фазочастотная характеристика одной RC цепи

На частоте среза около 153 кГц фаза выходного сигнала изменилась на 45 градусов, она будет продолжать меняться до 90^0. И такая же фазо-частотная характеристика для двух RC цепей.

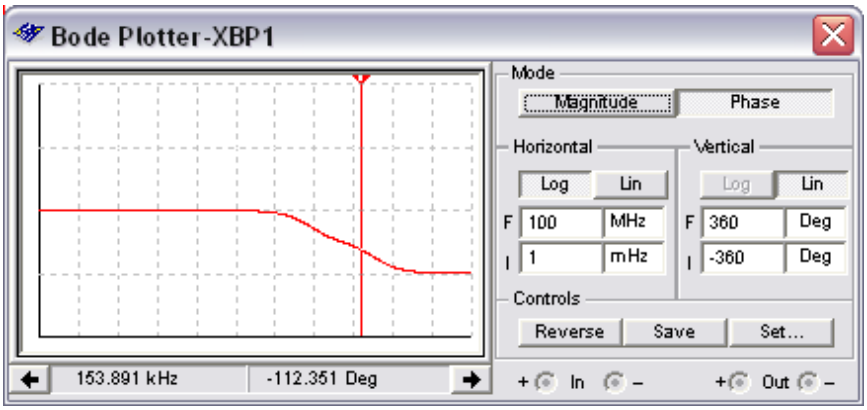


Рис. 7.27. Фазо-частотная характеристика для двух RC цепей

Значение сопротивлений и конденсаторов я брал в точности те же, что и раньше. С двумя RC цепями сдвиг фаз еще больше, а затем он завершится на некоторой частоте, где станет равен 180^0.

К чему я это все? По аналогии между однокаскадным усилителем и одиночной RC цепью двухкаскадный усилитель будет похож по своим частотным свойствам на двойную RC цепь. То есть, будет некоторая частота, на который фазовая характеристика «повернется» на 180^0. Казалось бы, и что? А то, что отрицательная обратная связь, если ею охвачены оба каскада усиления, а это делается достаточно часто, отрицательная обратная связь на этой частоте превратится в положительную. А двухкаскадный усилитель, если на этой частоте его усиление больше единицы, превратится в генератор. Генератор заказывали?

Особенно это относится к усилителям мощности звуковой частоты. Это прекрасная область приложения сил для начинающих в том, что касается электроники, но усилители мощности, как правило, имеют три каскада усиления: входной каскад, предоконечный и оконечный. Проблема устойчивости в таком усилителе, традиционно охваченном общей петлей обратной связи от выхода ко входу, становится весьма актуальна.

Давайте посмотрим, как влияет замена транзисторов на частотные свойства однокаскадного усилителя. Используем схему рисунка 7.23 в качестве базовой, будем выбирать из библиотеки компонентов программы Qucs разные транзисторы и посмотрим, как

меняется частота среза, и не будем забывать, что на частоте среза фаза меняется на 45^0. Затем, проведя эти эксперименты, попробуем оценить разные схемы включения транзистора: с общим эмиттером, с общим коллектором, с общей базой, — с точки зрения частотных свойств усилителя.

Выбирая транзисторы для схемы из библиотеки компонентов обращайте внимание на верхнюю граничную частоту транзистора. А если будете проводить эксперименты на макетной плате, загляните в справочник по транзисторам, верхняя граничная частота довольно часто фигурирует в качестве справочного параметра.

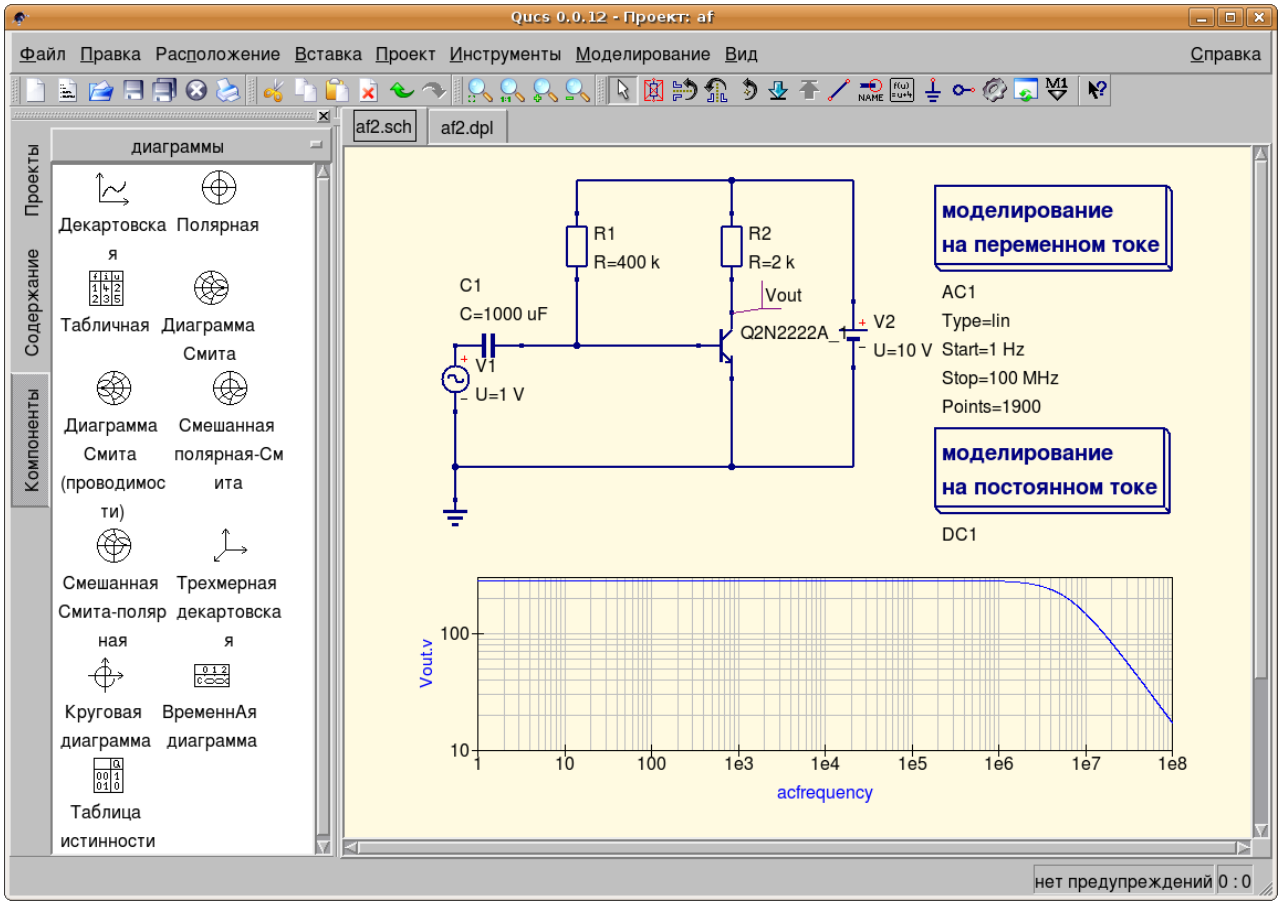


Рис. 7.28. Амплитудно-частотная характеристика каскада с транзистором 2N2222A

Транзистор 2N2222A имеет граничную частоту 300 МГц. А транзистор BC237BP имеет граничную частоту 150 МГц.

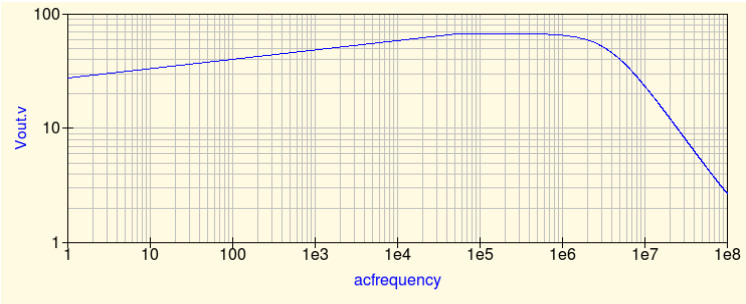


Рис. 7.29. Амплитудно-частотная характеристика каскада с транзистором BC237BP

А теперь проверим работу транзистора 2N2222A при разных вариантах включения транзистора. Первая схема, как на рисунке 7.5, схема с общей базой. Изменилось значение резистора нагрузки и напряжение питания, увеличилась емкость конденсатора на входе усилителя, все остальное осталось без изменений.

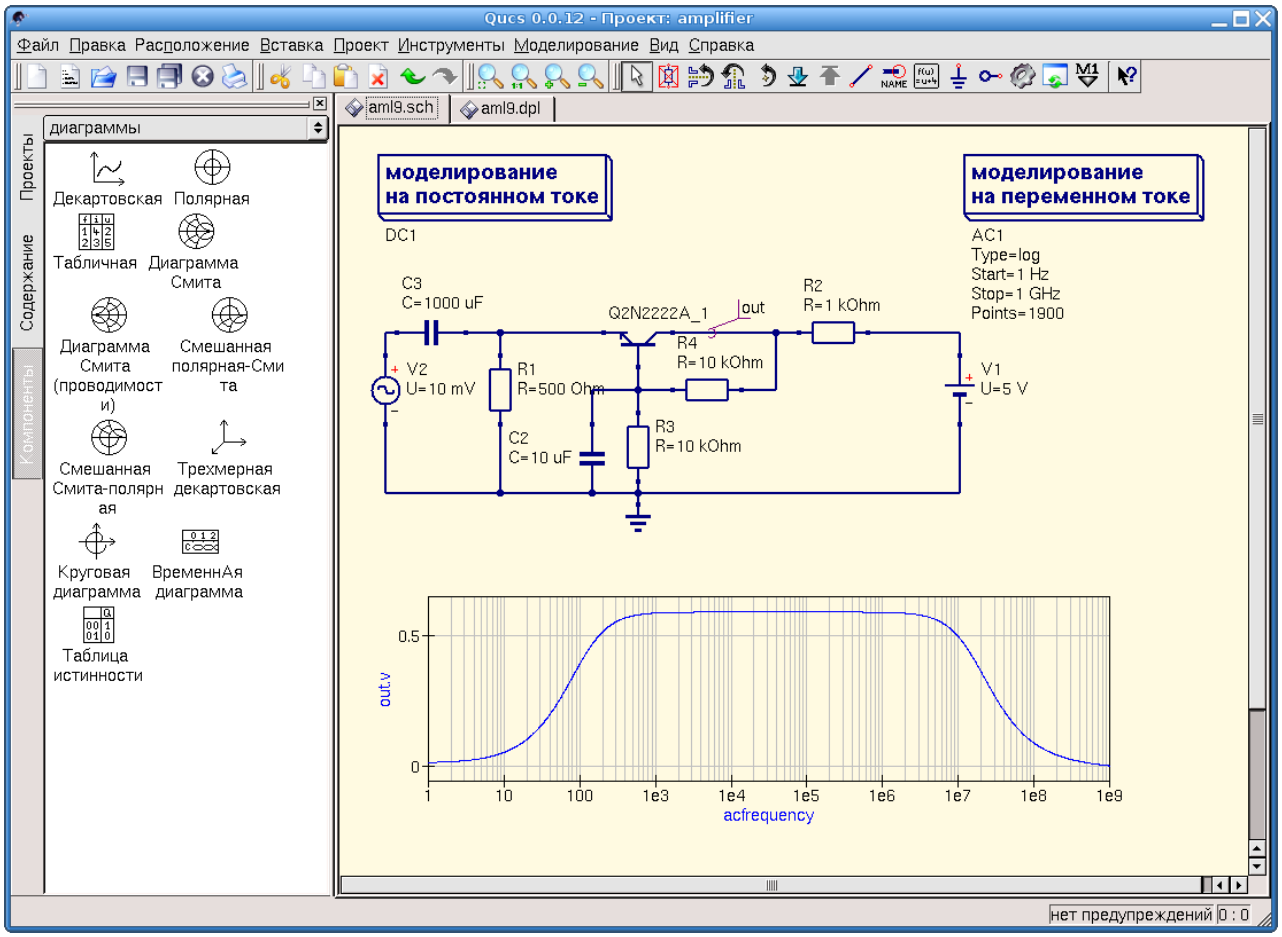


Рис. 7.30. АЧХ усилителя с общей базой

Не меняя ничего, кроме резистора R2, новое значение 100 Ом, и напряжения питания, стало 3 В, мы получим следующую амплитудно-частотную характеристику.

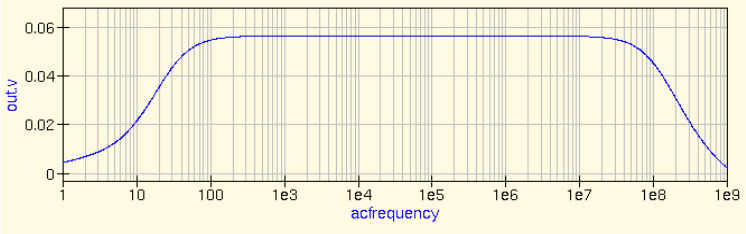


Рис. 7.31. АЧХ усилителя с общей базой при уменьшении сопротивления нагрузки

Очень интересно, если сравнить обе характеристики, то заметно, что частота среза с 10 МГц сместилась к 100 МГц.

Похожие исследования проведем для схемы с общим коллектором, взяв за основу схему на рисунке 7.6.

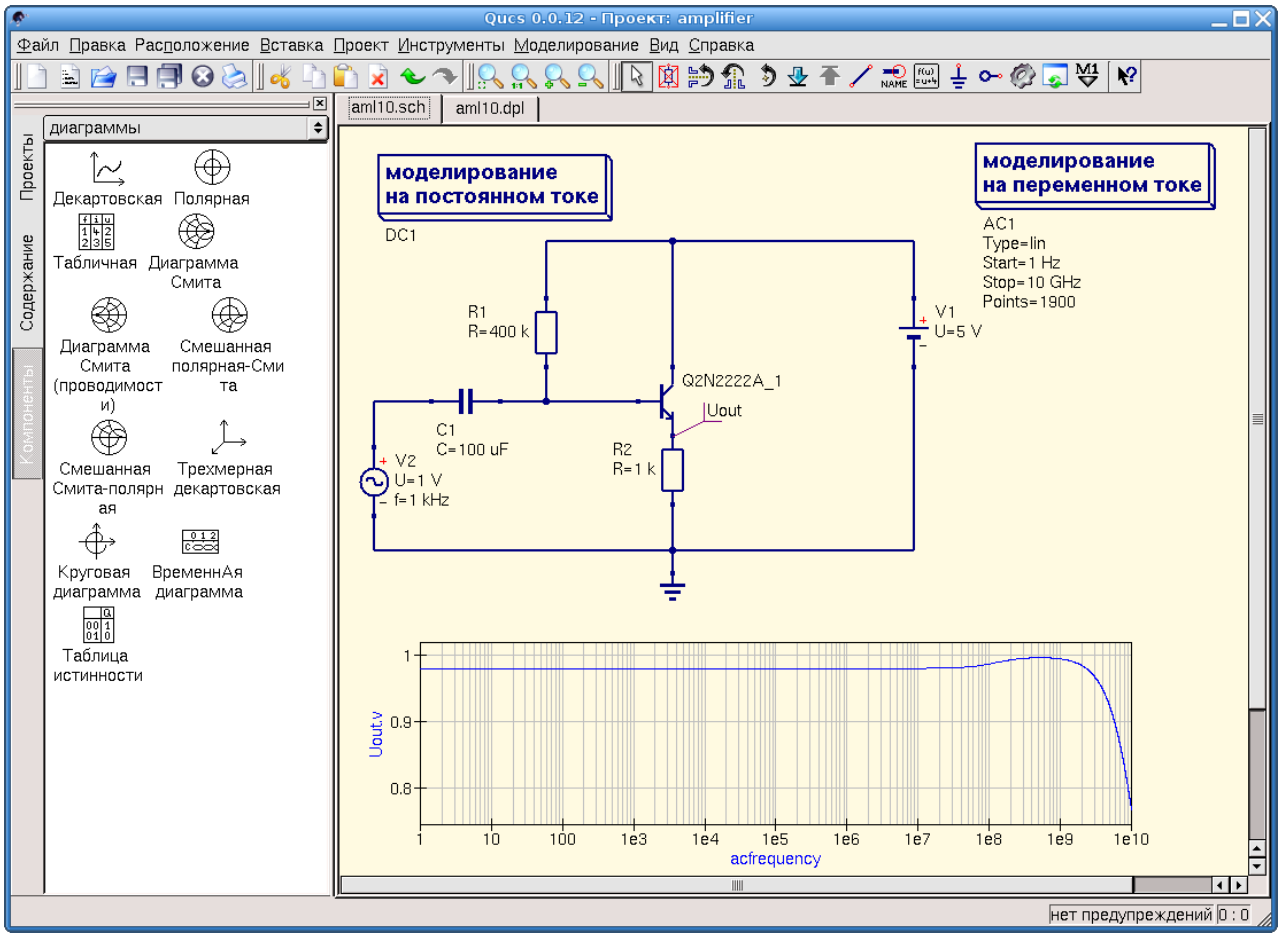


Рис. 7.32. АЧХ усилителя с общим коллектором

Аналогично предыдущему эксперименту уменьшим сопротивление нагрузки R2 до 100 Ом, а напряжение питания до 3 В.

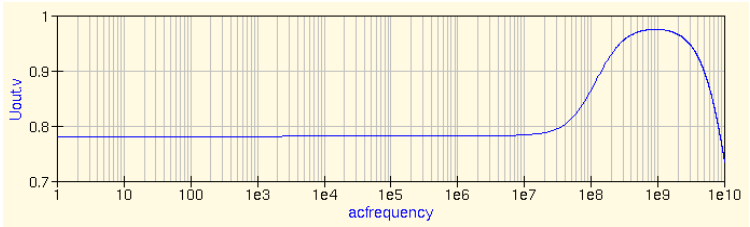


Рис. 7.33. АЧХ при изменении сопротивления нагрузки

А теперь повторим все для простой схемы с общим эмиттером. Начнем с сопротивления нагрузки в 1 кОм и напряжения источника питания 5 В. Второй эксперимент проведем с резистором 100 Ом и питанием 3 В.

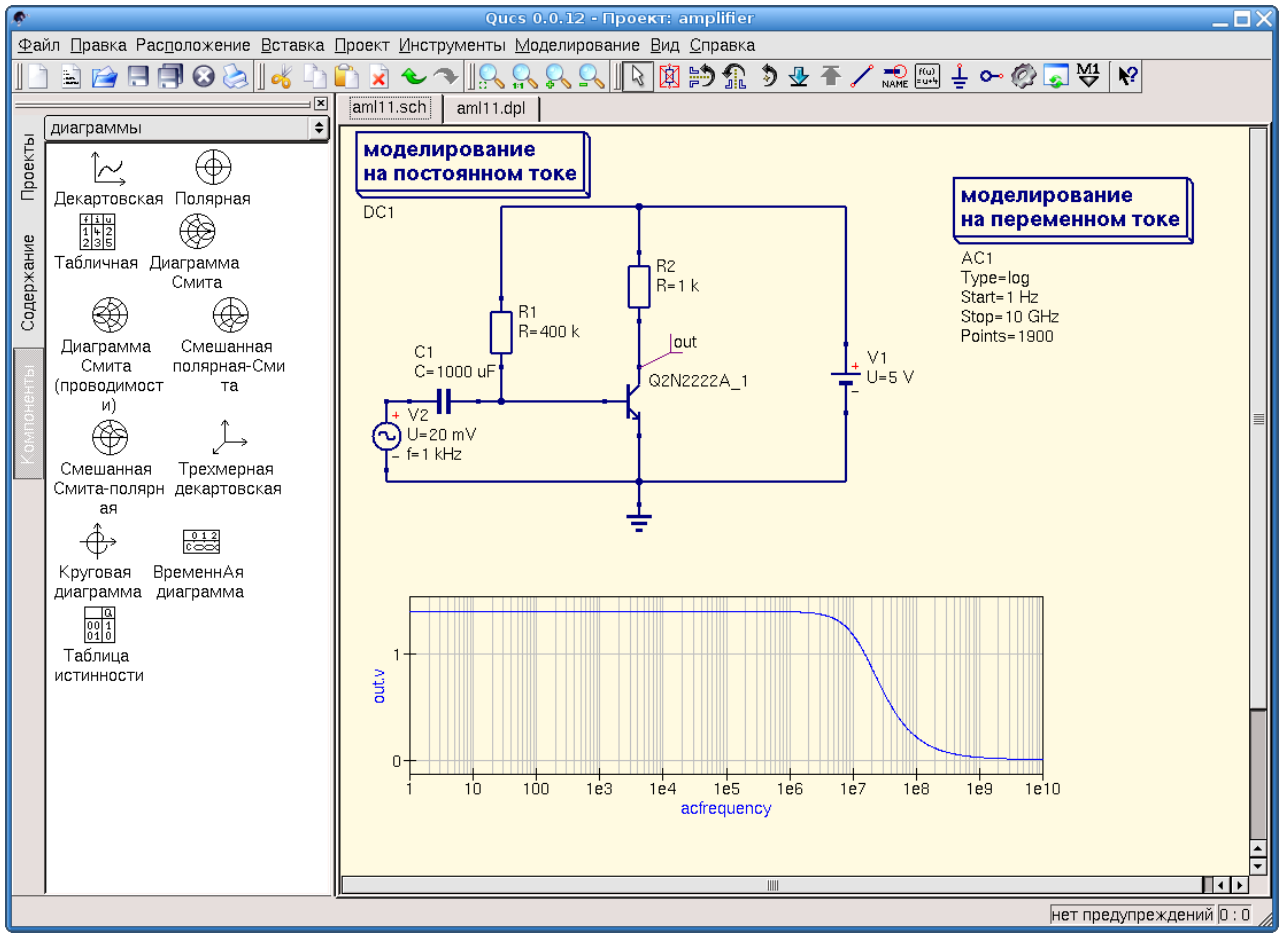


Рис. 7.34. АЧХ каскада с общим эмиттером

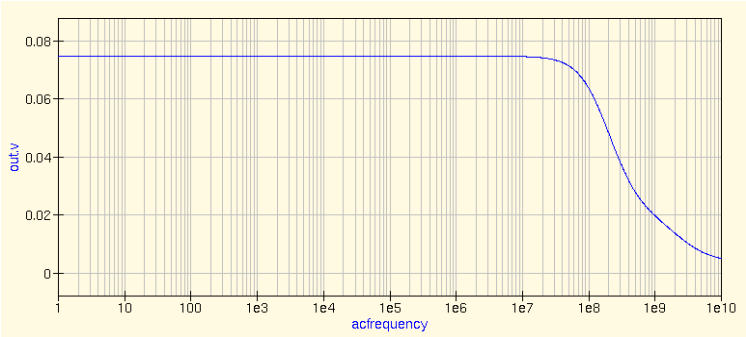


Рис. 7.35. АЧХ при нагрузке 100 Ом

Стоп, стоп, стоп!

Я вообще не собирался рассказывать о способах включения транзистора, о задании рабочей точки, поскольку со студенческой поры мне не довелось уделять много времени этим вопросам. Но, каюсь, увлекся возможностями поэкспериментировать в программе Qucs. А есть и еще один, из той же серии, аспект, касающихся усилителей, точнее выбора транзисторов для усилителя.

В справочной литературе по транзисторам можно встретить параметры, которые относятся к представлению транзистора, как четырехполюсника. Всего несколько слов об этом. И,

пожалуй, несколько слов о многокаскадных усилителях. И... вот, вот опять я подумал, что можно было бы еще столько экспериментов проделать в этой области! Конечно, можно. Рассказу о проектировании усилителей можно посвятить несколько книг...

Так, что же такое транзистор, как четырехполюсник? Это модель транзистора с двумя входными и двумя выходными выводами. На входе такого четырехполюсника действуют напряжение U1 и ток I1, а на выходе U2 и I2. Мы можем, например, выразить напряжения через токи записав уравнения:

$U1 = f1(I1,I2)$

$U2 = f2(I1,I2)$

Вводя определенные ограничения: малый сигнал, при котором можно пренебречь нелинейностью в рабочей точке, и низкие рабочие частоты – мы можем записать уравнения для переменных напряжений в виде

$u1 = a_{11}*i1 + a_{12}*i2$

$u2 = a_{21}*i1 + a_{22}*i2$

Получается система двух уравнений с двумя неизвестными. Если мы знаем коэффициенты a_{ij}, они в данном случае имеют размерность сопротивлений, то всегда можем найти интересующие нас сигналы. Коэффициенты с размерностью сопротивлений можно найти по статическим характеристикам транзистора. Так $a_{11} = \Delta U1/\Delta I1$, $a_{12} = \Delta U1/\Delta I2$ и т.д. Не хочу продолжать этот математический экскурс в теорию, но добавлю, что мы можем выбирать в качестве искомых величин не напряжения, а токи, считая напряжения заданными, мы можем выбрать пару из тока и напряжения, считая вторую пару заданной, и в любом случае мы получим уравнения схожие с приведенными выше, у которых будут меняться коэффициенты, соответственно речь будет идти об r-параметрах или h-параметрах. Мы можем изобразить эквивалентную схему четырехполюсника, используя генераторы тока и напряжения, сопротивления, и исследовать полученную модель. Мало того, мы можем придать подобной модели более физический вид, получая такие параметры как сопротивление эмиттера или базы через, например, h-параметры. Можно добавить зависимость всех параметров от частоты и, работая с комплексными величинами, получить еще больше пользы от модели, но это выходит за рамки рассказа. Если у вас есть желание углубиться в изучение свойств подобных моделей, самое лучшее, что я могу посоветовать, это выбрать подходящий учебник и попытаться использовать программу Qucs или другую, быть может более удобную, для изучения этого вопроса. Мне хотелось только напомнить вам, что среди справочных параметров для любого транзистора могут быть параметры, относящиеся к подобным моделям.

Более важным вопросом, как мне кажется, для практической работы со схемами был бы вопрос о многокаскадных усилителях. Например, тот же усилитель мощности звуковой частоты имеет, как правило, не меньше трех каскадов. Как мы выяснили, каждый каскад имеет свою граничную частоту, за которой усиление падает со скоростью 20 дБ на октаву, а фазовая характеристика на частоте среза меняется на 45 градусов. Суммарная характеристика трех каскадов будет образована всеми этими характеристиками в совокупности. При введении общей отрицательной обратной связи, а это тоже почти общая практика, возникает вопрос об устойчивости усилителя с обратной связью, поскольку ясно, что есть частота, на которой отрицательная обратная связь превращается в положительную, а усилитель, если его коэффициент усиления по напряжению на этой частоте больше единицы, может превратиться в генератор.

В предыдущих экспериментах, меняя нагрузочное сопротивление, я заметил, что меняется

полоса пропускания усилителя, что, как мне кажется, в значительной мере связано с усилением. Отрицательная обратная связь тоже уменьшает усиление каскада, но расширяет полосу пропускания. К моему сожалению, от недопонимания или неумения, мне не удалось получить быстрый результат в программе Qucs, который проиллюстрировали бы мой рассказ. Так что с этого места я прибегну к пояснениям «на пальцах».

Возьмем два каскада усиления на транзисторах T1 и T2. Это идеальные модели транзисторов, в свойствах которых я изменяю параметр, обозначенный как *Cjc* – емкость коллекторного перехода. Для первого он пусть будет равен $8*10^{-10}$, а для второго 10^{-8}. С помощью этих параметров я хочу задать разные частоты среза для первого и второго каскадов – верхние граничные частоты усиления. Кроме того, я добавлю несколько уравнений с тем, чтобы выразить усиление в децибелах. Здесь требуются пояснения. Я не нашел возможности использовать десятичный логарифм и воспользовался натуральным, отчего коэффициент получился не 20, а 8.6. Используя напряжение генераторов 1 В, я избавился от необходимости делить выходное напряжение на входное, но деление на 1 В подразумевается.

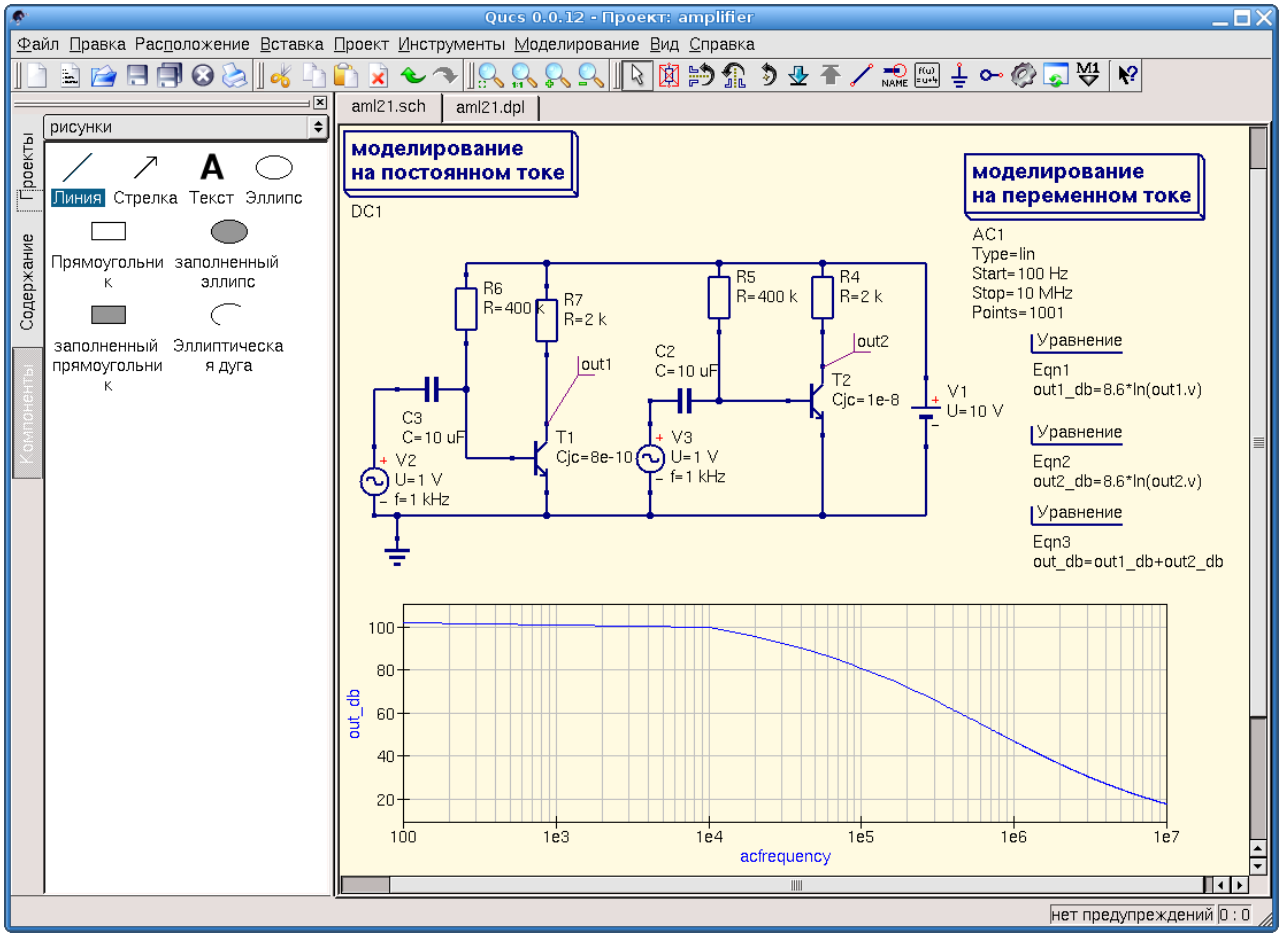


Рис. 7.36. Схема эксперимента с двумя каскадами усиления

На диаграмме приведена результирующая АЧХ двух каскадов. Я сделаю вид, что получил ее естественным образом, а вы хотите верьте мне, хотите, нет.

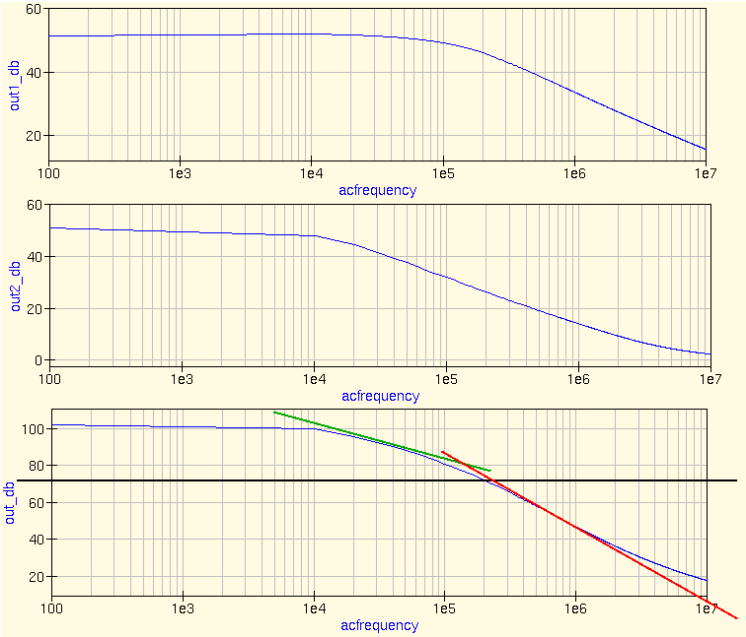


Рис. 7.37. Амплитудно-частотные характеристики двухкаскадного усилителя

Частота среза, как видно из диаграмм, первого каскада в районе 100 кГц, второго 10 кГц, и для каждого из каскадов амплитудно-частотная характеристика за верхней граничной частотой спадает со скоростью 20 дБ/декада. Все то, о чем мы говорили раньше. Но на результирующей АЧХ, если посмотреть в район 100 кГц, то видно, как после 100 кГц скорость изменения АЧХ меняется, становясь равной 40 дБ на декаду. Чтобы это подчеркнуть, я нарисовал касательные к графику частотной характеристики.

Амплитудно-частотная характеристика после введения общей петли отрицательной обратной связи будет выглядеть так, как если бы ее срезали, как показано на нижней диаграмме на уровне 70 дБ, линией параллельной оси x. При этом сама линия пересекается с характеристикой без обратной связи в том месте, где она спадает со скоростью 40 дБ, образуя новую частоту среза усилителя (на диаграмме это около 200 кГц, 2e5). Если бы пересечение было выше (пресекаясь при спаде 20 дБ/декада), мы могли бы не беспокоиться об устойчивости усилителя после введения отрицательной обратной связи. Но сейчас мы должны проверить, будет ли наш усилитель устойчиво работать после введения общей петли отрицательной обратной связи. Для этого можно было бы построить фазо-частотную характеристику усилителя без обратной связи и посмотреть угол сдвига фазы на новой частоте среза с обратной связью. На практике, имея генератор и вольтметр, строить фазо-частотную характеристику усилителя слишком сложно (я бы не взялся), поэтому используют амплитудно-частотную характеристику, снимаемую после введения обратной связи на частотах близких к полученной граничной частоте усилителя. Она, как правило, имеет характерный выброс. По величине этого выброса и решают вопрос об устойчивости усилителя. Она определяемой запасом фазы, который желательно иметь не менее $20\text{-}25^0$, но нет смысла делать запас больше $50\text{-}70^0$. При запасе фазы 45^0 АЧХ усилителя получается без подъема, при запасе 25^0 подъем характеристики около 6 дБ, дальнейшее увеличение подъема свидетельствует о неустойчивости усилителя с введенной ООС. Характер получающейся АЧХ можно проиллюстрировать следующей диаграммой.

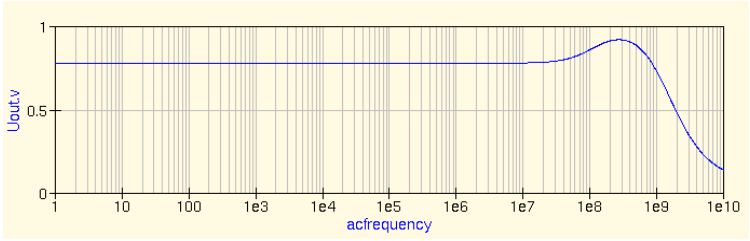


Рис. 7.38. АЧХ усилителя после введения ООС

Предварительные усилители звуковой частоты, а особенно усилители радиочастот, редко охватывают общей петлей обратной связи, поэтому для них все выше изложенное не так актуально, но все что касается амплитудно-частотных характеристик, фазо-частотных характеристик усилителей, линейных моделей – все это остается в силе. Я бы не советовал начинающему радиолюбителю увлекаться высокими частотами в плане изучения усилителей. На высоких частотах, на сверхвысоких частотах интенсивно работают многие параметры, трудно поддающиеся измерению, например, паразитные емкости и индуктивности. Эксперименты на этих частотах требуют соответствующей измерительной аппаратуры и определенных подходов при измерении, но кроме знания и понимания своеобразия усилителей этих частот, эксперименты не дадут ничего принципиально нового. Лучше отложить освоение высоких частот на тот период, когда все станет ясно и понятно с низкими частотами. Если вас интересуют только радиоприемники и радиопередатчики, то осваивать эту область предпочтительней, повторяя готовые и проверенные схемы, или используя программные возможности.

Чтобы закончить обзор низких, то есть, звуковых частот, я хочу немного рассказать об электроакустике. Я не специализируюсь в этой области, но усилители звуковых частот никому не нужны сами по себе. Поэтому следующая глава будет кратким рассказом о том, к чему «прицепляется» в качестве паровоза усилитель мощности звуковых частот.

Глава 8. Бабахать или нет – вот в чем вопрос

Если вы начинающий радиолюбитель или начинающий электронщик, а не начинающий хулиган, то мои рассуждения об умеренности в части выбора мощности должны прозвучать для вас, если не убедительно, то приемлемо в качестве информации к размышлениям.

Разговоры, разговоры

Многие электронные устройства в своем составе имеют низкочастотный тракт, заканчивающийся усилителем мощности, работающим на громкоговоритель. Последний может быть просто излучающей головкой, может быть многополосным, может быть специальным устройством для выполнения специальных задач. Есть, по меньшей мере, несколько способов преобразовать электрический сигнал в звук. Уверен, вы все знаете, что звук – это волновое изменение давления, например, в воздухе, распространяющееся в виде уплотнений и разряжений. Если подобная волна изменяется в пространстве по закону синуса, то мы слышим чистый тон. Для создания таких волн наиболее часто употребляется электродинамический громкоговоритель (я надеюсь, вы простите мне, что я порой буду называть громкоговорителем и излучающую головку, и акустическую систему).

Громкоговоритель, напомню, это электромеханическое устройство для преобразования электрического сигнала в звук. Сейчас я буду рисовать громкоговоритель. С моими художественными задатками это зрелище не для слабонервных...

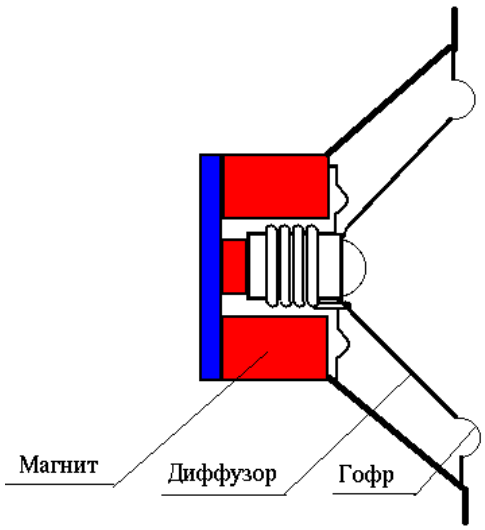


Рис. 8.1. Конструкция громкоговорителя

Диффузор громкоговорителя прикреплен к цилиндру, на который намотана катушка, помещенная в поле постоянного магнита. Магнит имеет внешнее кольцо и внутренний, входящий в катушку стержень. Таким образом, катушка оказывается в постоянном магнитном поле. Чтобы катушка сохраняла свое положение, она центрируется специальной шайбой, прикрепленной к ее верхней части в месте соединения с диффузором. Сам конусообразный диффузор прикреплен к диффузородержателю с помощью гофра. В современных громкоговорителях гофр уже не гофр, а выпуклое кольцо (как на рисунке) из синтетической резины или каучука. Центрирующую шайбу тоже делают гофрированной для облегчения перемещений диффузора и катушки вдоль оси громкоговорителя, но так, чтобы избежать

смещений в поперечном направлении. Диффузор должен быть легким, но жестким, чтобы он двигался как единое целое. Это удается сделать только в некоторой области звуковых частот, где диффузор работает, как поршень, например, в диапазоне 60-200 Гц. На более высоких частотах внутренняя часть диффузора начинает обгонять в своем движении внешнюю, а диффузор движется подобно одеялу, с которого стряхивают крошки после утреннего завтрака в постели. Эффективность работы всей площади диффузора уменьшается, поэтому кроме широкополосных делают узкополосные громкоговорители: среднечастотные и высокочастотные громкоговорители имеют, как правило, диффузоры меньше.

Громкоговоритель при работе на низких частотах ведет себя по отношению к сигналу почти как активное сопротивление, исключая область резонансной частоты. Как эта резонансная частота получается и от чего она зависит?

Диффузор и катушка имеют некоторую массу, подвешенную к упругому гофру. Как любой упругий материал с прикрепленной к нему массой, диффузор, если его вывести из состояния равновесия начинает колебаться. Чем больше масса, и чем менее упругими свойствами обладает гофр, тем ниже будет резонансная частота колебаний громкоговорителя. Но чем больше масса, тем менее эффективна будет работа громкоговорителя – для создания звукового давления потребуется большая мощность, подводимая к громкоговорителю. Так что, увеличивать массу особого смысла нет. Как нет смысла слишком уменьшать гибкость подвеса диффузора, поскольку могут появиться смещения катушки не вдоль оси, а поперек оси громкоговорителя. Компромисс обычного громкоговорителя выражается в наличии резонансной частоты в области низших частот. Этот механический резонанс похож, если говорить о поведении сопротивления громкоговорителя на разных частотах, на резонанс колебательного контура.

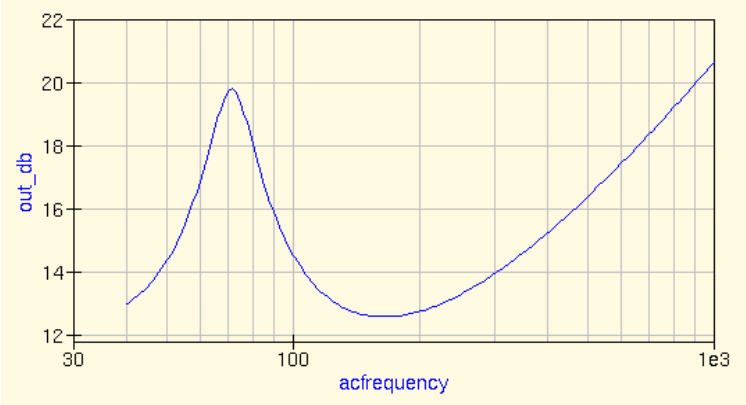


Рис. 8.2. Зависимость сопротивления громкоговорителя от частоты

Конечно, эта диаграмма, созданная в программе Qucs, не является реальной характеристикой, но хорошо поясняет характер изменения сопротивления громкоговорителя. Ниже резонансной частоты сопротивление быстро спадает, а вместе с ним и отдача громкоговорителя, на средних частотах сопротивление громкоговорителя близко к активному сопротивлению катушки, а выше начинает сказываться индуктивный характер звуковой катушки, что приводит к росту сопротивления.

Как громкоговоритель должен работать? При движении диффузора вперед он должен увеличивать давление в воздухе, а при движении назад уменьшать его. Эти изменения давления воздуха и есть звук. Но вспомним, что длина волны на частоте, скажем, 20 Гц будет примерно 17 м. То есть, диффузор при своем движении должен формировать изменения давления на всей этой дистанции вокруг себя. Обычно громкоговоритель имеет диффузор с

диаметром около 15-25 см. Если ничего не предпринимать, оставив громкоговоритель в подвешенном состоянии, то воздух пойдет кратчайшей дорогой и будет просто перетекать с передней части диффузора на его заднюю часть, а громкоговоритель будет работать как никудышный вентилятор вместо формирования протяженных изменений давления. Звука не будет. Простейшей мерой пресечения этого явления стал бы самый простой плоский экран – плоская поверхность, в которую врезан громкоговоритель. Однако размеры этого экрана для формирования давления потрясают воображение – он должен иметь стороны значительно больше длины волны. О-очень большой получается экран. Либо следует мириться с увеличением нижней граничной частоты громкоговорителя в десятки раз. Но есть еще один вариант. Если сложить наш экран в закрытую коробку, то мы полностью изолируем внешнюю поверхность диффузора от внутренней. Решим ли мы проблему? Частично, посеольку при этом воздушный объем, заключенный в закрытой коробке, станет активно сопротивляться движению диффузора. Чем больше этот объем, то есть, чем больше размеры такой коробки, или чем меньше диаметр диффузора, тем легче диффузору двигаться. Определение размеров коробки, необходимой для акустической системы с заданными параметрами, и становится основной задачей. В первую очередь в области самых низких звуковых частот, поскольку с ростом частоты эта задача упрощается тем, что на средних, а особенно высоких частотах объем коробки почти совпадает с минимальным конструктивным оформлением, призванным скорее радовать глаз, чем ухо. Но не следует думать, что там все получается само собой, что нет никаких проблем. Их хватает во всем звуковом диапазоне частот, если мы говорим о качественном звуковоспроизведении. Но этот предмет, как и любой, столь интересен сам по себе, что о нем написано множество умных книг и статей, и если вас эта область деятельности, связанная с электроникой, интересует больше остальных, то советую поискать на полках книжных магазинов, в интернет-магазинах и на компакт дисках все, что будет представлять для вас интерес по мере того, как вы будете осваивать звуковоспроизведение. Найдете много интересного и полезного.

Первоначально я не планировал обращаться к теме акустики, но мой помощник Александр спросил о моем отношении к параметрам Тиля-Смолла. Пока он не слышит, признаюсь, что об этих параметрах я услышал от него. Пришлось «побегать» по Интернету, забросив на несколько дней книгу, пришлось открыть книжный шкаф, чтобы немного переделать предыдущую главу, пришлось вспомнить те далекие дни, когда мне пришлось погрузиться в размышления и расчеты, связанные с акустикой. Мне трудно переключится с рассказа о транзисторах и схемах на звук, поэтому я начну с баек. Они не относятся к делу, но надеюсь, что это поможет мне плавно перейти к вопросу конструирования громкоговорителей.

Первая байка относится к давним временам, когда я начинал работать с микропроцессорами и компьютером. Как все, я начал увлекаться компьютерными играми. Они были не столь красивы, как сегодня, но не менее увлекательны. Но однажды мне потребовалось написать программу, чуть большую чем несколько строк в машинных кодах. И это стало началом конца моих «геймерских» начинаний. Я понял, что программировать гораздо интереснее, чем пользоваться программой.

Вторая байка относится к временам еще более давним. Создавая прототип для решения одной задачи, который, впрочем, не пригодился, я, чтобы понять, успешно ли продвигается дело, решить послушать несколько музыкальных фрагментов. Прототип был создан как квази-квадрофоническая система. Вот эти несколько фрагментов заставили меня полностью изменить мое отношение к понятию качества звуковоспроизведения.

А теперь вернемся к громкоговорителю. В электроакустике давно используется принцип электромеханических аналогий, который основан на том, что уравнения для электрических цепей и механических систем ничем не различаются с точки зрения математики. А это позволяет использовать электрические цепи, как аналоги механических систем, а

математический аппарат для расчета этих цепей использовать при анализе поведения, например, электродинамического громкоговорителя в замкнутом объеме акустического оформления.

Когда мы говорили об электрических цепях в самом начале, мы использовали не так много основных понятий. Позже, рассматривая работу усилителей, расширили количество параметров, которые могут вызывать у нас интерес. Если говорить о расчетах закрытого ящика или фазоинвертора для громкоговорителя, то есть несколько основных сущностей, определяющих дальнейший подход, и ряд параметров, необходимых для расчетов. К счастью, сегодня есть программы, которые позволяют сделать это даже любителю без опыта работы в этой области. Я выбрал одну из таких программ, но вы можете остановить свой выбор на любой из них.

Прежде, чем рассказать о программе, я хочу немного рассказать о том, о чем мне хотелось бы, чтобы вы подумали до начала своего первого проекта в этой области. Громкость звука можно охарактеризовать двумя важными величинами – порог слухового восприятия и болевой порог. Первый означает, что звук, исходящий от источника, расположенного на некотором расстоянии от человека, становится не слышен. А второй вызывает болевые ощущения. При численной оценке удобно использовать децибелы, единицы, о которых мы говорили выше. В этих единицах порогу слышимости будет соответствовать 0 дБ, а болевому порогу 120 дБ. Очень интересны с моей точки зрения несколько цифр, размещенных в этом диапазоне:

Шепот на расстоянии 1м	*25 дБ*
Шум на тихой улице	*30-35 дБ*
Спокойный разговор троих в средних размеров комнате	*45-50 дБ*
Аплодисменты в зрительном зале	*60-75 дБ*
Шум в поезде метро при движении	*85-90 дБ*
Духовой оркестр	*80-100 дБ*
Авиационный мотор на расстоянии 1 м	*110-120 дБ*

Начиная проект вы в первую очередь должны решить для себя, какой уровень шума, досаждающий вашим близким и соседям, вы намерены устроить. Сегодня есть возможность использовать громкоговорители, потребляющие 100 Вт, можно использовать усилители, способные отдавать в нагрузку мощность 100-200 Вт. Хотите ли вы использовать эти возможности, и нужны ли они вам?

Если ваша комната, в которой вы собираетесь слушать музыку, 15-20 $м^2$, то, задаваясь целью обеспечить уровень громкости 100 дБ, представьте себе, что вы поместите в этой комнате 20-50 человек оркестрантов. С их инструментами. И куда деваться вам? Я не говорю о соседях. Добавьте к этому шум с улицы в 30-40 дБ и необходимость в достижении 140 дБ, чтобы быть уверенным, что все «путем»...

Можно возразить, да я сделаю мощную систему, но буду включать ее не на полную мощность. Возражения приняты. Но, преодолевать трудности построения мощной качественной системы только для того, чтобы не использовать эту мощность... Вам тоже смешно?

Когда включен телевизор, то насколько я могу помнить, его не приходилось использовать с уровнем громкости более 10-15 процентов от номинальной. Нет нужды. А номинальная мощность его звуковой системы это всего 10 Вт.

Продолжение второй байки. Как все обычные люди, я никогда не был фанатом одного стиля или одного направления, что в литературе, что в музыке; с одинаковым удовольствием я слушаю вальсы Шопена и некоторые песни группы АББА или Модерн токинг. Никогда я не

любил тяжелый рок, и ничего с этим поделать не могу. Я не коллекционер своих пристрастий, лишь обычный собиратель. Некоторые особенности прототипа системы звуковоспроизведения волновали меня особенно: это относилось к работе акустических линз, действительно ли они расширяют стереобазу? Не самый удачный ревербератор в системе мог больше мешать, чем помогать восприятию звука. Поставив популярные в те времена записи, я слышал, в сущности, обычное «стерео», достаточно «смазливое», но обычное. И вдруг... я остановил запись, выключил и включил вновь оборудование – не помогало. На одной из записей, я ее проверил, она не была испорчена, на одной из записей, возможно, это была запись «Электрик лайт оркестра», не помню, начало музыкального фрагмента просто пропало. Оно сбилось в блеклую кучку куда-то к центру, какая там расширенная база! – оно «сгорбилось, скукожилось»... Никакие регулировки не спасали. Видимо, что-то все-таки вышло из строя, решил я, запустил запись с начала и сел «зализывать раны», вспоминать все сомнительные места... Но даже не успел «добрести» до разделительных фильтров, как накатывающий на меня блеклый ком вдруг взорвался, рассыпавшись по всей комнате фейерверком звуков. Вот так закончилась вторая байка.

Я понял, что звуковая картина – это не только частотная полоса и искажения, не только переходные качества системы, но в большей мере искусство композитора, звукорежиссера и инженеров звукозаписи. И что хорошая звуковая картина заставляет забыть о недостаточном диапазоне громкости или дешевых исходных элементах. Единственное, о чем я позже пожалел, так это о том, что мощность в десять ватт на канал не использовалась полностью, а я не сообразил задать ее меньше на первом этапе проектирования.

Продолжение первой байки. Не только программирование, но, взять хотя бы эту книгу, я уверен, что придумывать ее, писать ее, «заморачиваться» с деталями работы программы, используемой при написании книги, все это на порядок интереснее, чем читать результат. Уверен, что когда допишу книгу, не стану ее читать. Не интересно. Вот такая первая байка.

Вот такие мои рассуждения. А вы вольны к ним прислушаться или нет.

К счастью, никакого отношения к созданию громкоговорителя они не имеют. Основа расчета объема закрытой коробки или фазоинвертора базируется на применении электромеханических аналогий, на хорошо разработанных методах расчета фильтров, и сегодня даже не надо (или почти не надо) использовать калькулятор, все вычисления сделает программа, вам останется только собрать, проверить и настроить конструкцию.

Для начинающих я бы порекомендовал программу JBL SPEAKERSHOP. Программа имеет раздел *Test*, в котором можно выбрать подраздел *Loudspeaker*, чтобы провести все необходимые для расчетов измерения параметров. Каждый шаг отображается в виде картинки полностью объясняющей, что нужно сделать, а в левом окне подробная инструкция, описывающая параметры необходимого измерительного оборудования, включая рекомендации по отдельным элементам. Я не знаю, зачем нужен усилитель мощностью 100-200 Вт, но не готов оспаривать это. Думаю, если у вас нет такого усилителя, то можно использовать что-то другое. Возможно, мощность не столь важна, как напряжение, но это может быть вызвано использованием вольтметра (а не милливольтметра) при измерениях. Первые шаги, которые вы сделаете, нажимая клавишу **Continue** предназначены только для вас, чтобы вы могли сохранить в надлежащем виде несколько вариантов разработки, или для создания базы данных, включая такие параметры, как модель динамика и его серийный номер. Не пренебрегайте этой информацией, если она у вас есть.

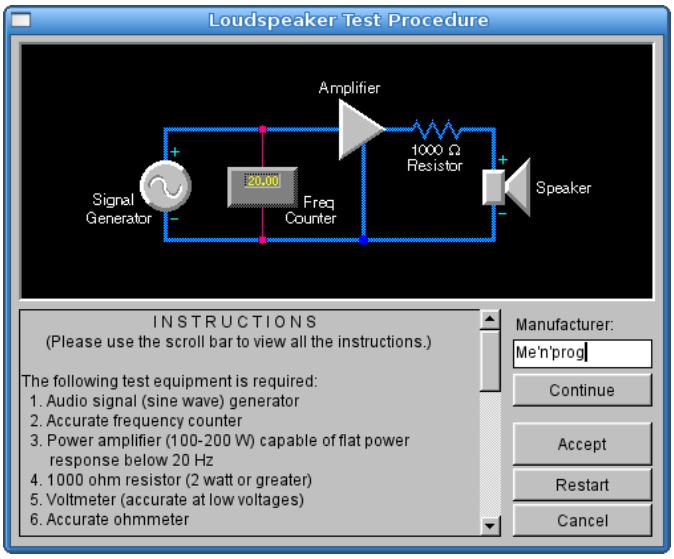


Рис. 8.3. Начало измерений параметров динамика в программе JBL

Если ваш генератор имеет хорошую шкалу или встроенный частотомер, то дополнительный частотомер вам не понадобится. Небольшая ошибка измерений, которая может возникнуть при работе, не скажется на результатах пагубным образом, больше внимания следует уделить процессу измерения – лучше проводить измерения несколько раз, записывая результаты, а затем взять среднее арифметическое от полученных данных. Порой трудно определить, достигли вы точки отсчета или нет.

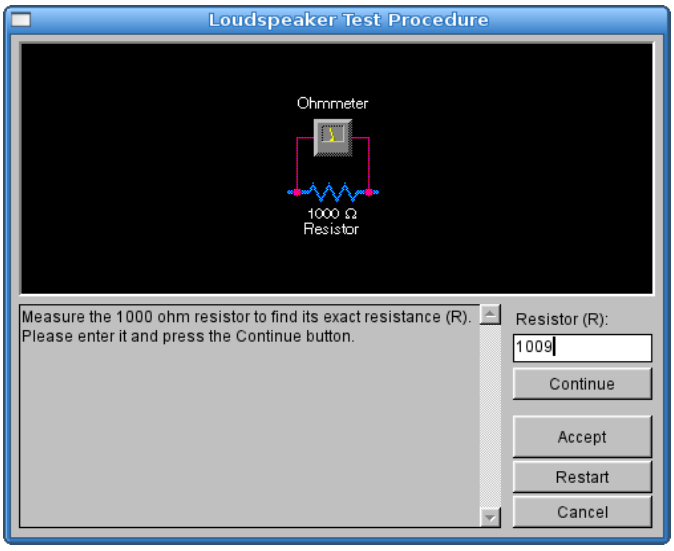


Рис. 8.4. Первый шаг в измерениях в программе JBL

Я не стал бы задерживаться на этом шаге, если бы ни одно обстоятельство. Программа, ориентированная на создание качественного громкоговорителя, выполнит необходимые вычисления, но она не сделает за вас всю работу, и если вы ошибетесь в данных, которыми снабдите ее, не ее вина, что окончательный результат, полученный вами, будет отличаться от ее выводов. Если в качестве дополнительного резистора вы используете прецизионный резистор класса точности 0.1-0.05%, то этот шаг можно пропустить, иначе не поленитесь измерить своим мультиметром, а он измеряет сопротивления с точностью, как правило, до 0.5%, тот резистор, который будете использовать в дальнейшем. Если вас интересует, как

повлияет это на результаты, можете в дальнейшем проделать в программе несколько вариантов расчетов. Возможно, достаточно 10-20% точности, но если вы приступили к измерениям, проделайте их тщательно, насколько возможно. Это же относится к последующему измерению диаметра и измерению активного сопротивления катушки. Я бываю подчас небрежен. Скорее всего, я бы измерял активное сопротивление у динамика, лежащего на столе. Но не следуйте дурным примерам, если программа советует подвесить динамик, лучше его подвесить даже при измерении активного сопротивления омметром. При следующем шаге, обратите внимание на отсутствие перегрузки генератора или усилителя, как об этом напоминает программа!

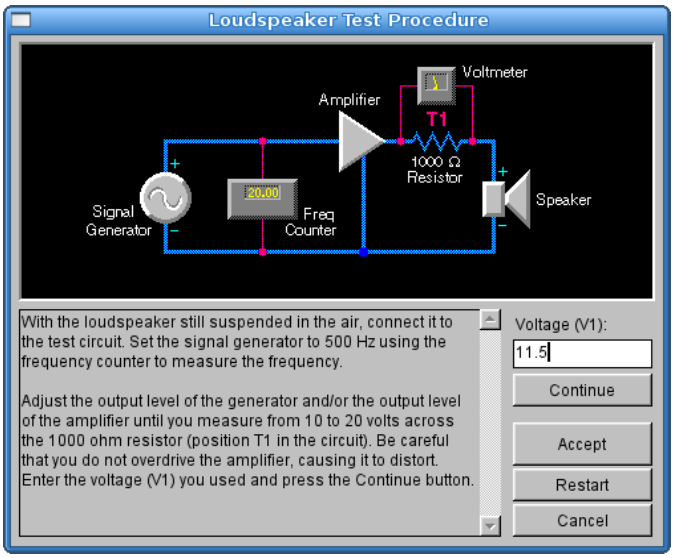


Рис. 8.5. Калибровка

В отсутствие осциллографа либо надо быть полностью уверенным в возможностях генератора (или усилителя, если он используется), либо быть готовым к появлению ошибки. Все-таки мультиметр (или вольтметр) рассчитан на точные показания при синусоидальном напряжении. Если сомнения останутся, то после окончания измерений попробуйте их повторить при задании напряжения ниже рекомендованного.

Следующий шаг, измерение собственной резонансной частоты громкоговорителя, вне сомнений следует проводить при подвесе громкоговорителя подальше от стен или больших поверхностей. Лучше, как я и говорил, повторить измерения несколько раз, взяв среднее арифметическое замеров. И не забудьте, хотя это следующий шаг в программе, каждый раз записывать показания мультиметра, подключенного к громкоговорителю, с тем, чтобы тоже взять среднее арифметическое от показаний! То же относится и к следующему шагу, хуже не будет, и не забудьте, что если вы при следующих измерениях выберете напряжение ниже рекомендованного при калибровке, то следует пропорционально уменьшить напряжение этого шага измерения, которое задается как 0.538 В. Столь высокая точность заставляет меня усомниться в том, что я был прав, советуя отступить от рекомендаций при калибровке, но попробовать стоило.

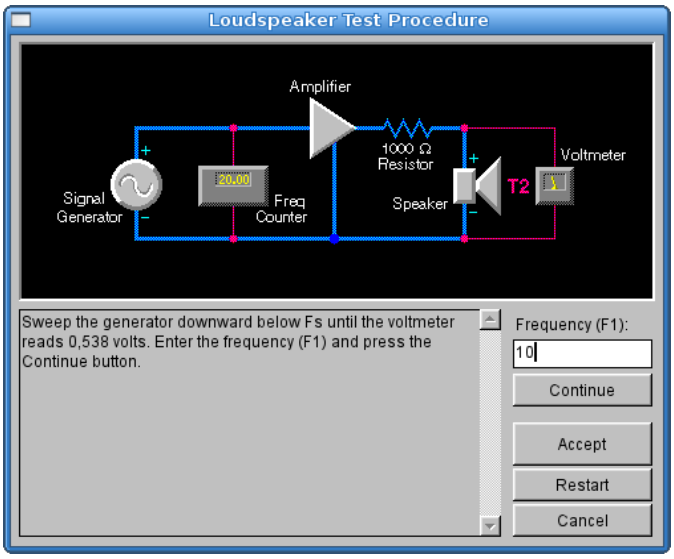


Рис. 8.6. Определение параметров резонанса

Вот и расплата за небрежность. Описывая программу, я, можете не сомневаться, не проделываю измерений, я даже не использую результатов измерений сделанных кем-то, а просто подставляю что-то из головы. И что из этого может выйти хорошего? Конечно, ничего. Я застреваю в программе, которая пытается мне тонко намекнуть, что так не бывает!

Пройдя «сквозь строй», пристыженный, я попадаю в следующую неприятность: в самом начале тестирования при описании необходимого оборудования программа давала рекомендации по объему тестовой коробки в зависимости от диаметра диффузора. Диаметр я брал «с потолка», а рекомендованный объем не записал. Пришлось запустить второй экземпляр программы (вернуться к началу) и, предполагая, что диаметр я брал где-то между 13 см и 20 см, выбрать число между 3.5 и 15 литрами.

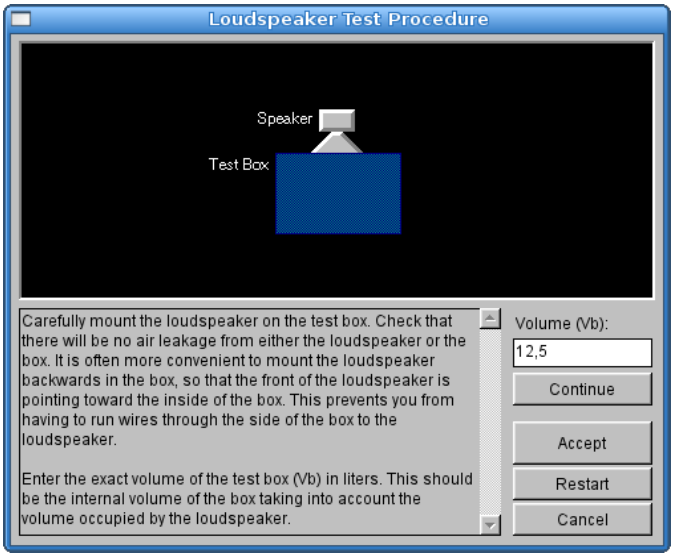


Рис. 8.7. Измерения с проверочным коробом

Добавляя этот параметр учтите, что следует добавить или вычесть, это как вы закрепили динамик, его объем. Думаю, можно для варианта показанного на рисунке взять объем конуса, образованного диффузором. И обязательно промажьте все щели пластилином при

проведении теста. Для эксперимента можно взять, если они еще есть, подходящий по объему ящик для отправки почтовых посылок. Если вас смущает жесткость стенок, укрепите их, но не думаю, что это так сильно скажется на результатах, хотя интересно было бы проверить... Проверяя новую резонансную частоту и параметры резонанса, не забывайте, если делаете несколько замеров, записывать напряжение, показываемое вольтметром! Пригодится, чтобы не возвращаться к этому. Заканчиваются ваши предварительные «мучения» с появлением таблицы всех параметров вашего громкоговорителя.

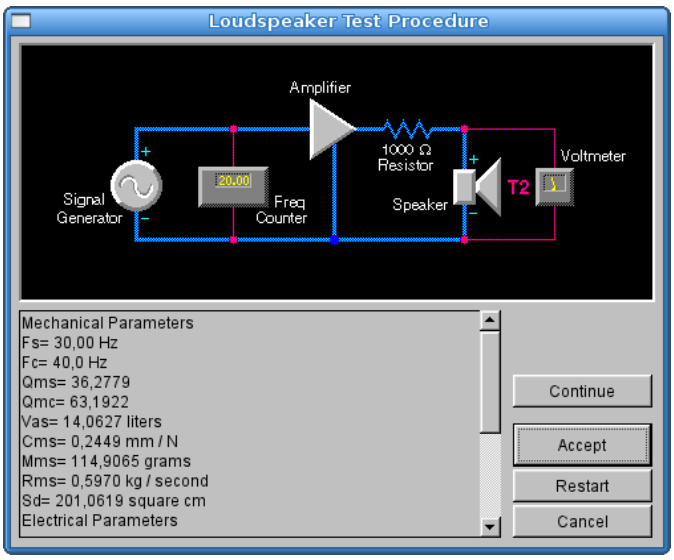


Рис. 8.8. Сводные параметры громкоговорителя после его испытаний

Программа предлагает один из вариантов получения этих важных для дальнейшего параметров. В Интернете можно найти, во всяком случае я нашел, несколько статей, посвященных расчетам громкоговорителей с использованием параметров Тиля-Смолла. В частности, вместо испытания громкоговорителя в проверочной коробке можно провести измерения с дополнительными грузиками. Возможно такой вариант для вас будет предпочтительней. Очень интересную статью я обнаружил по экспериментам с самой программой (не уверен, что JBL), где менялись параметры громкоговорителя и анализировались результаты. Очень полезные эксперименты.

А вы в качестве вознаграждения можете посмотреть, что у вас получится в оптимальном варианте при использовании, например, фазоинвертора (верхняя кривая) или закрытого ящика.

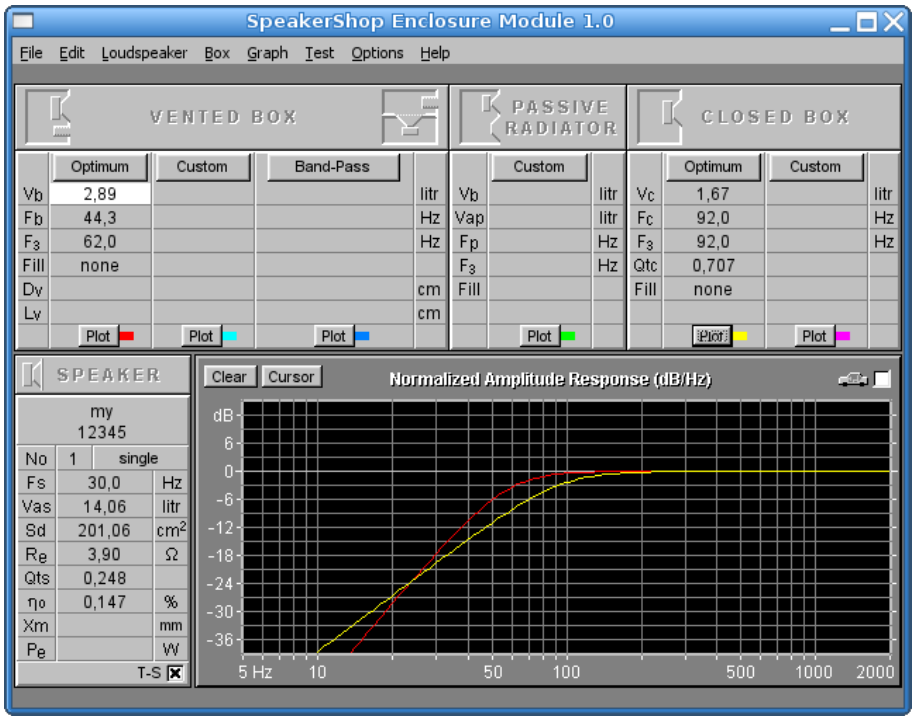


Рис. 8.9. Результаты расчетов в программе JBL

Все данные для громкоговорителя я брал «с потолка», но полученные результаты вполне подходят для продолжения разговора. Исходная резонансная частота громкоговорителя 30 Гц, в закрытом коробе этот динамик позволяет по уровню – 6 дБ получить низшую рабочую частоту 70 Гц. В фазоинверторе это 50 Гц и размеры закрытого короба:

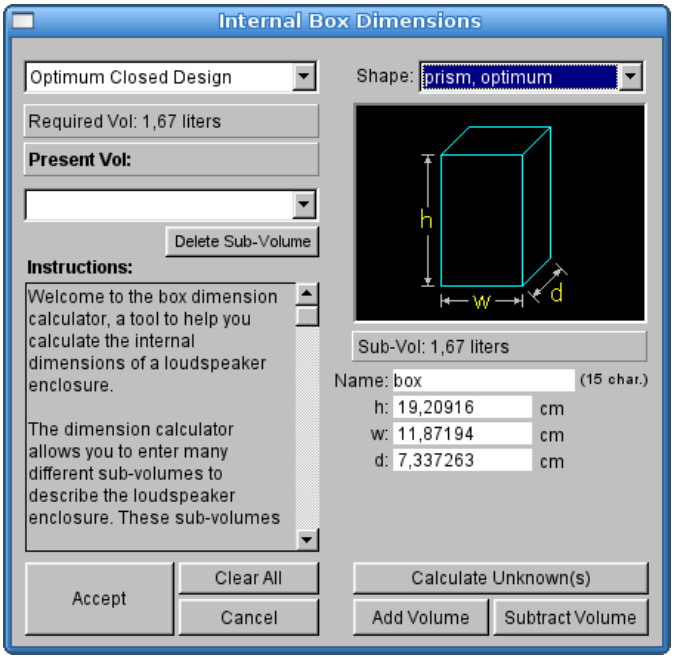


Рис. 8.10. Размеры коробки для громкоговорителя

Это очень небольшая упаковка для низкочастотного громкоговорителя. Фазоинвертор позволяет получить параметры лучше, но и размеры его больше. Программа позволяет поэкспериментировать с объемом (*Vc* для Closed box). Увеличение объема закрытого ящика

до 5 л сдвигает частоту к 60 Гц.

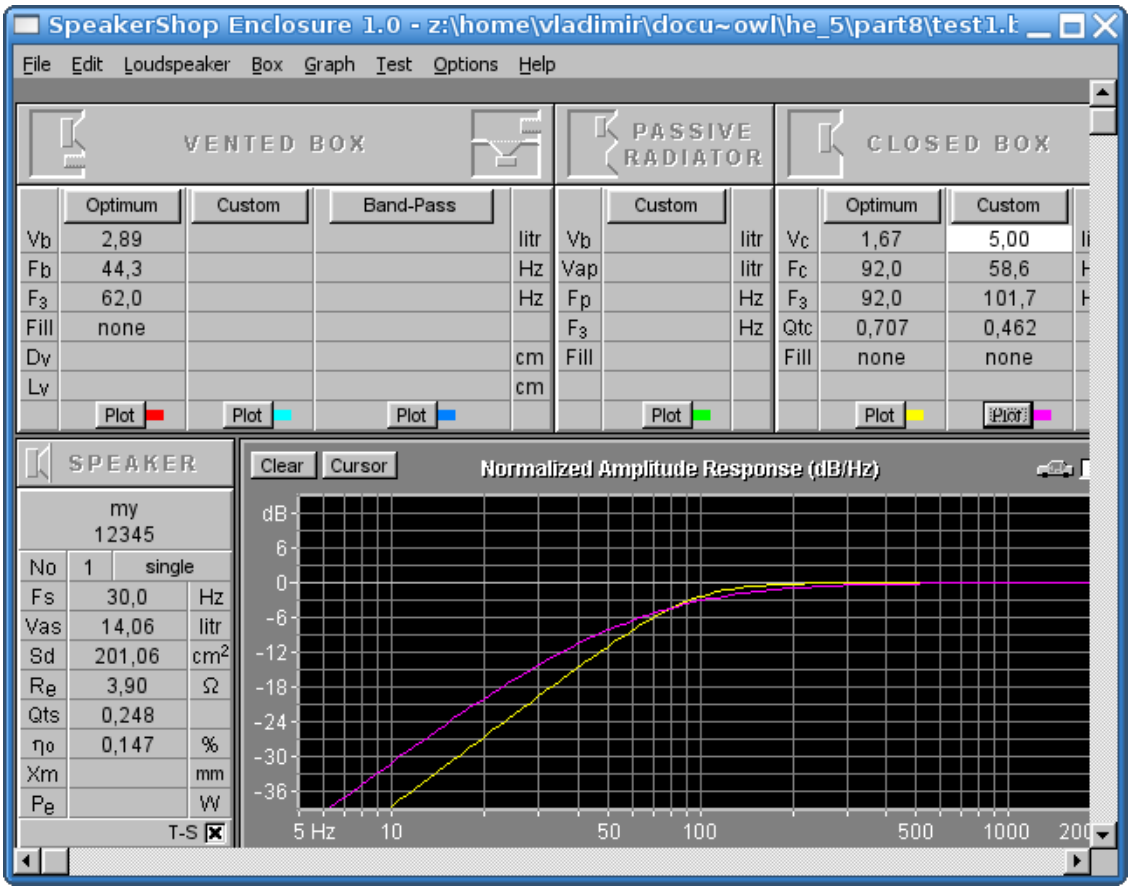


Рис. 8.11. Изменение объема коробки громкоговорителя

Работая с реальным динамиком можно выбрать такой вариант, который устраивает больше всего, но увеличивая объем, можно заметить, что после оптимального объема дальнейший рост привносит все меньше и меньше новизны в результат. Если с заданным динамиком не получается желаемая нижняя рабочая частота, то либо следует заменить динамик, либо рассмотреть вариант фазоинвертора (Vented box) с заполнителем или нет.

Кстати и здесь программа позволяет предварительно оценить результат, сравнивая частотные характеристики разных конструкций. Впоследствии, когда громкоговоритель будет выбран, реализован и настроен, было бы интересно сравнить его звучание, применяя заполнитель и без него, реальных музыкальных произведений.

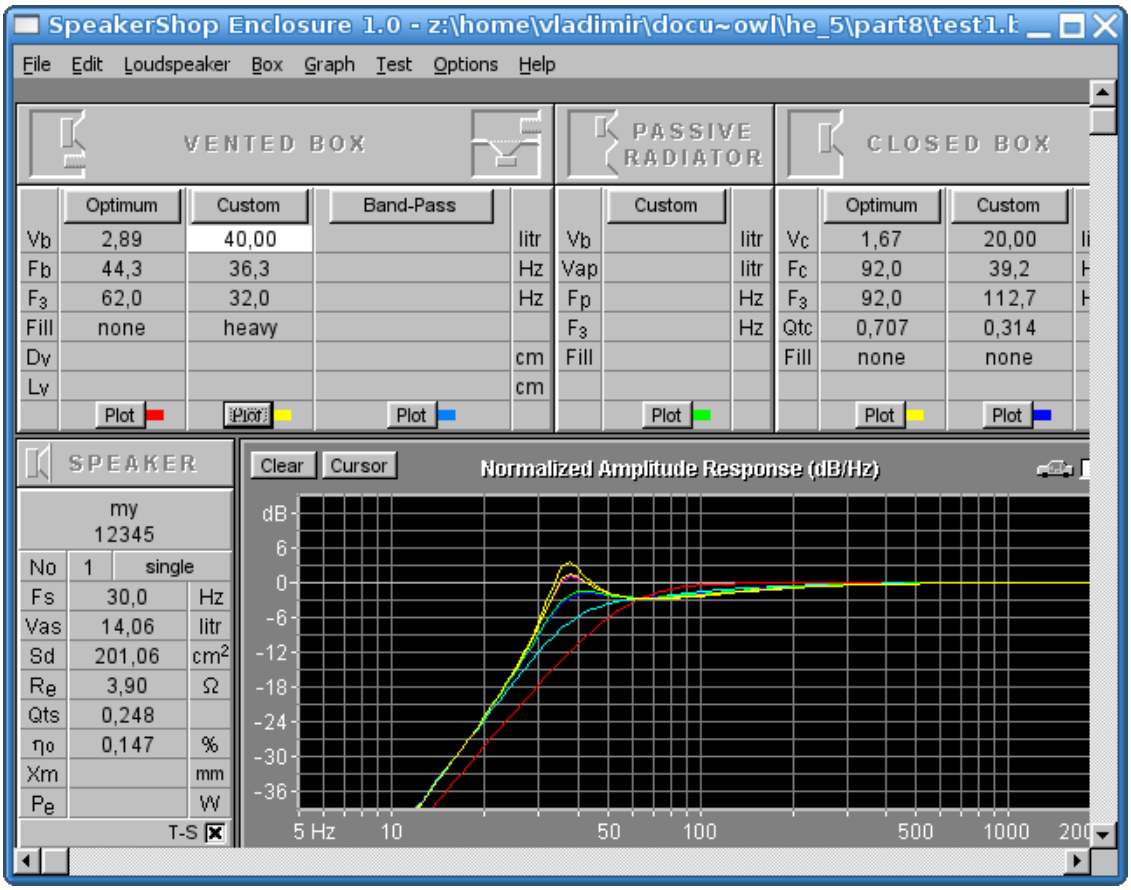


Рис. 8.12. Разные варианты с использованием фазоинвертора

Первоначальный оптимальный объем фазоинвертора дает достаточно гладкую кривую. Рост объема (особенно с заполнителем) вызывает появление небольшого выброса, который увеличивается с ростом объема, но это все меньше и меньше сказывается на снижении нижней границы диапазона.

Очень полезная и удобная программа. Если ваш интерес выходит за рамки рекомендаций по оптимальному использованию возможностей динамика, то вам кроме выполнения многократных работ по изготовлению короба громкоговорителя потребуется еще и проведение измерений результатов по звуковому давлению. Проводить их следует в чистом поле, иначе полученные результаты могут быть неверными. Отчего?

Прежде чем переходить к конкретному обсуждению того, какой вариант лучше выбрать начинающему, вспомним, что звуковая волна, произведенная громкоговорителем, будет отражаться от стен, возвращаться и отражаться и т.д. Поглощение звука на низких частотах не столь велико, а отраженная волна будет интерферировать. Вернемся к программе Qucs, где можно посмотреть, как будет происходить интерференция на примере электрической схемы. Это не будет полный аналог акустических экспериментов, но пример покажет «расстановку сил» на этом фронте.

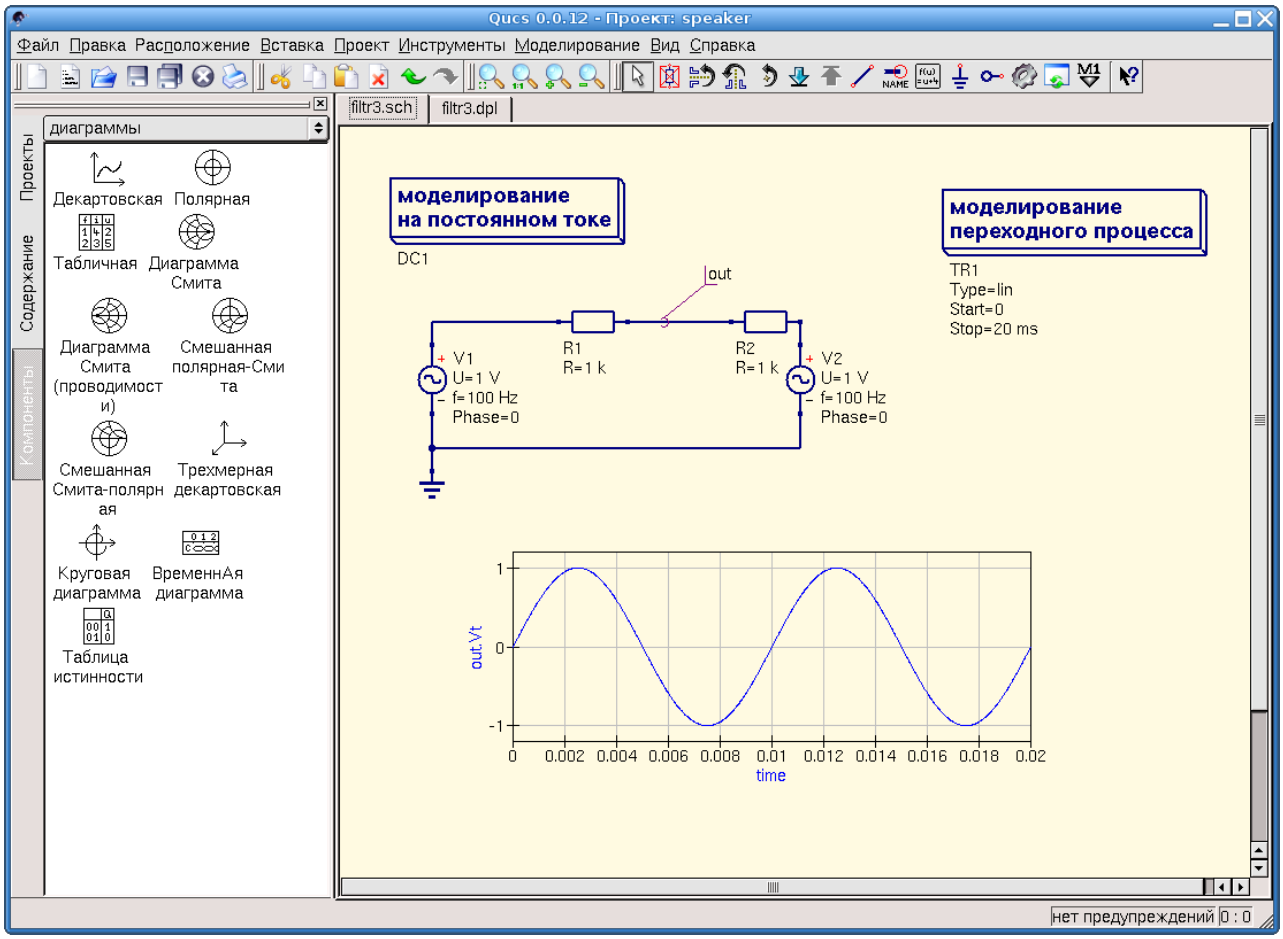


Рис. 8.13. Эксперимент с «отраженной волной»

В эксперименте источник V1 – наш громкоговоритель, а источник V2 имитирует отражение волны. Резисторы R1 и R2 (для электрической схемы) образуют делитель. Если удалить источник V2, то выходной сигнал в точке наблюдения (*out*) будет равен 0.5 В, отражения нет.

Если отражение полное, как на диаграмме выше, то оба сигнала складываются, и напряжение равно 1 В при условии, что фазы прямого и отраженного сигнала совпадают. На рисунке оба источника имеют нулевую фазу (Phase = 0).

Отчего будет зависеть, совпадут фазы или нет? От частоты и расстояний от источника звука до отражающих поверхностей и от места расположения «слушателя». Как выглядит результат, если фазы излученной и отраженной волны будут противоположны можно увидеть на рисунке ниже.

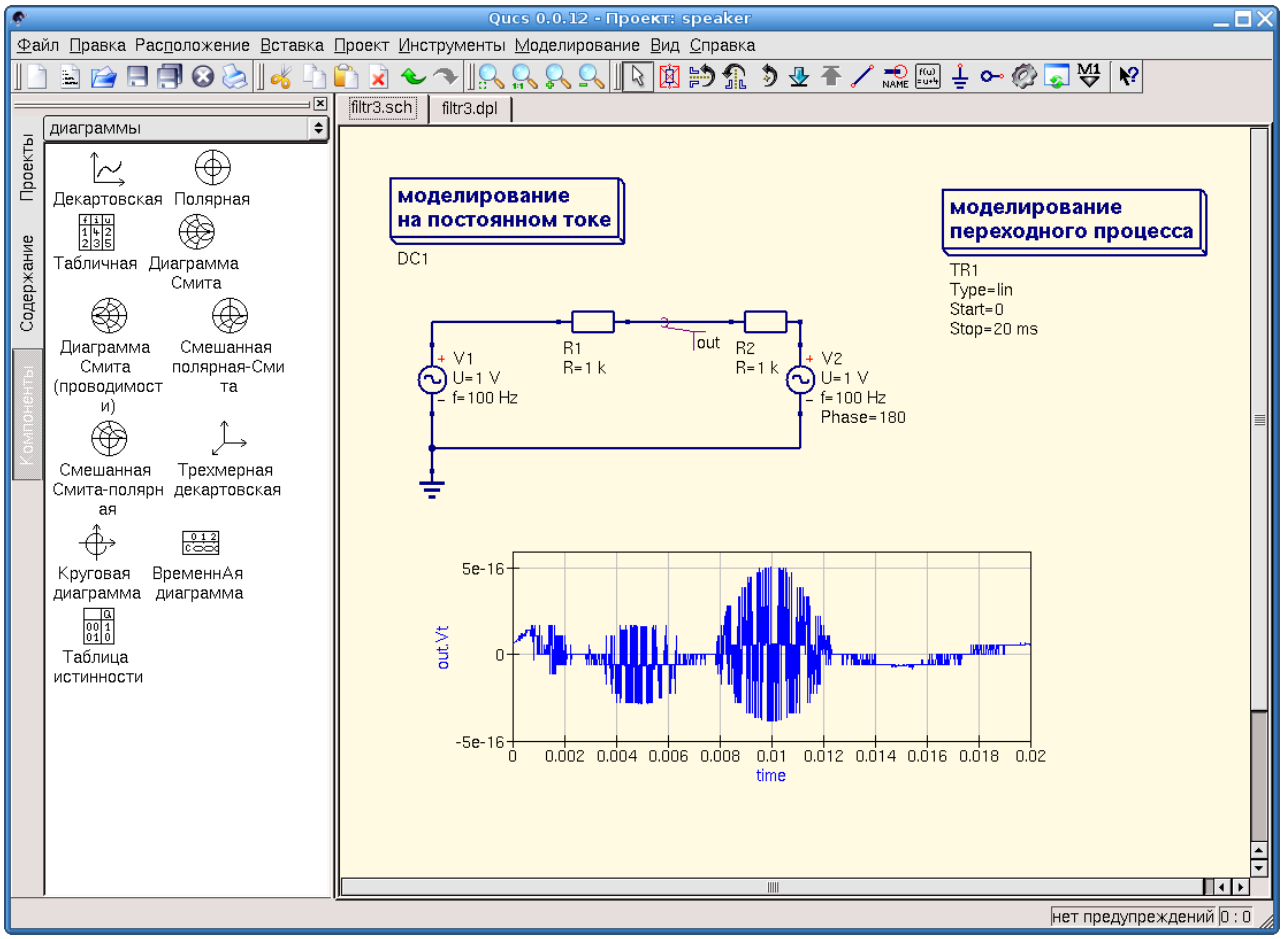


Рис. 8.14. Противофазная «отраженная» волна

На уровне долей микровольт программа честно показывает наличие каких-то шумов. Будем считать, что это в «ушах звенит». Если учесть, что отражающих поверхностей много – стены, потолок, пол, мебель; если добавить, что частот в нижней части звукового диапазона тоже хватает, то предсказать, как будет распределяться звуковое давление, созданное нашим громкоговорителем, затруднительно. Именно по этой причине измерения АЧХ громкоговорителей проводят либо в заглушенной камере, либо на открытом воздухе вдали от зданий и больших поверхностей.

А теперь вернемся, чтобы завершить эту небольшую главу о громкоговорителях, к вопросу о том, чтобы мы хотели получить взамен, потратив деньги на покупку всех компонентов и много времени и сил на реализацию проекта?

Если для вас реализация проекта сводится к выбору готового комплекта: источник звука, усилители, акустическая система, – для четко очерченного применения, например, для любимого автомобиля, и если у вас нет финансовых проблем, то принцип – чем дороже, тем лучше, вполне подходит в качестве начальных условий. Если вы прослушали как звучит это «чудо» в реальных условиях и пришли в неописуемый восторг, то я скажу вам так – вы поступили правильно. И не слушайте меня, когда я начинаю рассказывать, что вы не учли то, и не поняли это, а в результате выбросили деньги на ветер. Не слушайте, завистники всегда найдутся! Не верьте им.

Если вы реализуете проект с желанием своими руками создать нечто, что стоит дорого, но обошлось вам много дешевле, благодаря вложенному труду, ваша задача сильно усложняется. Вам придется потратить много времени на подготовку к реализации проекта. Вам предстоит

столько шагов по выбору каждого элемента вашего будущего устройства, что иначе как паломничеством этот путь не назовешь. Читайте книги, они не только развлекут вас на этом интересном пути, но и уберегут от ошибок. Экспериментируйте. Изучайте.

Если для вас реализация проекта выглядит как продолжение школьных уроков или лекций в учебном заведении, и вы не ждете чудес, но готовы идти от простого к сложному с намерением остановится, когда почувствуете, что идете не туда, или если вам просто интересно, как же это все устроено, тогда давайте попытаемся реализовать простой проект, не привязанный к конкретным условиям применения, не блещущий ни точностью, ни великолепием конечного результата, но позволяющий заглянуть в замочную скважину лаборатории производителей электроакустики.

Проект «Громкоговоритель»

Предварительные предположения: положим, что вы хотите немного разобраться в вопросе, что у вас есть немного денег, которые вы можете потратить, и есть много желания провести ряд экспериментов (на которые и уйдут деньги), что вы свободны до следующей осени (не каждый день, и не целый день, конечно). Предположим, что у вас есть источник достаточно качественного звука в виде CD-проигрывателя или проигрывателя грампластинок, есть необходимое оборудование и рабочее помещение.

При наличии всего этого я предлагаю реализовать проект, конечной целью которого будет возможность на столе (может и виртуальном, пока не важно) собрать некий звуковой комплекс, чтобы получить небольшое представление о тех трудностях, с которыми можно столкнуться при работе со звуком.

На первом этапе проекта я предлагаю собрать простую стереофоническую систему, которую на втором этапе проекта можно несколько модифицировать, проведя ряд дополнительных проверок.

Итак. Какие из параметров звучащего оборудования мы примем во внимание в первую очередь? Полосу пропускания. Наш слух позволяет говорить о полосе частот 20-20000 Гц. Далеко не всякий слышит эти частоты, далеко не всякая лаборатория позволит экспериментировать в такой полосе частот, и далеко не у всех хватит терпения и средств на реализацию этого параметра в полном объеме. Давайте поступим как всегда, спустимся с небес на землю и посмотрим, чем нам могут «потрафить» производители динамиков в этой части, при условии, что мы не намерены пока тратить много денег и запросы у нас достаточно скромные.

На сайте www.bluesmobil.com я обнаружил параметры старых отечественных динамиков. Первое, что я сделал бы на вашем месте, узнал бы у знакомых и друзей, нет ли у кого ненужных им музыкальных центров – многие покупают новые, а старые с удовольствием готовы подарить тем, кто заберет их. Попробуем оценить вариант со старыми отечественными громкоговорителями (если что-то будет новее, то это будет только лучше).

В качестве примера я выберу такой, для которого есть параметры необходимые для расчетов, но подразумевается, что вы эти параметры получите путем измерений. Я выбираю 15ГД-14 (25ГДН-3-8). Я уверен, что для проведения всех экспериментов, описанных ниже, этого не только хватит, но еще и останется. Номинальная мощность громкоговорителя 15 Вт. Много это или мало? Попробуем посчитать.

Характеристическая чувствительность этого динамика 86 дБ. Что должно означать – при подаче на него 1 Вт на расстоянии в 1 м он будет создавать звук с уровнем громкости на 86 дБ выше порога слухового восприятия. Если мы вспомним, что духовой оркестр создает столько

же шума, играя марш «Прощание славянки», то можем согласиться, что больше и не нужно. А если учтем, что при подведении мощности в 10 Вт (мы считаем по формуле 10 $\log(P_{10}/P_1)$, где P_{10} мощность равная 10 Вт, а делитель равен единице) этот шум усилится до 96 дБ, а максимальная мощность этого громкоговорителя 25 Вт (а это около 100 дБ)... для настольных экспериментов должно хватить.

Предположим, что я измерил необходимые для работы программы JBL параметры, и теперь хочу оценить полученный результат. Как выглядит полный набор параметров (часть из них я придумал).

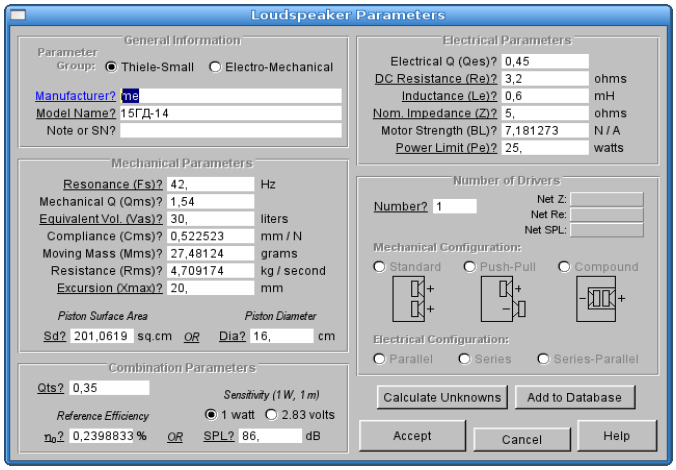


Рис. 8.15. Возможные параметры для расчета громкоговорителя

Нижняя граничная частота громкоговорителя после расчета в программе JBL оказывается равной 45 Гц по уровню – 12 дБ при размерах короба:

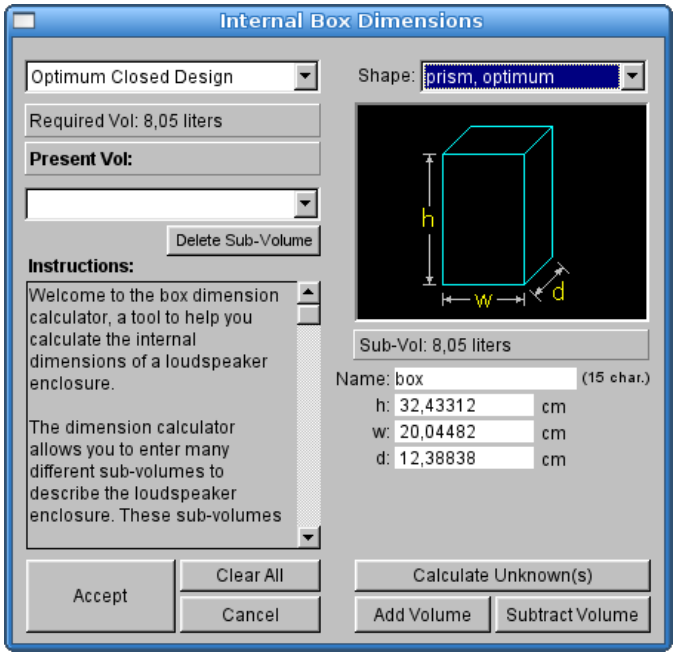


Рис. 8.16. Размеры громкоговорителя

Нелинейные искажения громкоговорителя при номинальной мощности должны получиться порядка 3-6%.

И размеры, и остальное с моей точки зрения приемлемо, даже если размеры увеличить до более реальных w = 46 см, h = 58 см, d = 37 см. Эти расчеты, проведенные с реальным громкоговорителем и воплощенные в реальную конструкцию предполагается использовать для воспроизведения низких частот. Для средних и высоких частот можно использовать динамик 3ГД-45 или двухполосный громкоговоритель с верхней граничной частотой 15 кГц.

После выбора динамиков и оценки их размеров, и до покупки чего-либо, следует определиться с необходимым оборудованием. Для измерения параметров низкочастотного динамика перед расчетами потребуется мультиметр (надеюсь он есть) и генератор низких частот (хорошо, если он есть). Для отдельных экспериментов достаточно генератора на несколько частот, но измерения параметров динамика потребует плавной перестройки частоты. Если генератора нет, то есть смысл собрать его до начала работы над проектом. Можно быстро собрать что-то подходящее, но генератор прибор очень полезный, уж если его собирать, то можно и постараться. Для градуировки его шкалы можно использовать, если есть, частотомер мультиметра (например, мультиметр UT30F позволяет измерять частоту до 10 МГц) или, если есть, осциллограф, или даже собрать частотомер, который можно объединить с генератором, чтобы не изготавливать шкалу. При выборе схемы генератора, которую можно найти в журналах «Радио» или на сайтах, следует здраво оценить свои возможности. Еще один удобный вариант для начала роботы – использовать компьютер. Есть программы, превращающие звуковую карту в генератор или осциллограф. Если такой вариант получается, то это самый простой и удобный начальный этап в части измерительного оборудования.

Кроме генератора понадобится усилитель мощности звуковой частоты. Усилитель для этих целей разумнее собрать на микросхеме, например, TBA820 или аналогичной, или собрать усилитель на транзисторах, что позволит провести дополнительные эксперименты именно с усилителем мощности звуковой частоты. Схему усилителя, как и другие схемы, можно взять из журнала, из книги или найти на радиолюбительском сайте. При выборе схемы в данный момент я ориентировался бы пока на быстрейшее достижение цели. Позже, когда вы будете с электроникой «на коротке», вы сможете вернуться к схемотехнике УМЗЧ. Это интересный предмет для исследования и собственно электроники, и, что не менее интересно, своей фонотеки после создания новой системы звукоусиления. Многие музыкальные произведения, которые вы хорошо знаете, могут приобрести настолько новое звучание, звуковая картина изменится настолько сильно, что вы будете поражены. Конечно, если сравнивать звучание обычного музыкального центра с системой, которую вы сможете создать.

Еще одно замечание, если вы намереваетесь сейчас собрать усилитель, который предназначен только к целям измерений, то обратите внимание на напряжение питания. По возможности используйте напряжение питания единообразное для ряда измерительных приборов, которые вы будете создавать. Это позволит в отельных случаях использовать один и тот же блок питания для разных измерений, если же мощность блока питания достаточная, а приборы, как правило, потребляют не так много, к одному блоку питания можно подключить все необходимые для эксперимента приборы.

А сейчас вернемся к схеме усилителя на транзисторах. Она может быть такой:

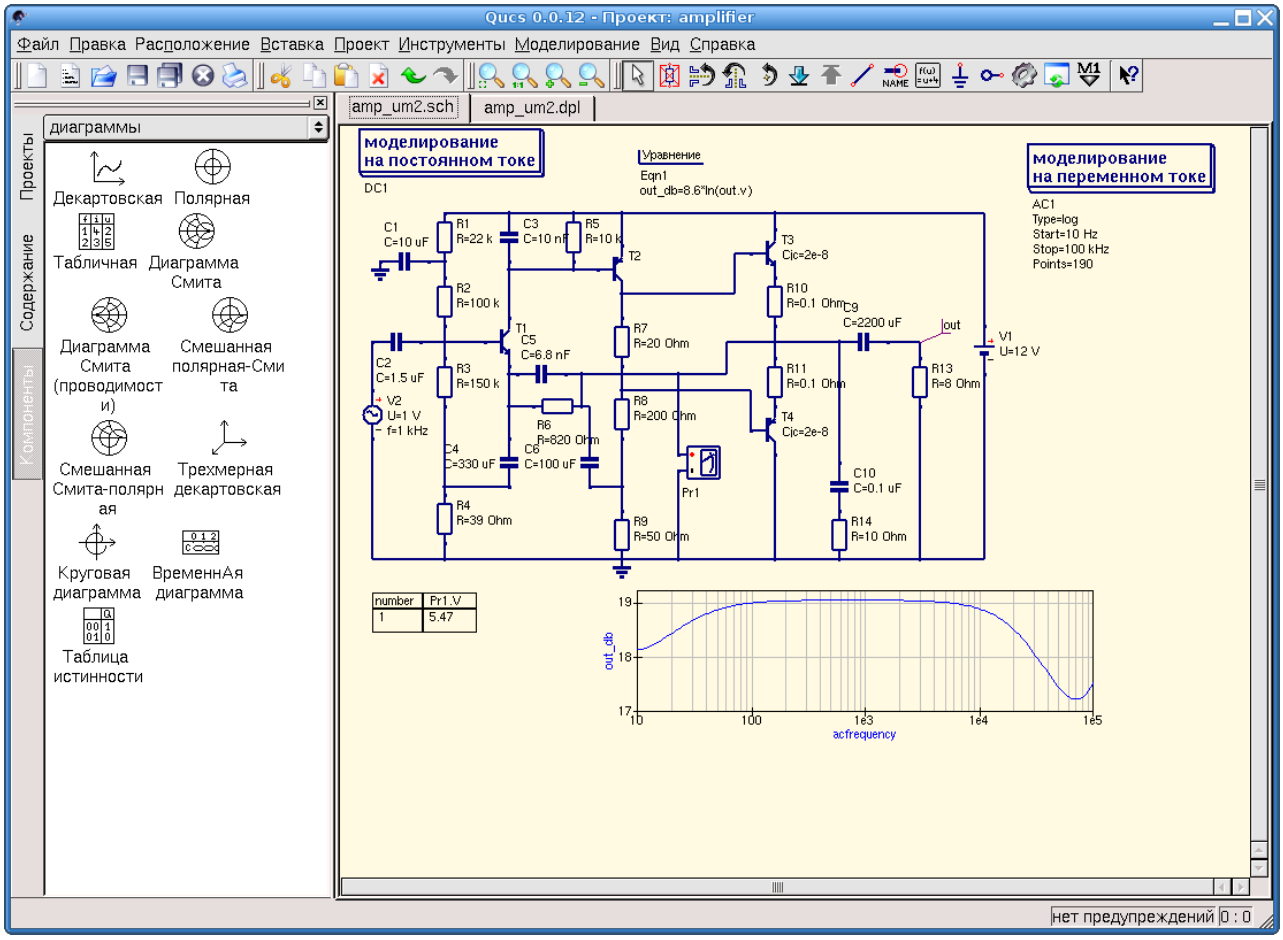


Рис. 8.17. Схема усилителя мощности для экспериментов

При питающем напряжении 12 В этот усилитель на нагрузке 8 Ом обеспечит мощность в один ватт. Этого должно хватить на все время экспериментов. Собирать усилитель из отдельных деталей есть смысл в том случае, если вы хотите познакомиться с работой усилителей мощности, проверить все токи и напряжения, рассчитать все мощности: мощность рассеяния на резисторах и транзисторах, мощность, отдаваемую в нагрузку, в зависимости от напряжения питания. Оценить влияние отрицательной обратной связи по постоянному току через резистор R6 и по переменному току через делитель R6R4 и конденсатор C6. Можно проверить работу усилителя в разных режимах, меняя величину резистора R7. В реальных схемах этот резистор часто делают переменным, а среднюю точку потенциометра подают на базу транзистора, включенного параллельно этому резистору. Сам транзистор располагают на теплоотводах, на которые устанавливают мощные выходные транзисторы (например, пару КТ816 и КТ817), чтобы автоматически смещать рабочую точку при нагреве выходных транзисторов.

Осмотримся. Мы подготовили все оборудование, собрали громкоговорители (их два: низкочастотный и широкополосный). У нас два усилителя мощности, один работает на низкочастотный канал, второй на широкополосный. Конечно, не обязательно иметь два канала, можно поставить разделительный фильтр на выход единственного усилителя, но лучше сразу разделить каналы полностью, поскольку первые эксперименты в большей мере относятся к низкочастотному каналу.

Для разделения каналов выберем разумную частоту раздела, например, 200 Гц. И выберем фильтр, который разделит каналы. Мы, в сущности, уже встречались с фильтрами, когда

обсуждали амплитудно-частотную характеристику усилителя. Хорошие фильтры получаются из конденсаторов и катушек индуктивности. Но на низких частотах габариты катушки получаются весьма «существенными», добавьте к этому то, что ее следует мотать толстым проводом и что не желательно применять сердечник из-за возможного его насыщения... впрочем, фильтр, обычно, размещается в громкоговорителе достаточно большого размера.

Фильтр меньших габаритов и легче настраиваемый — это RC фильтр в двухканальной системе. Его можно рассчитать. Его можно «подогнать» на макетной плате, используя паяльник и приборы, что полезно и для обогащения опыта работы с приборами, и опыта работы с паяльником, и опыта компоновки элементов на плате.

Однако лично я пойду путем «проб и ошибок» в программе Qucs, чтобы получить следующее:

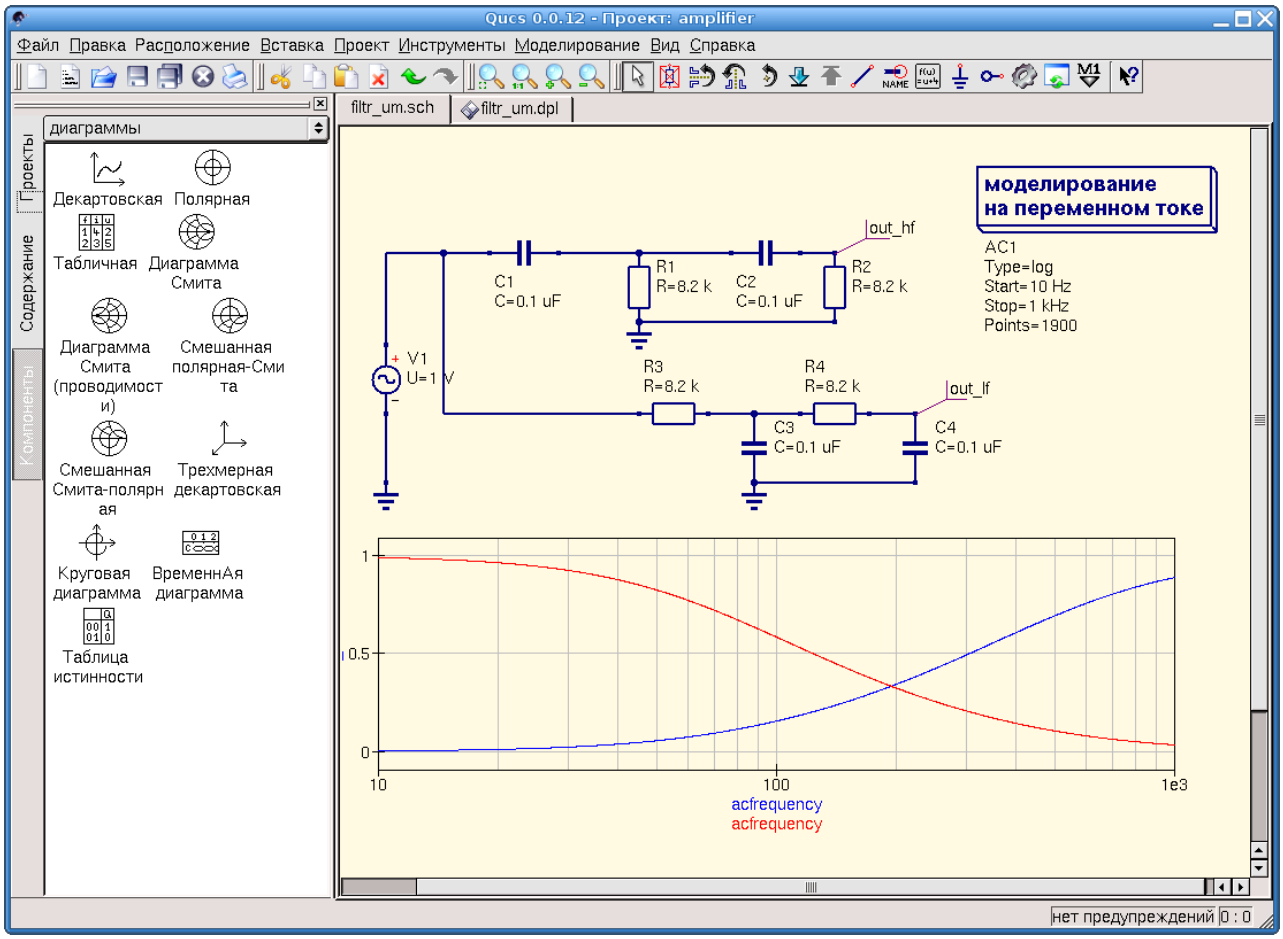


Рис. 8.18. Фильтр, разделяющий каналы на НЧ и ВЧ

С виду все чинно и благородно: сигнал с выхода *out_hf* пойдет на усилитель широкополосного канала, с выхода *out_lf* на усилитель низкочастотного канала. Два звена в верхнем фильтре настроены на 200 Гц, два в нижнем тоже. Частотные характеристики пересекаются, примерно, на частоте 200 Гц, этой точности более, чем достаточно. Но я чуть было не забыл об одном сомнительном месте, которое попробую показать.

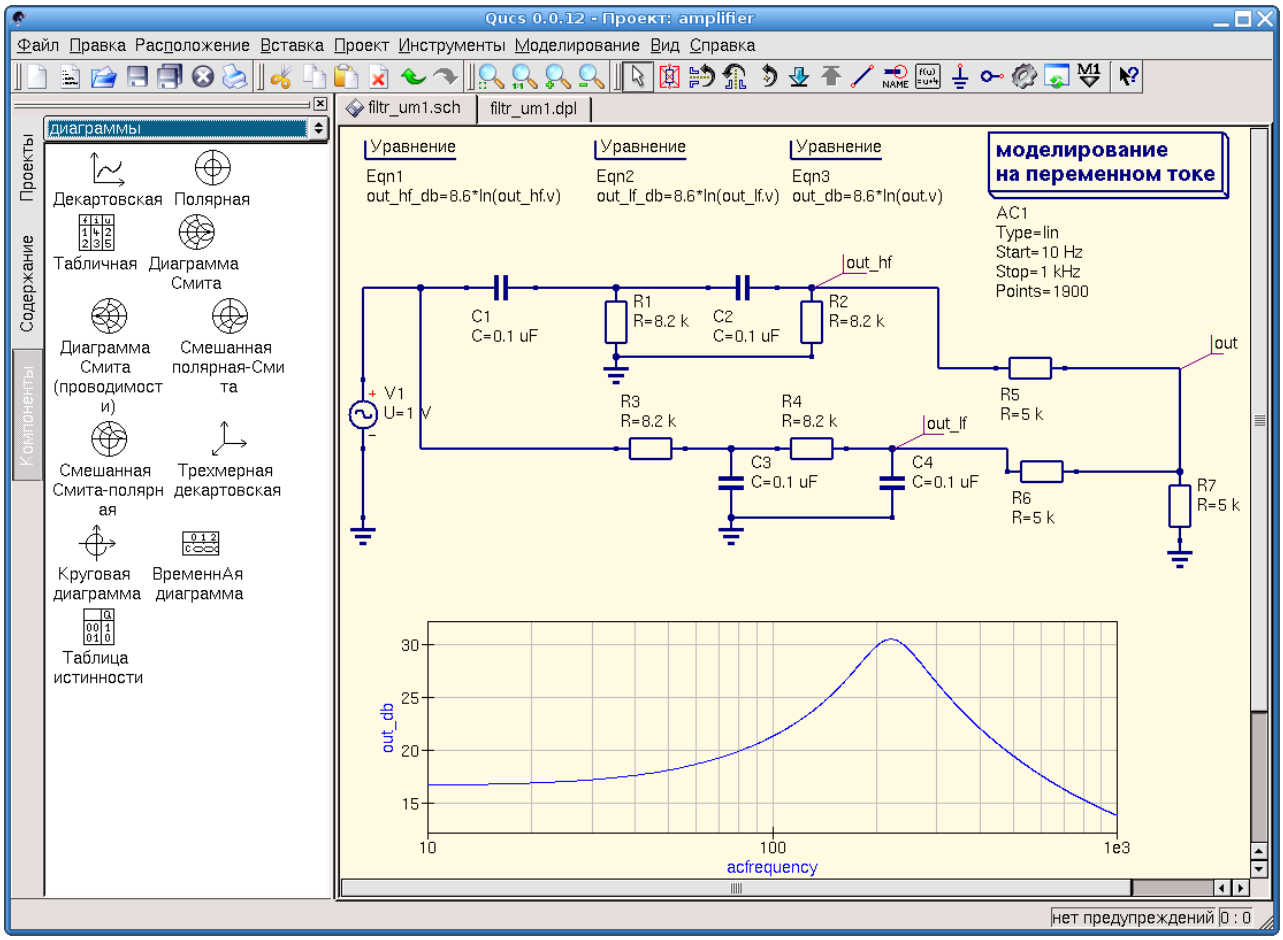


Рис. 8.19. Результирующий сигнал при объединении выходов фильтров

Для верности я даже использовал в качестве единиц децибелы. С помощью резисторов R5, R6 и R7 я объединяю оба канала, как если бы снимал АЧХ всего устройства с помощью микрофона по звуковому давлению. Думаю, вам уже ясно, о каких сомнениях я говорил. На частоте раздела работают оба канала, и сигнал от громкоговорителей складывается, в результате можно получить очень заметный выброс на результирующей амплитудно-частотной характеристике. Чтобы этого не произошло, следует, наверное, сместить частоты, как бы «раздвинуть» их.

Это еще одна из причин, по которой я предпочел бы менять емкость конденсаторов и величину резисторов, а не перематывать катушки. Конечно, лучше вначале рассчитать все, а потом мотать катушки. Но мы же экспериментируем!

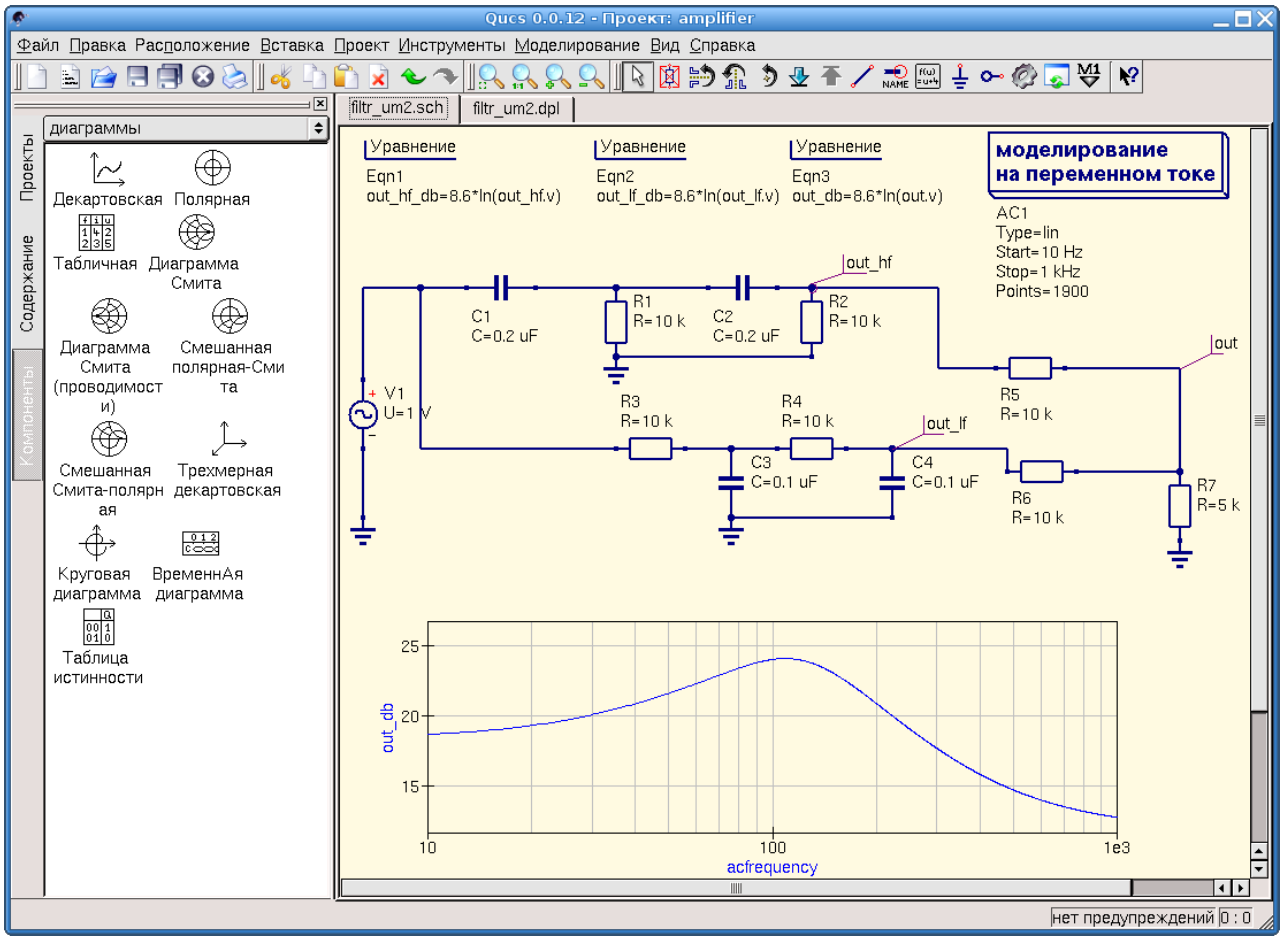


Рис. 8.20. Коррекция параметров фильтра

Проводя реальные измерения, можно сравнивать напряжение, положим на частотах 50 Гц, 200 Гц и 1000 Гц, включив вместо громкоговорителей эквивалентные резисторы, и подбирая элементы фильтра так, чтобы суммарная мощность обоих каналов была одинакова на этих частотах (в разумных пределах, конечно).

Первый эксперимент в области акустики, который я предлагаю, позволит вам определиться с тем, как влияет нижняя граничная частота на конечный результат – звуковую картинку ваших любимых музыкальных произведений. Почему ваших любимых музыкальных произведений, потому что требования к параметрам звуковой системы очень сильно зависят от характера музыки. Для эстрадной песни, скорее всего, потребуется меньший диапазон громкости и более узкая полоса частот. А вы ведь собираетесь сделать что-то именно для себя, а не для кого-то.

Чтобы оценить нижнюю границу частотного диапазона, можно применить фильтры, которые смещали бы нижнюю границу с 45 Гц (наши расчеты), к 55 Гц, а затем к 65 Гц. Сделав несколько фильтров, включая их в низкочастотный тракт, вы можете при одинаковой громкости прослушивать любимые музыкальные фрагменты и понять, действительно вам нужно именно 45 Гц или достаточно 65 Гц, если на слух нет существенной разницы.

Кстати, при изготовлении низкочастотного громкоговорителя вы можете использовать фанеру 10-12 мм толщиной, которую очень полезно оклеить линолеумом без матерчатой основы. Все щели, в панелях и между панелью и динамиком, следует промазать пластилином или похожей на него мастикой, только не твердеющей, если вы намерены разбирать громкоговоритель, чтобы уменьшить, например, его объем. Предварительно, перед

изменением объема, вы можете в программе JBL варьировать объем короба для оценки смещения нижней частоты. Если вы используете усилитель похожий на тот, что на рисунке 8.17, то можете изменять низшую частоту с помощью конденсаторов С2 и С4, уменьшая их значение, а предварительно оценить влияние этих изменений можно в программе Qucs. Окончательно следует провести измерения на эквиваленте громкоговорителя. И не забывайте, проводя эксперимент, что подчас мы слышим не то, что слышим, а то, что хотим услышать – зная, что частота нижней границы изменилась, вы можете на одном и том же музыкальном фрагменте заметить существенное ухудшение общего восприятия картинки. Повторите это прослушивание через некоторое время. Проделайте это несколько раз. И не забывайте, что если вы существенно меняете нижнюю частоту, скажем с 45 Гц до 80 Гц, пропорционально следует изменять и верхнюю границу, скажем, с 15 кГц до 12 кГц, или отверните широкополосный громкоговоритель при всех прослушиваниях несколько в сторону!

Итак. Первый эксперимент по определению необходимой лично вам нижней частоты воспроизведения вы закончили. Что можно сказать о результатах? Если вы заметили, и уверены в этом, явное ухудшение качества звука, то оставьте габариты низкочастотного громкоговорителя первоначальными. А после экспериментов с прототипом, когда будете планировать создание реального тракта, подумайте о выборе динамика с более низкой частотой резонанса.

Если вы не заметили разницы, то можно в окончательном варианте сделать громкоговоритель меньших габаритов, но, опять-таки, напомню, и верхнюю границу лучше привести в соответствие. Мало того, если вас вполне устраивает низшая частота 60 Гц, то ее может обеспечить подходящий широкополосный динамик, не нужно будет делать многополосную систему, что значительно упрощает построение электрической части системы.

И, наконец, если вы заметили существенную разницу при экспериментах по определению нижней частоты, вы можете, уменьшив планируемую мощность, поставить на входе низкочастотного канала фильтр, который поднимает низшие частоты, тем самым сдвинув полосу частот ниже 45 Гц. Теперь вы можете рассмотреть, «расслушать» и такой вариант.

Существо первого эксперимента именно в определении частотного диапазона будущей звуковой системы. Можно попытаться сразу построить ее с полосой 20 Гц – 20 кГц, но это сложно и дорого, лучше это отложить до той поры, когда у вас будет больше опыта. И не забудьте, мы еще только начали экспериментировать!

Следующий эксперимент, который я вам предлагаю, значительно проще. Нам нужно определиться еще с одним важным параметром – достаточной громкостью. Мы выбрали уровень громкости 86 дБ. Надеюсь, вы проводили предыдущий эксперимент, расположившись в метре от громкоговорителей. Этот эксперимент тоже проведите на этом расстоянии. Если громкости было достаточно, то попробуйте уменьшить мощность в 10 раз, что будет соответствовать уровню громкости в 76 дБ. Найдите максимальную громкость, которая позволяет вам с удовольствием длительно слушать (слушать, а не слышать) вашу любимую музыку. Если громкости недостаточно, а собирая экспериментальный усилитель вы установили выходные транзисторы (или микросхему) на радиаторы, то, возможно, изменяя напряжение питания вы можете увеличить мощность усилителя до 10 Вт. Прослушайте разные фрагменты при уровне громкости 96 дБ. При экспериментах имейте ввиду, что от длительного прослушивания человек устает, ему кажется, что звук слишком тихий. Повторяйте этот эксперимент после длительных перерывов, чтобы удостовериться, что ошибки, связанной с усталостью нет, потому что уровень в 106 дБ будет достигаться при мощности усилителя (и громкоговорителя) 100 Вт! И еще одно, если вам в какой-то момент

показалось, что хорошо бы громкость иметь с запасом, и вы решили, что прибавите мощность, но слушать будете не с полной громкостью, то обратите внимание на шумы при мощности усилителя, скажем, 10 Вт и 1Вт, когда вы меняете именно мощность, а не громкость регулятором. Суть в том, что если вы используете десяти-ваттный усилитель, но используете его с громкостью одно-ваттного, то шумы могут возрасти на 10 дБ, что неоправданно приведет к снижению динамического диапазона. Это явно не самый лучший результат. Запас нужен, но разумный. И последнее, что касается громкости. Когда вы измеряете мощность, проверяете работу фильтров (надеюсь, на эквивалентной нагрузке, иначе это может привести к ошибкам) вы используете генератор. Когда вы прослушиваете музыкальные фрагменты, то трудно определить и мощность и уровень, поскольку реальные сигналы имеют сильно меняющийся уровень и имеют совсем не синусоидальную форму, что затрудняет измерения с помощью вольтметра. Просто, имейте это ввиду.

Мы определились с частотным диапазоном и уровнем громкости, что определило требования к динамикам и усилителю. Теперь я предлагаю вам провести еще один эксперимент.

Природа его в том, что человек слышит звуки разной частоты с разной воспринимаемой им громкостью, и этот эффект сильно зависит от уровня громкости. Есть графики, которые называются «кривыми равной громкости», они-то и показывают, как человеку все это слышится. Не буду приводить графики, но запишу значения уровней для некоторых частот.

Для уровня громкости близкого к порогу слышимости, то есть, нулевого уровня и сигнала с частотой 1 кГц, равногромкий сигнал с частотой 50 Гц должен воспроизводиться с уровнем громкости на 40 дБ больше, а для сигнала с частотой 15 кГц на 15 дБ больше.

Для уровня громкости 40 дБ, на частоте 50 Гц следует прибавить 20 дБ и столько же на 15 кГц.

И для уровня 80 дБ, 15 дБ для 50 Гц и 10 дБ для 15 кГц.

Используя корректирующие фильтры, попробуйте, определив, скажем, три уровня громкости, послушать музыкальные фрагменты с частотной коррекцией и без нее. В качестве корректирующего фильтра для этих экспериментов можно использовать любую подходящую схему эквалайзера. И если в результате экспериментов вы будете склоняться к необходимости коррекции, то можно либо поискать схему регулятора громкости с тон-коррекцией, либо использовать эквалайзер, который лучше откалибровать так, чтобы не ломать голову каждый день, насколько менять положение регуляторов. Одновременно, по результатам последних двух экспериментов вы можете упростить решение задачи с регулятором громкости. Как правило, вы слушаете музыку либо в режиме «Я слушаю музыку», либо в режиме «Я что-то включил, чтоб играло, пока я занимаюсь делом». Двух-трех уровней громкости может хватать на все случаи жизни, а тогда можно вместо потенциометра поставить переключатель на три положения, его можно сделать с тон-коррекцией. Он будет работать много дольше, чем работает потенциометр, у которого от вращения изнашиваются и токосъемник и дорожка, и спустя некоторое время начинается треск при регулировке громкости.

Если у вас есть старый динамический микрофон, которым комплектовались некогда катушечные стерео магнитофоны, и который снабжался паспортом с типовой АЧХ чувствительности, то вы можете проделать несколько очень любопытных экспериментов по измерению параметров вашей системы по звуковому давлению, если у вас есть дача или автомобиль. И если, конечно, вы еще не настолько «достали» своих соседей, что они уже давно перестали с вами здороваться. Зачем нужна дача? Сейчас поясню.

С помощью микрофона и микрофонного усилителя с линейной характеристикой во всей полосе звуковых частот (можно использовать операционные усилители, включив микрофон

между прямым и инверсным входом, если микрофоны двух-проводные с экраном, или обычным образом, если микрофоны включены одно-проводным экранированным проводом) вы можете снять частотную характеристику громкоговорителей по звуковому давлению в полосе, скажем, 50-10000 Гц. Типовую характеристику чувствительности микрофона используйте для коррекции полученных данных, внеся соответствующие поправки. А получив АЧХ вашей системы в комнате, на автомобиле поезжайте в чистое поле в безветренную погоду и повторите измерения. Сравните неравномерность частотных характеристик в помещении и на открытом воздухе, и вспомните эксперимент рисунка 8.13. И питание в 12 В я рекомендовал «с прицелом» именно на эти эксперименты.

После проведения экспериментов вам легче будет определиться с основными параметрами громкоговорителей и усилителей мощности. Каждый лишний ватт, лишний децибел, каждые 10 Гц в сторону снижения нижней граничной частоты дадутся вам нелегко, и обидно, если они оказываются именно «лишними».

Итак. С параметрами мы определились. Положим, что к настоящему времени у вас есть (или есть возможность сделать) трехканальную испытательную стерео систему, где для левого и правого громкоговорителей вы используете отдельные каналы с полосой 200 Гц- 15 кГц, а до частоты 200 Гц вы используете общий низкочастотный канал, суммирующий сигналы левого и правого канала.

Теперь добавьте к этому еще один усилитель мощности, к которому подключите два громкоговорителя (широкополосных, положим, 200 Гц — 10 кГц), но включенных в противофазе друг к другу. На вход этого усилителя подайте разностный сигнал, то есть, сигнал левого канала минус сигнала правого канала. Используйте эту конструкцию в качестве тыловых громкоговорителей. Регулируя громкость этого «тылового» канала при прослушивании музыкальных фрагментов, послушайте получающуюся звуковую картину. Если к этому каналу добавить еще и ревербератор, то звуковая картина, которая, правда, может сильно зависеть от характера записи музыкальных фрагментов, может полностью изменить ваше представление о том, что бы вы хотели сделать. А главное, теперь вы, приступая к реализации высококачественной системы, сумеете выбрать и схему усилителя, и громкоговорители. И для усилителя, и для громкоговорителей очень важных нюансов, отражающихся на качестве звука, еще очень много, но начало положено.

Вот такая получилась короткая глава о звуке. Звук гибкий и пластичный, хотя и своенравный материал, вылепить из него можно все, что угодно. А что получится? Что получится, то и получится...

Глава 9. Теплоотвод

Когда мы говорили об усилителе мощности, мы вспоминали радиаторы (или теплоотводы), как нечто само собой разумеющееся, но это не так. Как резисторы бывают разной точности, разных номиналов и мощности, которые рассчитываются в процессе разработки устройства, так и теплоотводы рассчитываются для рассеивания определенной мощности.

Разные режимы работы оконечных каскадов

Вернемся к оконечному каскаду усилителя мощности и постараемся понять, что получается с мощностью, выделяемой на коллекторе транзистора, в режимах B, AB и A. Вернемся к схеме, например, рисунка 8.17. Все, что мы получим в этой главе, в равной мере может относиться не только к усилителям низкой частоты, но к любым усилителям, только при других частотах могут появиться особенности, которые должно соответствующим образом учитывать. Упростим схему для наших целей и добавим амперметр Pr1 и вольтметр PR2.

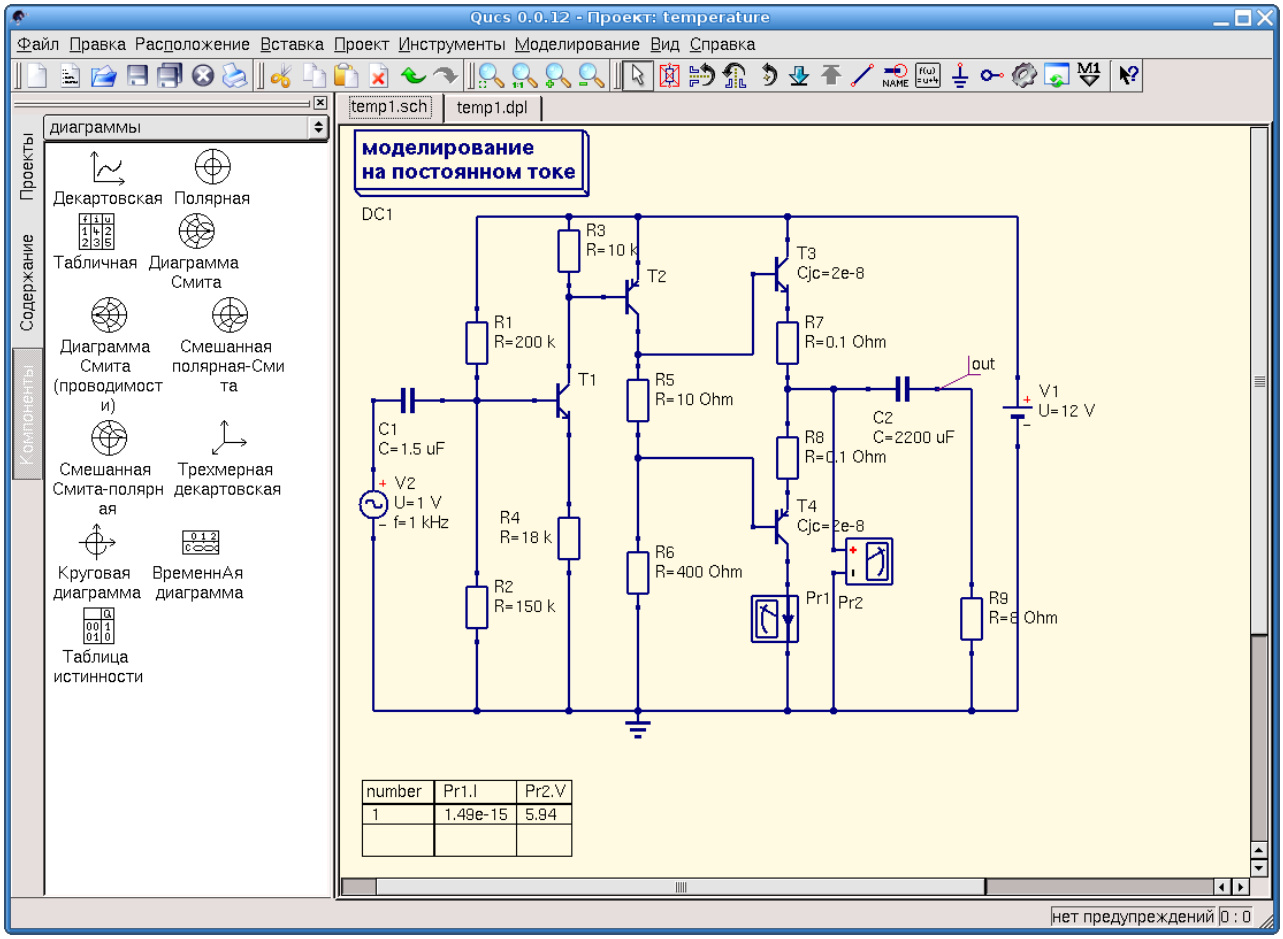


Рис. 9.1. Усилитель мощности в режиме B

Ток, протекающий через выходные транзисторы T3 и T4, установлен настолько маленьким, что можно считать, что его нет в отсутствии сигнала. Постоянное напряжение на выходе усилителя регулируется так, чтобы оно было равно половине питающего напряжения. Цепь общей отрицательной обратной связи ликвидирована для устранения влияния на параметры усилителя. Остались только местные обратные связи. Посмотрим, как выглядит сигнал на выходе такого усилителя.

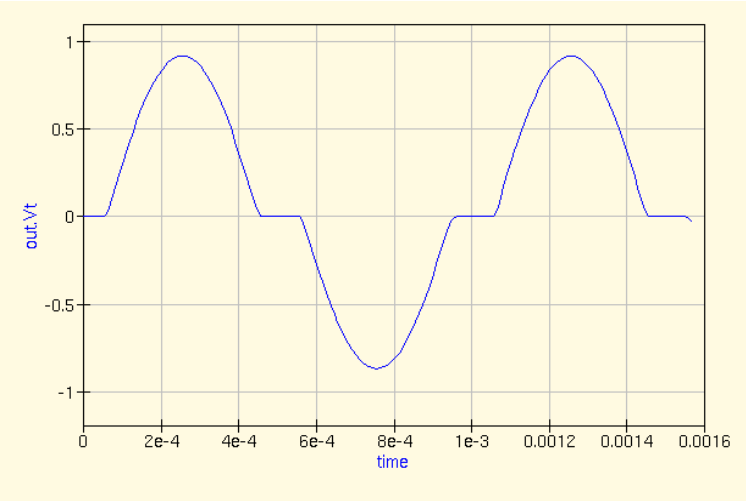


Рис. 9.2. Сигнал на выходе усилителя в режиме B

В подобном режиме работы каждый из выходных транзисторов открывается в свой полу-период, отрабатывая свою полу-волну сигнала. Характерным в этом случае является наличие искажений, названных «ступенькой». Пока входной сигнал не достигает при своем изменении некоторой величины относительно половины питающего напряжения, увеличивается ли он, или уменьшается, транзистор остается закрыт, ток через него в нагрузку не протекает, что и вносит характерные искажения.

Если мы умножим показания амперметра на показания вольтметра в отсутствии сигнала, то есть, при измерении на постоянном токе, а эти значения показаны в таблице на рисунке 9.1, то мы получим... мой калькулятор показывает ноль. То есть, в этом режиме на постоянном токе в отсутствии сигнала мощность на коллекторах транзисторов не рассеивается.

Вместе с тем, при отдаче полной мощности в нагрузку равную 8 Ом (схема без «упрощений» с питающим напряжением 12 В), сигнал имеет амплитуду 5 В.

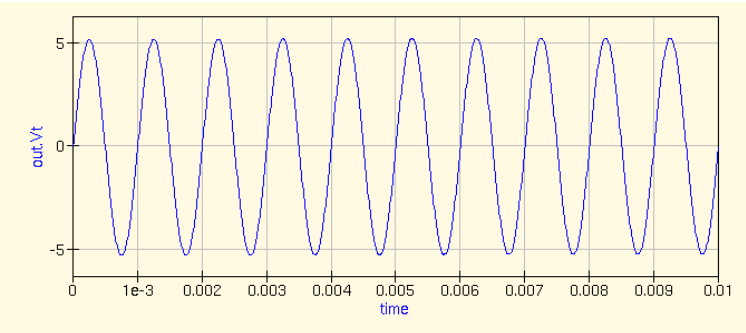


Рис. 9.3. Сигнал на выходе усилителя при полной мощности

Мы можем рассчитать пиковый ток, разделив амплитуду на 8 Ом, что даст нам ток 0.63 А, и мощность на коллекторе транзистора равную произведению 1 В (разность между половиной напряжения питания и амплитудой сигнала) на этот ток, то есть, 0.63 Вт. Пиковая мощность в нагрузке, однако, равна 5*0.63 = 3.2 Вт. Но это не полная информация о мощности, рассеиваемой на коллекторе каждого из транзисторов. Чтобы яснее понять, что же происходит с мощностью, воспользуемся теми преимуществами, что дает программа Qucs. Немного переделав схему измерений, добавив к режимам моделирования уравнение, которое

должно (как мне кажется) показывать текущую мощность, можно получить следующий результат моделирования.

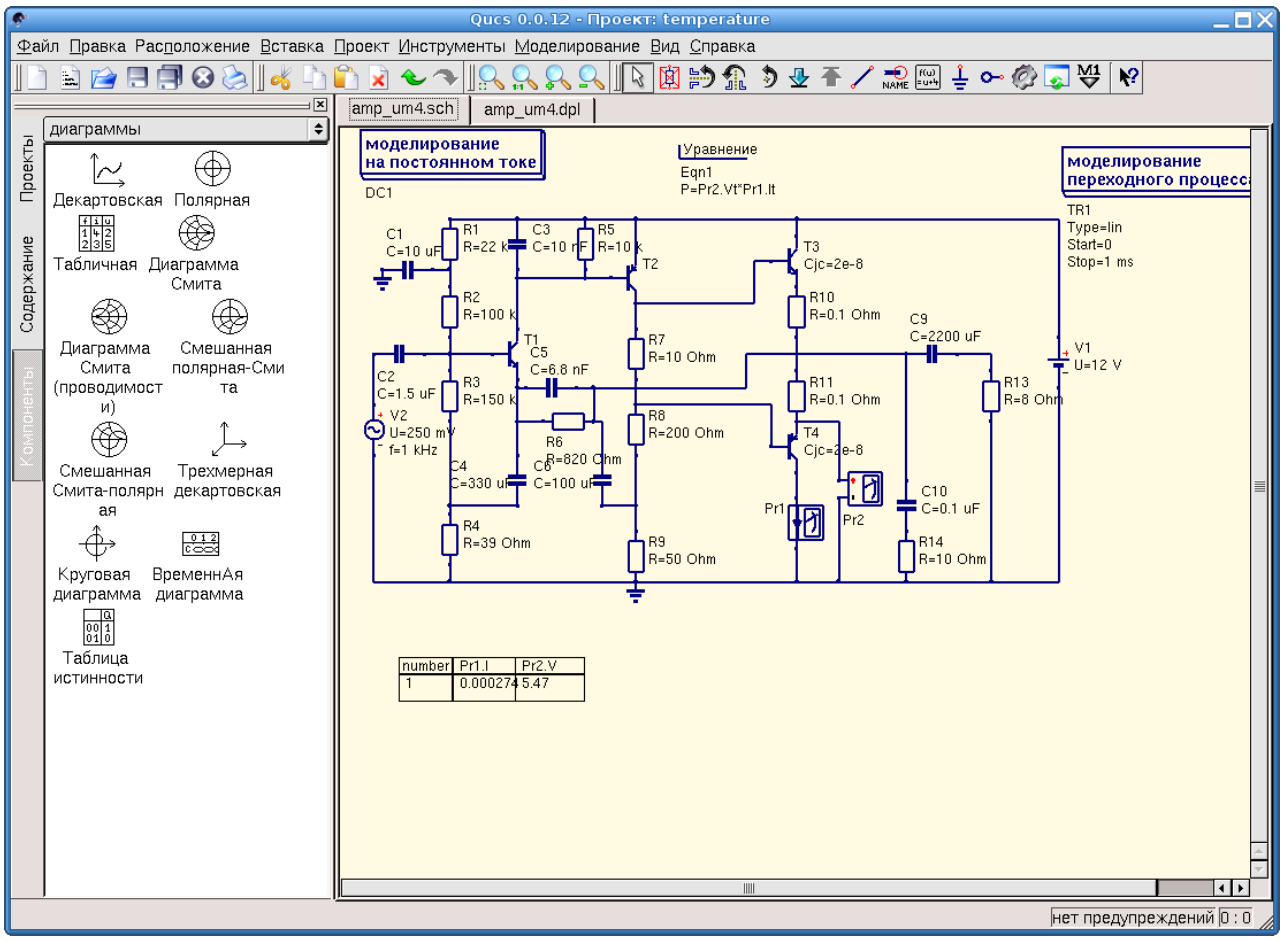


Рис. 9.4. Схема усилителя мощности для оценки мощности рассеивания

Ток и напряжение при отсутствии сигнала показывают, что усилитель работает в режиме B. А диаграммы ниже показывают сигнал на выходе усилителя (верхняя диаграмма) и мощность рассеиваемую на транзисторе.

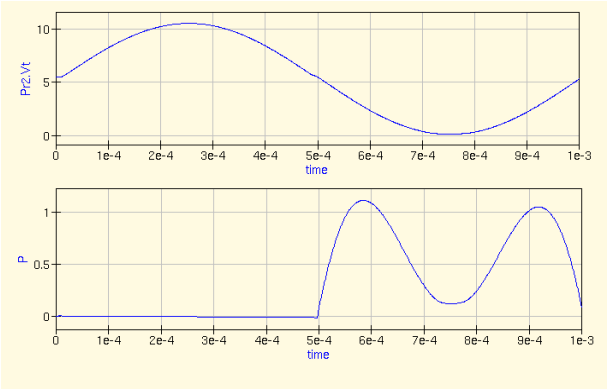


Рис. 9.5. Диаграммы сигнала и мощности на транзисторе T4

На нижней диаграмме видно, что половину периода транзистор «отдыхает», а это уменьшает рассеиваемую на нем мощность в два раза. Кроме того, видно, что мощность,

рассеиваемая на транзисторе в тот момент, когда сигнал достигает своего пика, не является максимальной. Можно перенести приборы на транзистор T3, что даст похожие диаграммы.

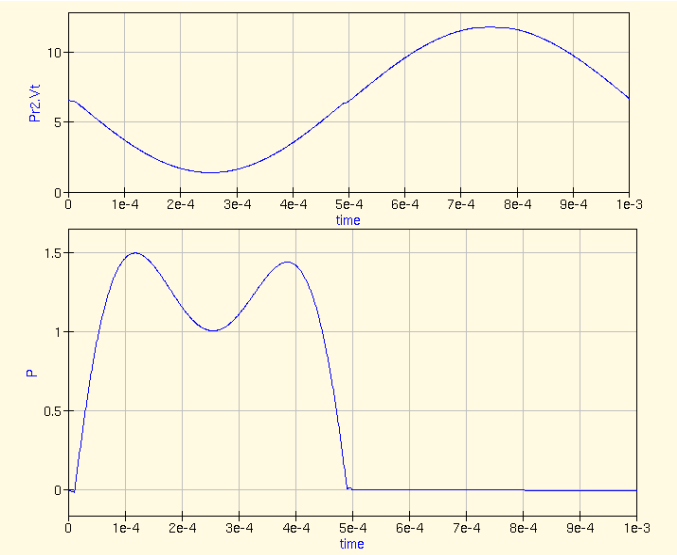


Рис. 9.6. Диаграммы сигнала и мощности на транзисторе T3

Наибольшая рассеиваемая мощность, если судить по диаграмме, составляет 1.5 Вт. В данном случае для расчета мощности можно взять 1.5 Вт и уменьшить это значение в два раза, что даст 0.75 Вт. Многие мощные транзисторы позволяют использовать их без радиатора при такой мощности. Но если вы планируете, как мы и обсуждали выше, провести ряд экспериментов, и, в частности, увеличивать мощность за счет увеличения напряжения питания, то следует принять во внимание большую мощность рассеяния на транзисторах и рассчитать необходимые теплоотводы для них. Попробуем понять, как работают радиаторы, и рассчитать их для мощности рассеяния на транзисторе в 5 Вт. Но прежде, совсем я об этом забыл, переведем наш усилитель в режим AB. Для этого достаточно увеличить резистор R7 до 100 Ом. Ток в отсутствии сигнала увеличится до значения в 10% от пикового. Посмотрим, что происходит с усилителем в этом режиме.

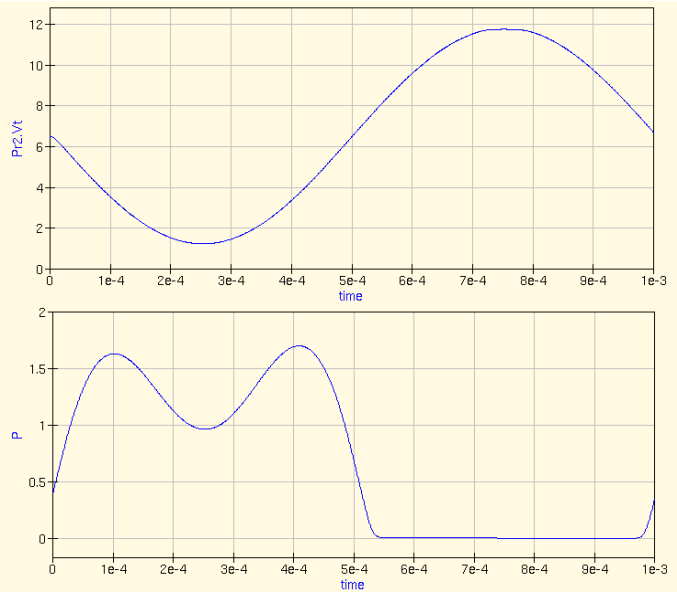


Рис. 9.7. Сигнал и мощность в режиме AB

В режиме AB ток через выходные транзисторы протекает и в отсутствии сигнала.

Рассеиваемая мощность возрастает, транзистор перестает отдыхать половину периода, ему приходится «подрабатывать». Зачем нужен перевод транзисторов в режим работы АВ? Этот режим позволяет избавиться от характерной для режима В «ступеньки», что уменьшает суммарные нелинейные искажения. Дальнейшее увеличение тока переведет режим работы усилителя из АВ в А.

Тепло. Что с ним делать?

Выходные транзисторы усилителя мощности греются, тем больше, чем больше рассеиваемая на них мощность. Кристалл, основа транзистора, спрятан внутри корпуса, он может оставаться работоспособен только до тех пор, пока его температура не превысит некоторого предела. Эта температуру можно найти в справочных параметрах транзисторов, так для транзистора КТ602 она составляет 393^0К или 120^0С (по шкале Цельсия, формула перехода выглядит так: С = К - 273^0), что характерно для кремниевых транзисторов. Измерить температуру перехода, не ломая транзистор, мы не можем, поэтому для транзисторов, впрочем как и для других полупроводниковых приборов, в справочных данных приводят параметр, который связывает температуру корпуса, ее мы можем измерять, и температуру перехода. Очевидно, что температура корпуса ниже температуры перехода, что не следует забывать. Тепловые расчеты и рассмотрение тепловых процессов удобно проводить с использованием модели, основанной на аналогии между электрической схемой и тепловой. Действительно, тепло, определяемое мощностью, растекается по всем элементам тепловой схемы подобно току, протекающему по электрической цепи. Перепады температуры, как разница между температурой перехода и температурой корпуса транзистора, подобны падению напряжения. А, значит, мы можем, как в законе Ома, определить тепловое сопротивление, по которому протекает тепловой ток, зная перепад температур. Или, зная тепловое сопротивление, рассчитать перепад температур. Точные расчеты достаточно сложны, хотя можно использовать программы для этих расчетов. Конечной целью подобных расчетов будет выбор теплоотвода, если мы знаем мощность, рассеиваемую на полупроводниковом компоненте схемы. Для отвода тепла используются радиаторы. Простейший из них – это вертикально установленная на плате металлическая пластина, на которой закреплен, например, транзистор. Выделяемое транзистором тепло растекается по пластине, соприкасающейся с воздухом, а последний при нагреве поднимается вверх, унося с собой тепло и освобождая место для более холодного притекающего к пластине воздуха. Чем больше поверхность пластины, тем лучше происходит процесс охлаждения. Но не следует забывать, что тепло «растекается» по пластине с некоторой скоростью, а, значит, в пластине образуется область, где температура выше, чем на краях пластины, и в итоге края могут не достигать температуры этой области, а могут и не принимая участия в процессе охлаждения. Сделав радиатор очень большим мы не достигнем поставленной цели, но лишь увеличим размеры устройства.

В качестве материала для изготовления радиаторов чаще используют алюминий и его сплавы, реже медь или другие материалы. Для увеличения поверхности контакта с воздухом при сохранении разумных габаритов пластинчатый радиатор складывают, изготавливают из нескольких сложенных пластин или делают его поверхность ребристой и игольчатой. В последнее время часто применяют принудительное охлаждение с помощью вентиляторов. Радиаторы для такого вида охлаждения делают тоже специальным образом, у ребристых радиаторов ребра располагаются значительно ближе друг к другу. Если охлаждение происходит при естественном движении воздуха, то в такой конструкции воздух не будет проникать между ребрами теплоотвода, значительно уменьшая его эффективность.

На месте стыков: кристалл-корпус, корпус-радиатор, радиатор-воздух, – образуются

тепловые сопротивления, которые можно найти в справочной литературе. Но далеко не всегда. Я листаю справочник по транзисторам, хороший справочник, к которому я привык, но не для всех транзисторов нахожу такой важный параметр, как тепловое сопротивление переход-корпус. Для начинающих, как мне кажется, лучше использовать те рекомендации, которые, как правило, присутствуют в описании схемы усилителя. Хотя, чтобы лучше представлять, как все происходит в действительности, можно использовать программы, можно провести ряд экспериментов – многие мультиметры позволяют измерять температуру, а имея в своем распоряжении транзистор, радиатор и источник питания, можно проверить, например, как влияет способ крепления транзистора к радиатору на разницу в температурах корпуса и радиатора. Обычно для транзистора на теплоотводе фрезеруют место установки, которое затем шлифуют для улучшения теплового контакта. С этой же целью применяют специальные пасты, которые наносят на место установки транзистора. Вы можете посмотреть, как влияет на температуру корпуса способ крепления транзистора, начав эксперимент с установки транзистора на не шлифованный участок, продолжив его перемещением в шлифованную область, и закончить опыт нанесением теплопроводящей пасты на место установки транзистора.

Программа Qucs позволяет рассмотреть тепловую модель, если принять, что есть соответствие:

- температура эквивалентна напряжению;
- теплота (мощность) эквивалентна току;
- тепловое сопротивление эквивалентно сопротивлению.

Для создания такой модели потребуется знание мощности, рассеиваемой на транзисторе. Примем ее равной 5 Вт. Нужно знать тепловое сопротивление переход-корпус. Для транзистора П701 (что нашлось в справочнике) это 10^0К/Вт, а для транзистора П702 – 2.5^0К/Вт. Такое же тепловое сопротивление для транзистора КТ802. И еще одно тепловое сопротивление, которое потребуется, это тепловое сопротивление радиатора. Его можно рассчитать по приближенной формуле: Rт.р. ~ $1/\acute{a}_т S_р$, а если принять во внимание, что оно много меньше, чем тепловое сопротивление корпус-среда, то оно и определит второй компонент схемы. В формуле $\acute{a}_т$ = 3 вт/м^2*град, коэффициент теплоотдачи радиатора при естественном охлаждении. А $S_р$ – полная площадь поверхности радиатора в м^2 с учетом обеих сторон и всех поверхностей ребер, если это ребристый радиатор. Самый простой приближенный расчет: площадь радиатора определяется условием, на каждый 1 Вт рассеиваемой мощности должно приходиться 30 см^2 поверхности радиатора. Посмотрим, какие температуры мы получим, если для 5 Вт возьмем радиатор 150 см^2 (0.015 м^2). В этом случае Rт.р. = 22 град/Вт. Если принять изменения температуры окружающей среды в диапазоне 10-40 градусов, то можно построить тепловую модель по этим данным.

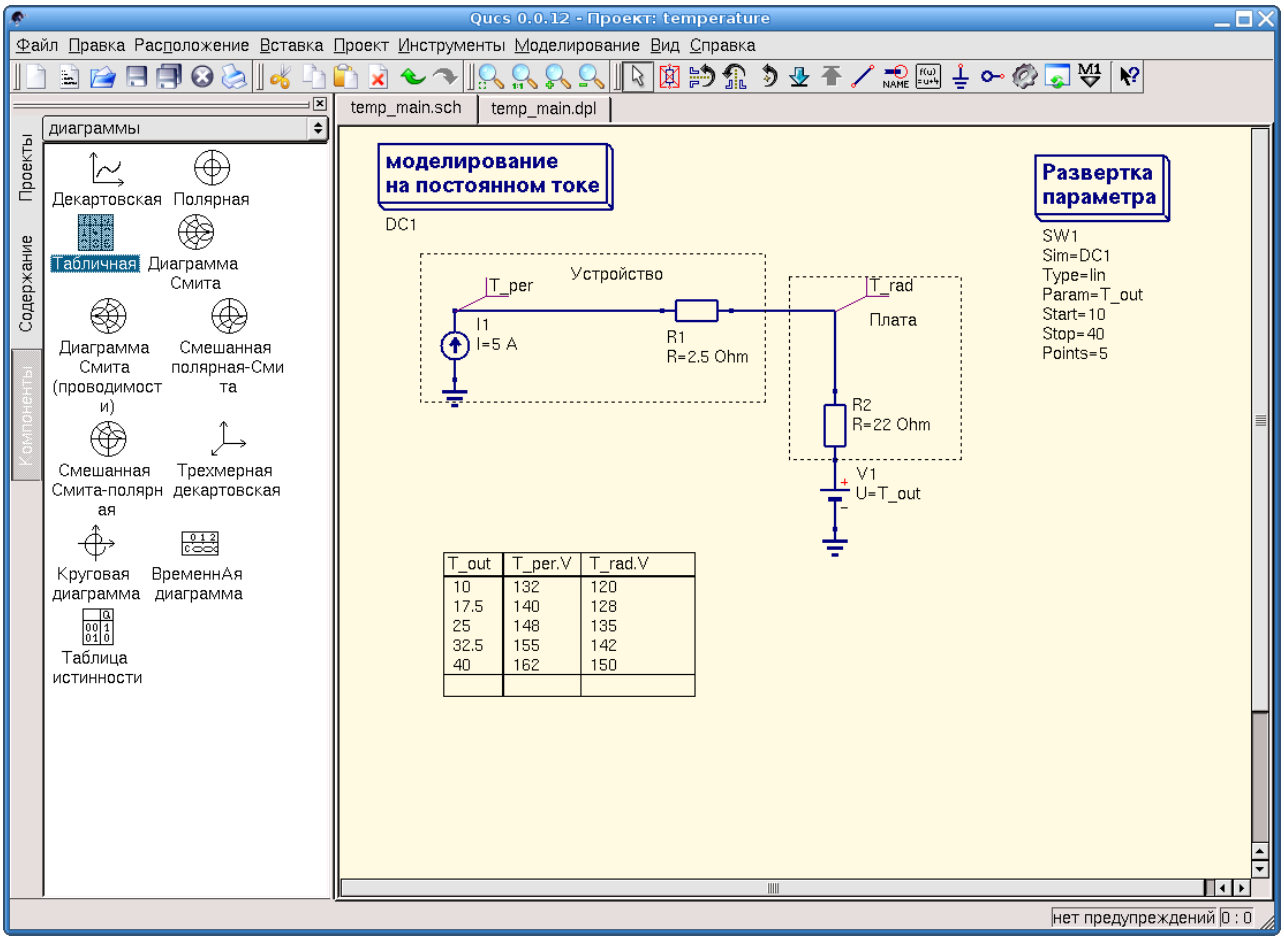


Рис. 9.8. Тепловая схема в программе Qucs

Таким образом, если я ничего не напутал, то при заданной площади радиатора кристалл будет перегреваться. Попробуем увеличить поверхность радиатора вдвое до 300 см^2. Новая таблица температур получается следующей.

T_out	T_per.V	T_rad.V
10	77.5	65
17.5	85	72.5
25	92.5	80
32.5	100	87.5
40	108	95

Рис. 9.9. Температуры при увеличении радиатора вдвое

Теперь температура радиатора не превосходит 95 градусов, а температура перехода 108. Эта таблица соответствует радиатору с поверхностью 300 см^2, что получится, если взять алюминиевую пластину толщиной, положим, 2 мм и с размерами 10x15 см, и придать ей «П» образную форму. Меньшие габариты даст ребристый радиатор. Но мне интересно, а что получится, если тепловое сопротивление переход-корпус будет 10 град/ватт?

T_out	T_per.V	T_rad.V
10	115	65
17.5	122	72.5
25	130	80
32.5	138	87.5
40	145	95

Рис. 9.10. Таблица температур для транзистора П701

Для этого транзистора поверхность теплоотвода потребовалось бы увеличить еще больше. Таким образом, для расчета теплоотвода следует знать тепловые параметры транзистора и тепловые параметры радиатора. Они сильно влияют на конечный результат, и без их точного учета транзистор можно вывести из строя.

Очень часто при установке транзистора на радиатор его требуется электрически изолировать, например, при установке пары транзисторов на один теплоотвод, который используют зачастую и как заднюю стенку конструктива. Изолятор, конечно, ухудшает условия отвода тепла, и это тоже следует учитывать. Мощные транзисторы в металлическом корпусе крепят к радиатору с помощью специальной шайбы и винтов. Закручивая винты, следует внимательно смотреть за тем, чтобы корпус транзистора не был перекошен, иначе образуется воздушный зазор, а воздух плохой проводник тепла. В некоторых случаях кроме транзисторов оконечного каскада в установке на радиаторы нуждаются транзисторы (или транзистор) предоконечного каскада.

И последнее, о чем следует упомянуть, не имея опыта создания усилителей, будь то УМЗЧ или мощный выходной каскад ВЧ устройства, многие стараются выбрать максимально достижимые параметры, не задумываясь о тех трудностях, которые их подстерегают на пути реализации желаемого. Сами трудности не представляют опасности для начинающего, если он готов к любым неожиданностям, опасность представляют разочарования. Они могут отвратить начинающего от продолжения работы с электроникой, если он не примет во внимание, что то устройство, которое он хотел быстро и легко собрать, может вызывать серьезные затруднения и у профессионалов, долгое время работающих в этой области. Каждый ватт дополнительной мощности, каждый килогерц (или мегагерц, или километр расстояния), расширяющий область частот, даются, порой, большими усилиями. А некоторые задачи требуют для своего решения смены и технологий, и создания новых материалов, словом, серьезной научно-исследовательской работы. Работы во многом схожей с теми экспериментами, которые предлагались выше. Надеюсь, что проделав их, вы яснее сможете сформулировать свои требования, яснее увидеть цель, составить реальный план своей работы, которую выполните с тем большим успехом, чем больше цель будет соответствовать вашим возможностям. В любом случае не забывайте, что ток, протекая по любому сопротивлению, выделяет некоторую мощность в виде тепла, которое следует отводить от радио-элементов, особенно полупроводниковых.

Глава 10. Сигналы, немного больше

То, что мы наблюдаем с помощью осциллографа при создании или ремонте электронных устройств, может быть полезно, если мы знаем, какой вид сигнал должен иметь в этом месте электрической цепи, и будет бесполезно, если то, что мы видим, никак не соотносится с нашими знаниями о схеме. Осциллограмма может помочь решить проблему, но может создать дополнительные трудности при полном ее непонимании или неправильной трактовке увиденного. Следовательно, прежде чем приступать к исследованию схемы, следует представлять ее работу хотя бы в общих чертах, особенно в части, относящейся к наблюдениям с помощью осциллографа. Осциллограф особенно удобен для наблюдения периодических сигналов. Наблюдение непериодических сигналов значительно труднее.

Прямоугольные импульсы

При работе со схемами усилителей звуковой частоты чаще других используют синусоидальные сигналы. Причина, по которой это происходит, связана с математикой. Функция синуса относится к элементарным, хорошо изученным и не разлагающимся на более простые. Например, имея дело с сигналом прямоугольной формы, мы можем его представить, используя разложение Фурье, в виде совокупности (пусть даже бесконечного числа) элементарных тригонометрических функций. Такое представление любых сигналов, когда мы работаем с усилителями, позволяет рассматривать нелинейные искажения, если разложить сигнал на выходе усилителя на составляющие, набор которых может даже помочь в выявлении причин искажения сигнала. Такой метод называют еще гармоническим Фурье анализом. Название «гармонический анализ» указывает на то, что при разложении исходной функции использовались функции синуса и косинуса. Само преобразование не требует этого, мало того, могут быть использованы любые функции, удовлетворяющие определенным условиям. Но в работе с электрическими цепями гармонический анализ стал общепринятым методом анализа нелинейных искажений. Ниже в этой главе я постараюсь представить некоторые практические эксперименты, которые помогли бы математический аспект преобразований Фурье перенести в плоскость работы с электрическими схемами, и именно в практическую плоскость.

Первый эксперимент касается оценки частотных, заметьте не нелинейных, а частотных, искажений, возникающих в усилителе с помощью генератора прямоугольных импульсов.

Посмотрим, как реагируют на прохождение прямоугольных импульсов разные электрические RC цепи (это же относится и к LC, и к LR, и к LCR цепям).

Для этого в программе Qucs мы будем пользоваться источником прямоугольных импульсов и такими компонентами, как резисторы, конденсаторы и индуктивности. Если в вашем арсенале приборов есть осциллограф, генератор прямоугольных импульсов или функциональный генератор, то было бы очень полезно повторить эти простые опыты на макетной плате.

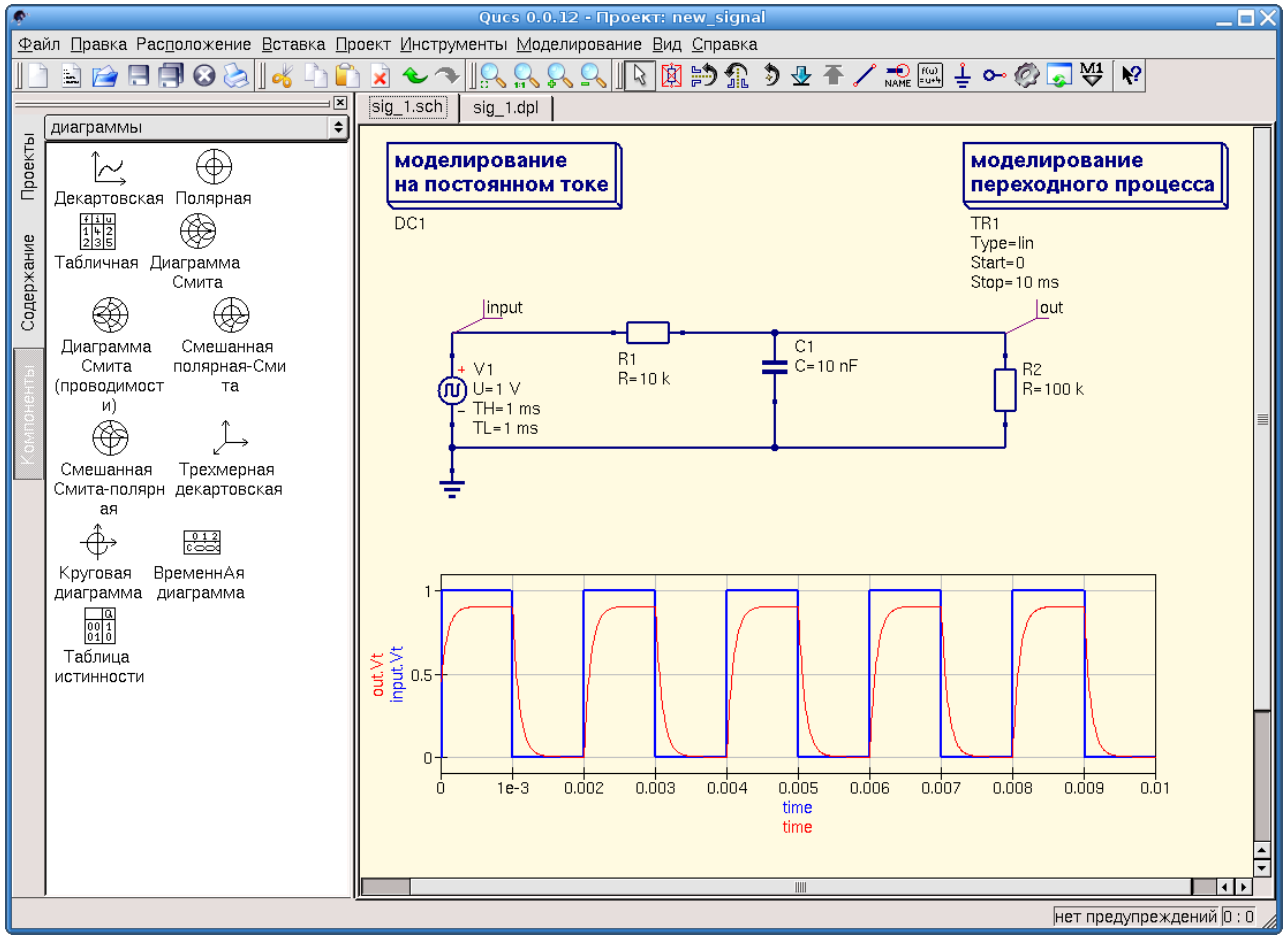


Рис. 10.1. Наблюдение сигнала прямоугольной формы в RC цепи

На рисунке представлена обычная *интегрирующая RC цепочка* из резистора R1 и конденсатора C1. Сопротивление R2 — это сопротивление нагрузки, скажем, входное сопротивление осциллографа. Когда мы говорили о частотных свойствах усилителей, рассматривали амплитудно-частотные характеристики каскадов усилителя, то каждый из них можно было рассматривать как идеальный, не частотно-зависимый усилитель, к которому добавляется подобная эквивалентная RC цепь.

Очень красивые прямоугольные импульсы на входе цепи (*input*) на выходе (*output*) превращаются в не менее красивые фигуры, но их уже нельзя назвать прямоугольниками. «Затянутые» фронты сигнала прямоугольной формы, и передний и задний, свидетельствуют именно о наличии интегрирующей RC цепи. Мы можем двояко рассматривать этот процесс. Как переходной процесс, связанный с зарядом конденсатора при переходе испытательного сигнала от нуля к заданному напряжению, в данном случае к одному вольту, когда в начальный момент времени конденсатор полностью пропускает ток, как бы «закорачивая» выход, но по мере своего заряда перестает пропускать ток, а напряжение на нем возрастает до величины, обусловленной делителем напряжения R1R2. При обратном переходе испытательного сигнала разряд конденсатора «затягивает» процесс перехода к нулевому напряжению. И мы можем рассматривать этот процесс с другой точки зрения. Изменение формы сигнала на выходе происходит под влиянием RC фильтра (R1C1), который «срезает» высокие частоты. Делитель переменного напряжения, коэффициент деления, R1C1 зависит от частоты: на низких частотах это одно значение, на высоких, другое. Изменение величины высокочастотных составляющих в спектре исходного сигнала, их уменьшение, и вызывает

изменение формы выходного сигнала.

Для грубой оценки частотных свойств испытуемого устройства можно считать, что если на экране осциллографа вы наблюдаете прямоугольные импульсы с частотой, положим, 1 кГц, то ваше устройство вполне успешно работает до частоты 10 кГц. Это утверждение не бесспорно, советую вам проверить его либо в программе Qucs, либо на макетной плате, используя вначале RC цепи и испытывая их прямоугольными импульсами с последующим построением АЧХ цепи, а затем используя однокаскадные усилители. Практика подобных наблюдений поможет вам в последствии быстрее оценить частотные свойства реальных устройств.

А теперь поменяем местами резистор R1 и конденсатор C1, сигналы примут вид:

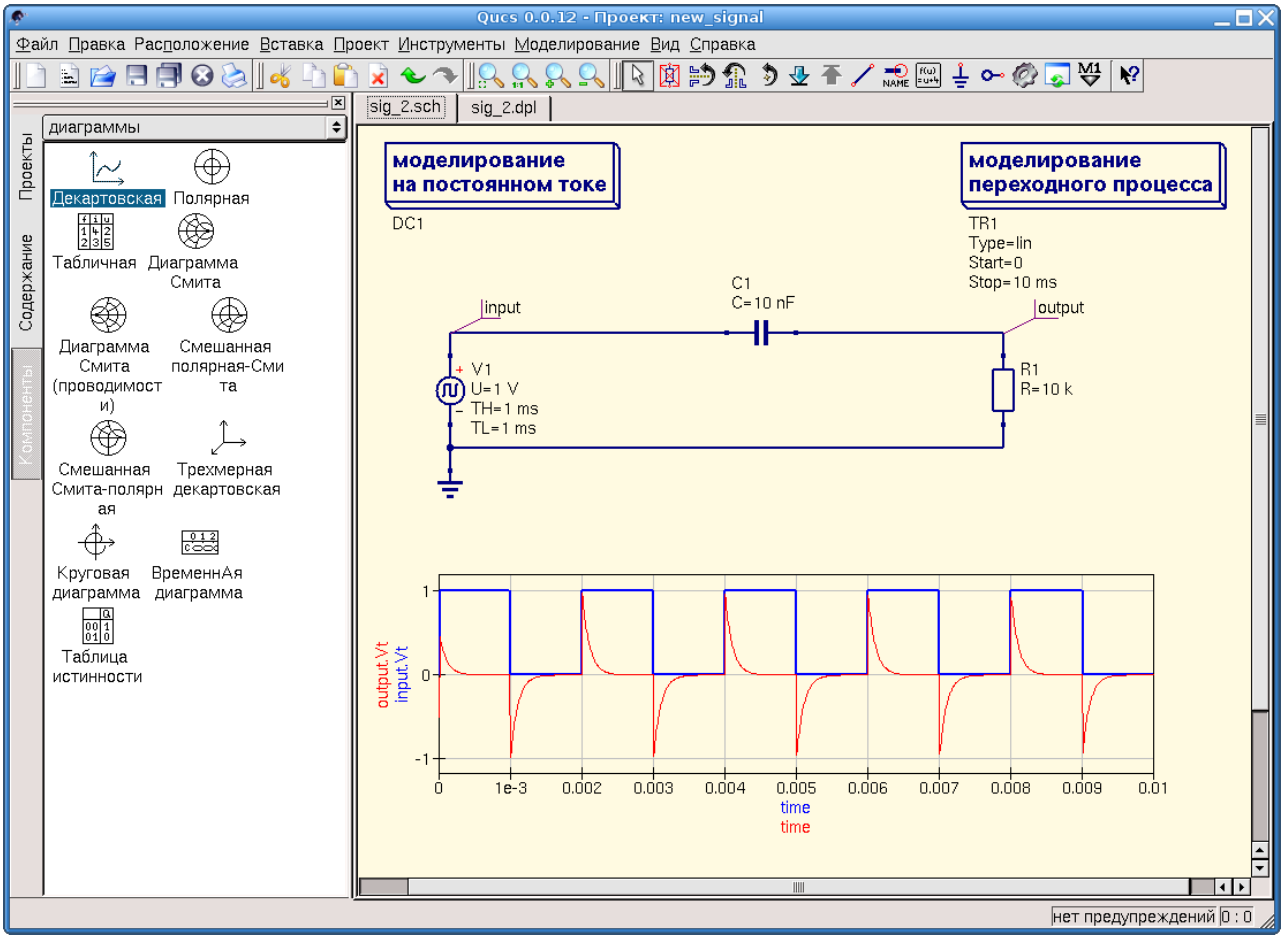


Рис. 10.2. Прохождение прямоугольных импульсов через дифференцирующую цепь

Да, такая RC цепь называется *дифференцирующей*. И выходной сигнал (*output*), как можно видеть, отличается и от исходного сигнала, и от предыдущего. Дифференцирующая цепь как бы укоротила импульсы, превратив их в короткие и остроконечные. И в этом случае рассматривать процесс можно с двух точек зрения, как и предыдущий. Только теперь фильтр R1C1 «срезает» низкие частоты, нехватка которых в выходном сигнале обуславливает его форму. Касательно же переходного процесса, в начальный момент времени, когда испытательный сигнал переходит с нуля к напряжению в один вольт, конденсатор пропускает ток, напряжение на резисторе R1 получается равным одному вольту, но следом идет заряд конденсатора, после которого конденсатор перестает пропускать постоянный ток. За время от нуля до 1 мС (1e-3 на диаграмме) источник напряжения V1 представляет собой источник

постоянного напряжения равного 1 В.

А вот, что получится, если прямоугольные импульсы попадут в LC цепь.

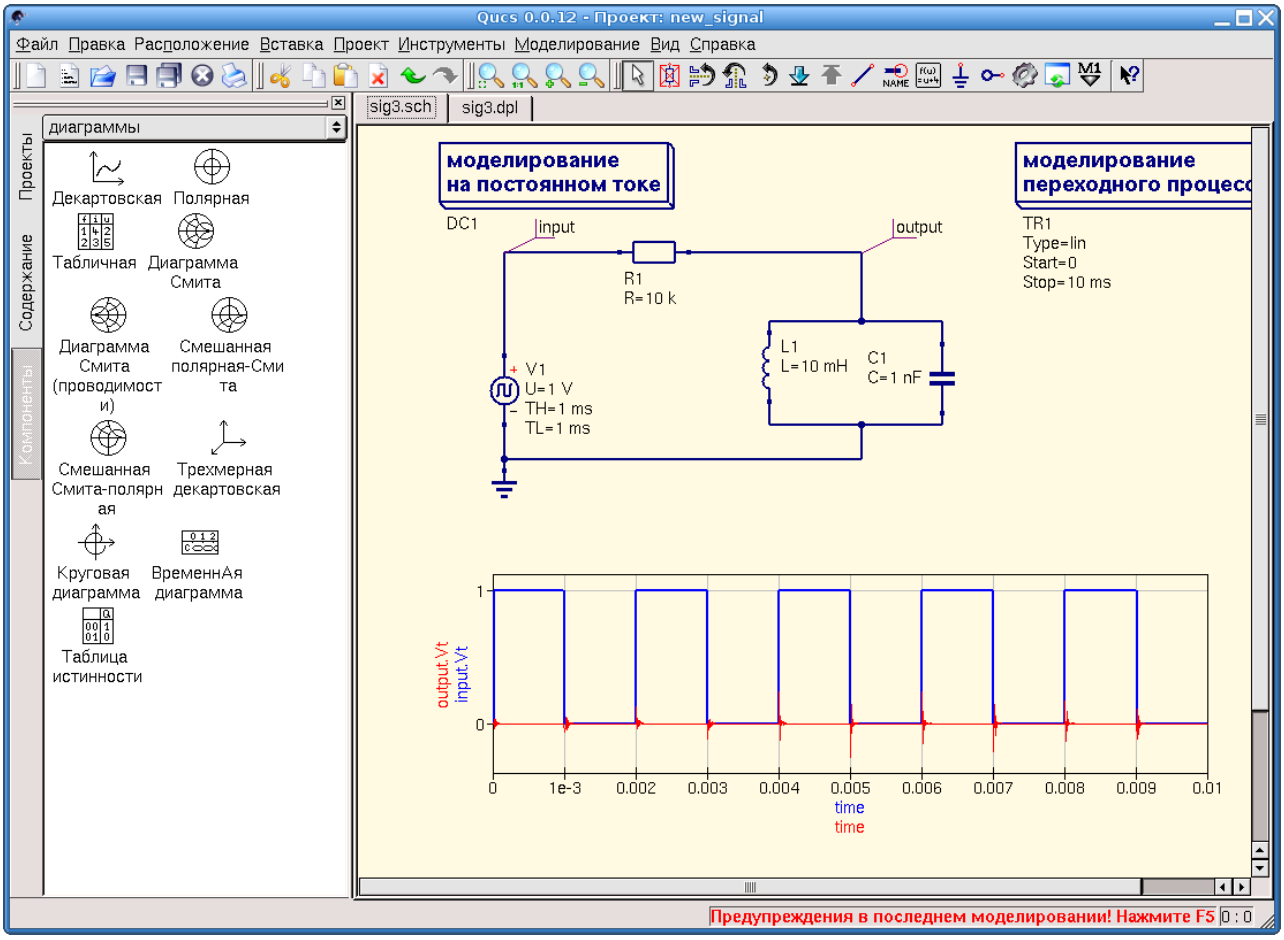


Рис. 10.3. LC контур и прямоугольные импульсы

Я нарисовал цепь, выделив ее основные (для рассказа) компоненты: L и C.

Выходной сигнал (output) на диаграмме почти не виден. Но, что там с ним... как-то он размыт, что ли? Уберем входной сигнал и выделим только его.

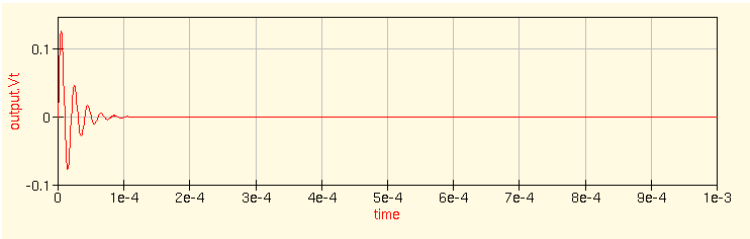


Рис. 10.4. Автоколебания в LC контуре

Такой сигнал возникает в LC цепи с параллельно соединенными элементами в тот момент, когда входной сигнал изменяется от нуля до максимального. Он очень похож на затухающий синусоидальный сигнал, которым и является. Частота этого сигнала определяется величиной индуктивности и емкости и носит название собственной частоты колебаний LC контура. Источник прямоугольного сигнала в этот момент времени ведет себя подобно цепи, состоящей из выключателя и батарейки, и именно тогда, когда выключатель включается, а

напряжение, подаваемое в цепь, меняется от нулевого до напряжения питания.

Таким образом, повторю еще раз, исследуя нашу цепь с помощью сигнала прямоугольной формы, мы можем сделать некоторые заключения о характере этой цепи. Рассмотрим диаграмму для интегрирующей цепи. В момент изменения импульса от нуля до максимума напряжение на конденсаторе C1 равно нулю, а затем, по мере заряда конденсатора, увеличивается почти до максимально напряжения. Это значение ограничено, как вы сами заметили, делителем напряжения на резисторах R1 и R2, и вы даже можете рассчитать это напряжение.

Поведение конденсатора на следующей схеме, с дифференцирующей цепочкой, иное. В начальный момент (импульс меняется от нуля до максимума) напряжение на выходе равно максимальному, а затем спадает по мере заряда конденсатора. А когда импульс переходит от максимума к нулю, заряженный им конденсатор начинает играть роль источника тока, напряжение на котором противоположно по знаку импульсу. Конденсатор начинает разряжаться через резистор R1, через который ток теперь течет в обратную сторону.

Когда мы говорили об усилителях мощности звуковой частоты, то немного внимания уделили нелинейным искажениям, возникающим в них. Нелинейные искажения «искажают» спектр входного сигнала, который, как я уже говорил, с помощью методов гармонического анализа Фурье можно представить в виде совокупности, их называют гармоническими составляющими сигнала, синусоидальных сигналов. Эти дополнительные сигналы мы воспринимаем как «обертоны» — звуки, добавляющие тембровую окраску к чистым тонам. С их помощью мы отличаем звук флейты, от звука трубы, а звучание клавесина от пианино. Чуть ниже мы попробуем не раскладывать исходный сигнал на составляющие, а наоборот, с помощью нескольких синусоидальных сигналов «собрать» сигналы другой формы. А сейчас я хотел немного добавить к рассказу об искажениях в усилителях мощности.

Транзисторные усилители кроме нелинейных характеризуются еще одним видом искажений – динамическими. Эти искажения могут возникать в транзисторном усилителе мощности при глубокой отрицательной обратной связи. Ведь, чем она глубже, тем лучше. Лучше параметры усилителя: и выше стабильность, и меньше нелинейные искажения. Но... при очень глубокой общей отрицательной обратной связи из-за запаздывания сигнала, проходящего от входа усилителя к его выходу и обратно, в какой-то момент усилитель, как бы оказывается не охвачен обратной связью, его усиление выше, и входной каскад может перегружаться. Такого рода искажения придают транзисторному усилителю характерное звучание, названное «транзисторным». Заметить и оценить эти искажение может помочь генератор прямоугольных импульсов. В момент перехода прямоугольного импульса от нуля к максимуму на выходном сигнале усилителя можно заметить резкие всплески, они-то и могут обнаружить перегрузку входного каскада.

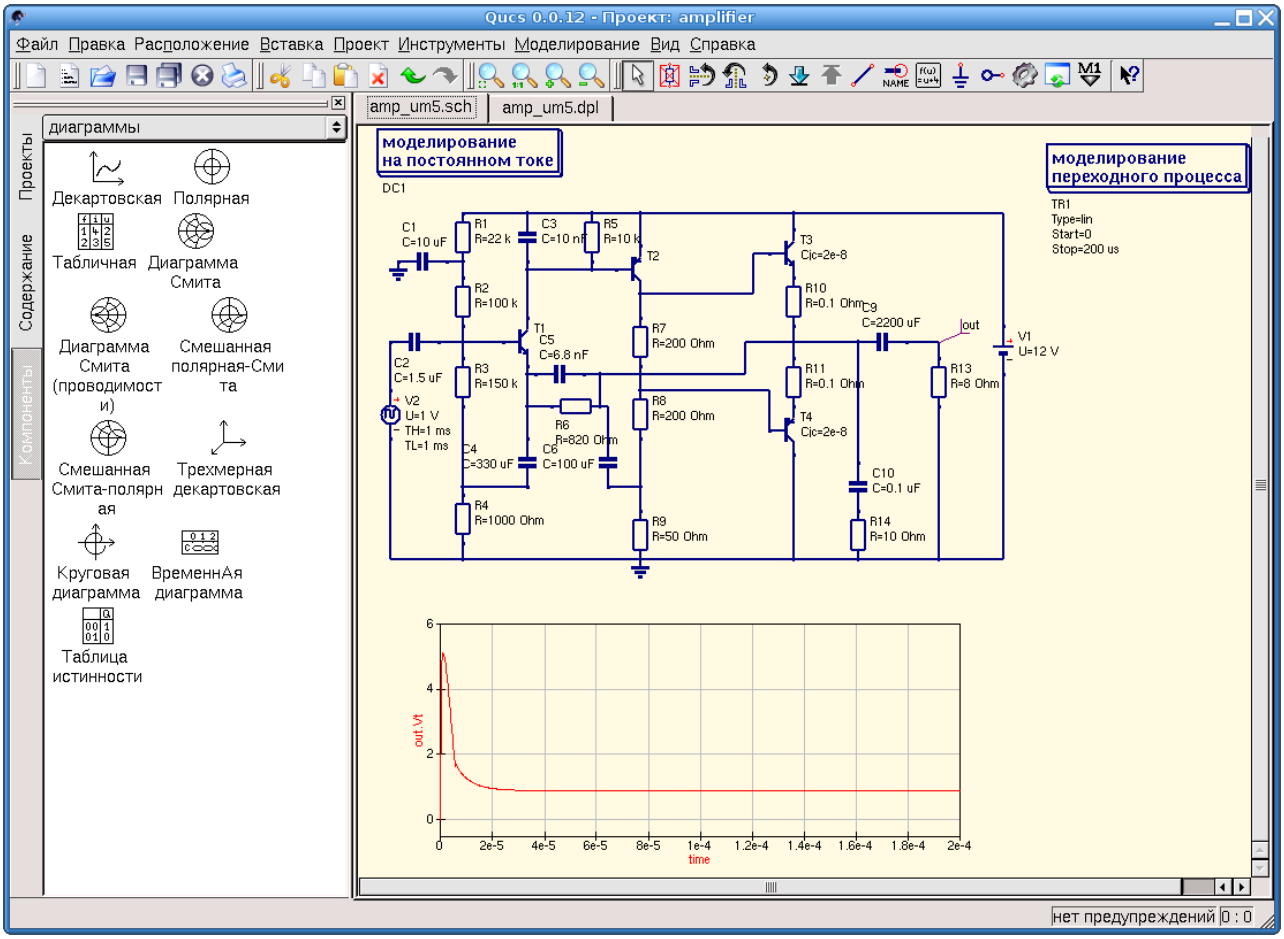


Рис. 10.5. Испытания УМЗЧ с помощью прямоугольных импульсов

Получение импульсов некоторых видов из простых сигналов

А сейчас посмотрим, верно ли было мое утверждение, что периодические импульсы разного вида можно получить складывая синусоиды с разными частотами, амплитудами и фазами, но, конечно, вполне определенные для каждого вида сигнала.

Программа Qucs позволяет использовать множество источников синусоидальных сигналов, параметры которых легко можно изменить в диалоге свойств источника. Обратите внимание на наличие такого параметра, как начальная фаза сигнала. Изменив этот параметр легко превратить синусоидальный сигнал в косинусоидальный.

Прямоугольные импульсы хорошей формы получаются от сложения большого количества составляющих. Но провести эксперимент я предлагаю с сигналом более простого вида.

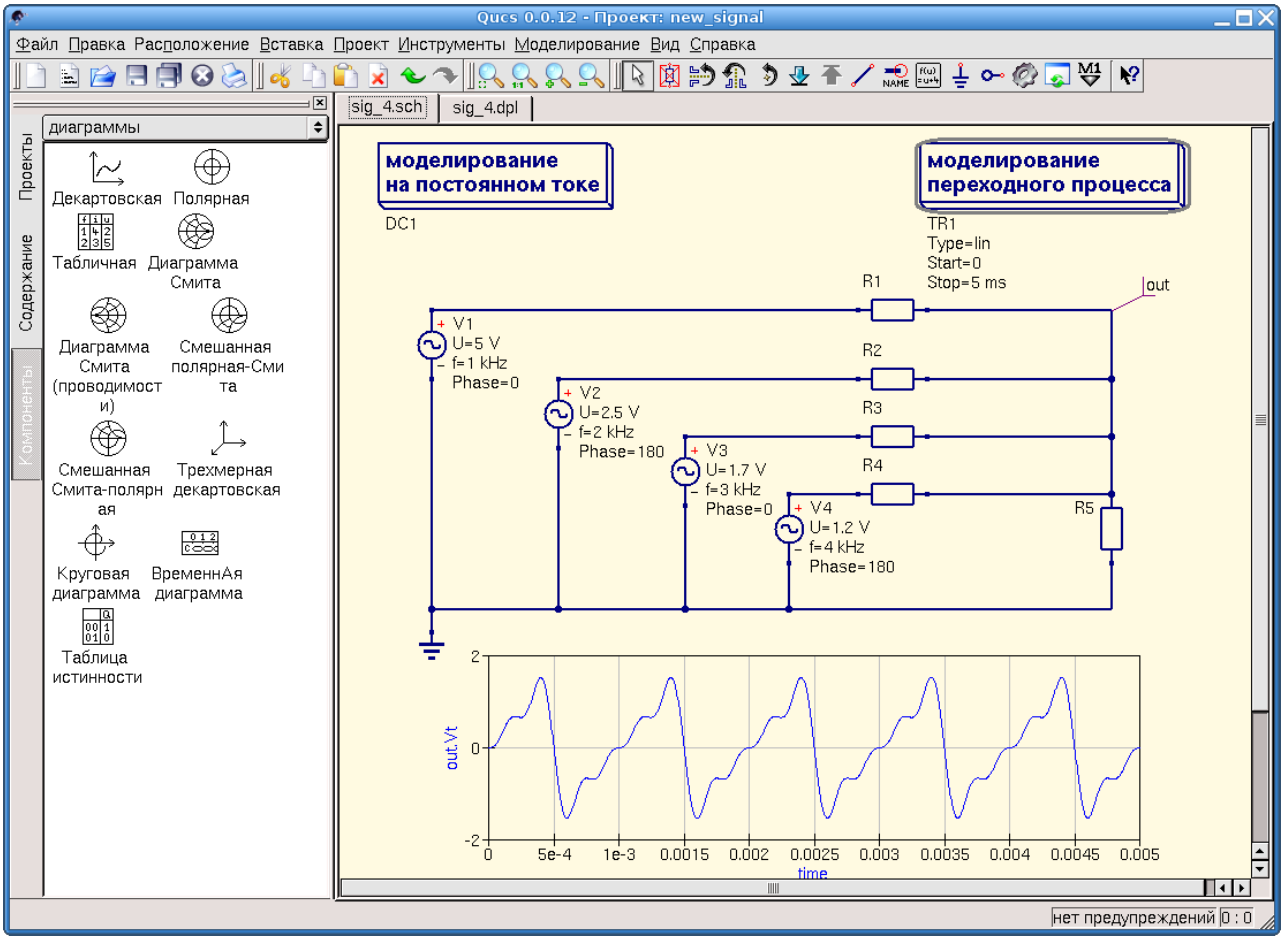


Рис. 10.6. Получение пилообразного сигнала из синусоидальных

Сигнал на диаграмме, согласитесь, по форме очень напоминает пилообразный, хотя получен всего из четырех синусоидальных сигналов. И обратите внимание на частоты, амплитуды и фазы, показанные на рисунке для генераторов (источников переменного напряжения), участвующих в эксперименте. Если мы будем увеличивать их количество, сигнал, я полагаю, все больше будет приобретать характерный пилообразный вид.

В первой книге я уже описывал подобный эксперимент, а в качестве его продолжения было предложено заменить генераторы синусоидального напряжения на генераторы прямоугольных импульсов. Подобная замена, возможно, и спекулятивная, обусловлена тем, напомню, что разложение в ряд Фурье функции произвольного вида возможно с применением не только функций синуса и косинуса, но любых, удовлетворяющих условию ортонормирования. Я не проверял это условие, не готов утверждать правомерность и математическую правильность полученного результата, но эксперимент с прямоугольными импульсами показался мне любопытным.

В этом случае вид полученного сигнала несколько меняется.

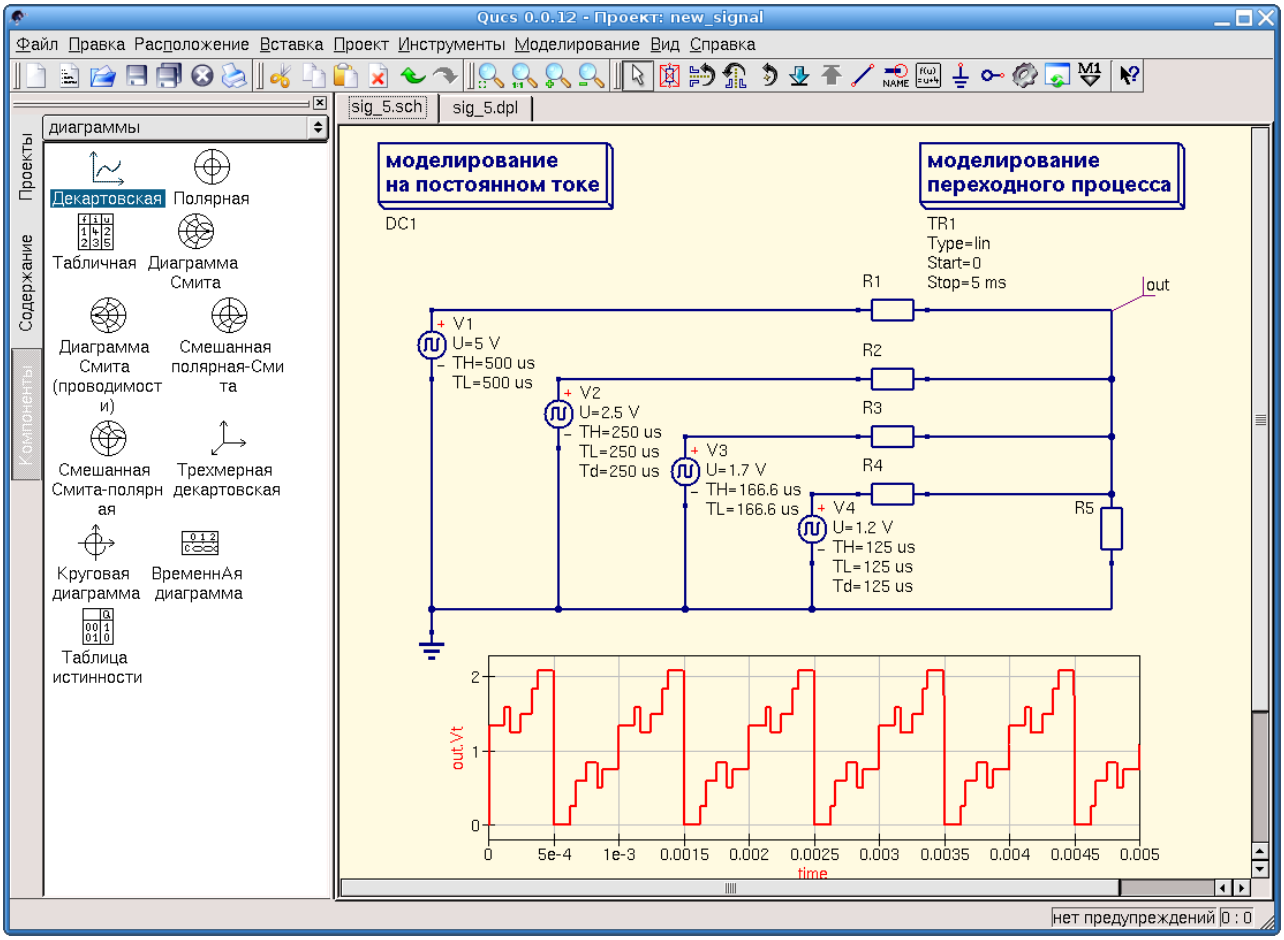


Рис. 10.7. Вид сигнала с генераторами прямоугольных импульсов

Этот вариант эксперимента я привел сейчас для того, чтобы рассказать о еще одном способе получения сигналов более сложного вида, чем прямоугольные. Как их получить в программе Qucs без построения многостраничной схемы, я не знаю. Возможно, это невозможно. Но идея, в общем-то, простая. Для получения пилообразного напряжения предположим, что в равные промежутки времени мы будем увеличивать напряжение от источника постоянного напряжения на равные, но небольшие, значения, пока не достигнем напряжения источника. После чего снизим напряжение до нуля, и будем повторять эти операции многократно.

Как бы это сделать в программе Qucs... Сейчас подумаю...

Скоро сказка сказывается, да не скоро дело делается. Ничего умнее, чем заменить генераторы периодических импульсов в предыдущей схеме на генераторы одиночных импульсов, я не придумал. Результат этой работы следующий.

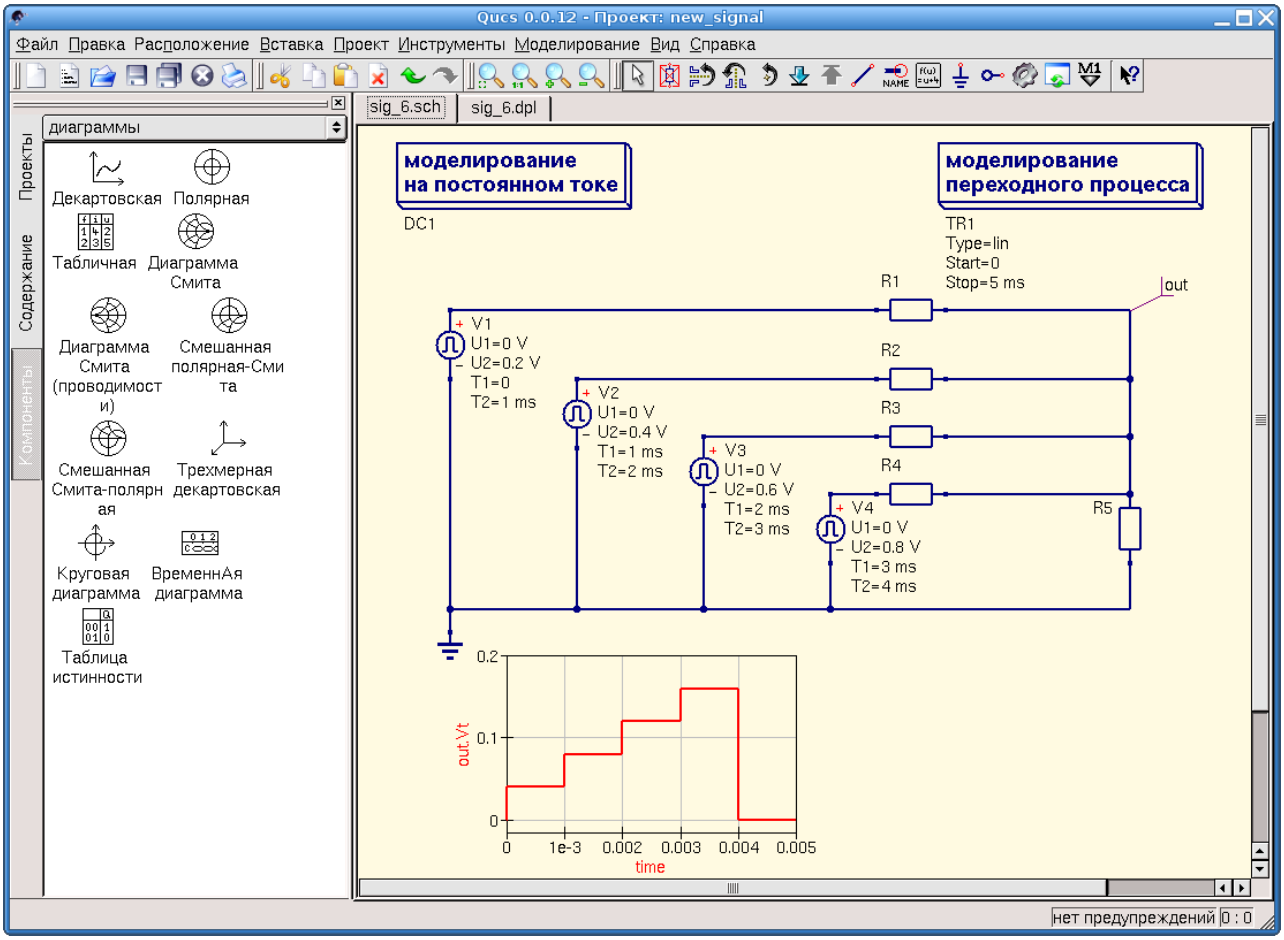


Рис. 10.8. Вид предыдущего сигнала в виде ступенек

Не самое хорошее решение, но оно дает представление о том, как форму сигнала, в сущности любого вида, можно представить ступенчатой функцией. Чем меньше ступеньки, то есть, короче шаг и меньше прирост напряжения, чем больше всего ступенек, тем ближе вид функции к заданному. Такой метод генерации сигналов помогает решить вопрос по созданию генераторов не только очень сложной формы, но может оказаться полезен и для замены генераторов простых форм, например, синусоидальных сигналов. Естественно возникает вопрос, а зачем? Зачем генератор синусоидального сигнала, который мы можем выполнить по вполне привычной схеме, создавать таким непонятным способом?

В первую очередь такой подход при создании генераторов позволяет использовать один метод для создания сигналов разных видов. Так функциональные генераторы кроме синусоидальных сигналов генерируют и треугольные сигналы, и прямоугольные сигналы. А в практической деятельности могут потребоваться сигналы и других форм, для которых хотелось бы иметь единый механизм формирования выходного сигнала.

Кроме того, представления сигнала в виде ступенчатой функции помогают справиться и с другими возникающими проблемами. Самым наглядным примером этого может служить задача создания преобразователя постоянного автомобильного напряжения (напряжения автомобильного аккумулятора) в переменное напряжение, скажем, 220 В со свойствами силового, то есть, с частотой 50 Гц. Самое простое решение, которое сразу приходит в голову, применить схему генератора синусоидального напряжения с частотой 50 Гц и добавить к нему усилитель мощности.

Но мы уже видели, что мощность, рассеиваемая на коллекторах транзисторов, связывает

нам руки по созданию усилителей мощности с любыми желаемыми выходными параметрами. Чем больше мощность, отдаваемая в нагрузку, тем сложнее решение по отводу тепла от выходных транзисторов.

Чем же в этом случае может помочь генератор ступенчатого напряжения?

Первая итерация синусоиды с помощью ступенчатой функции выглядит как периодическая последовательность разнополярных прямоугольных импульсов.

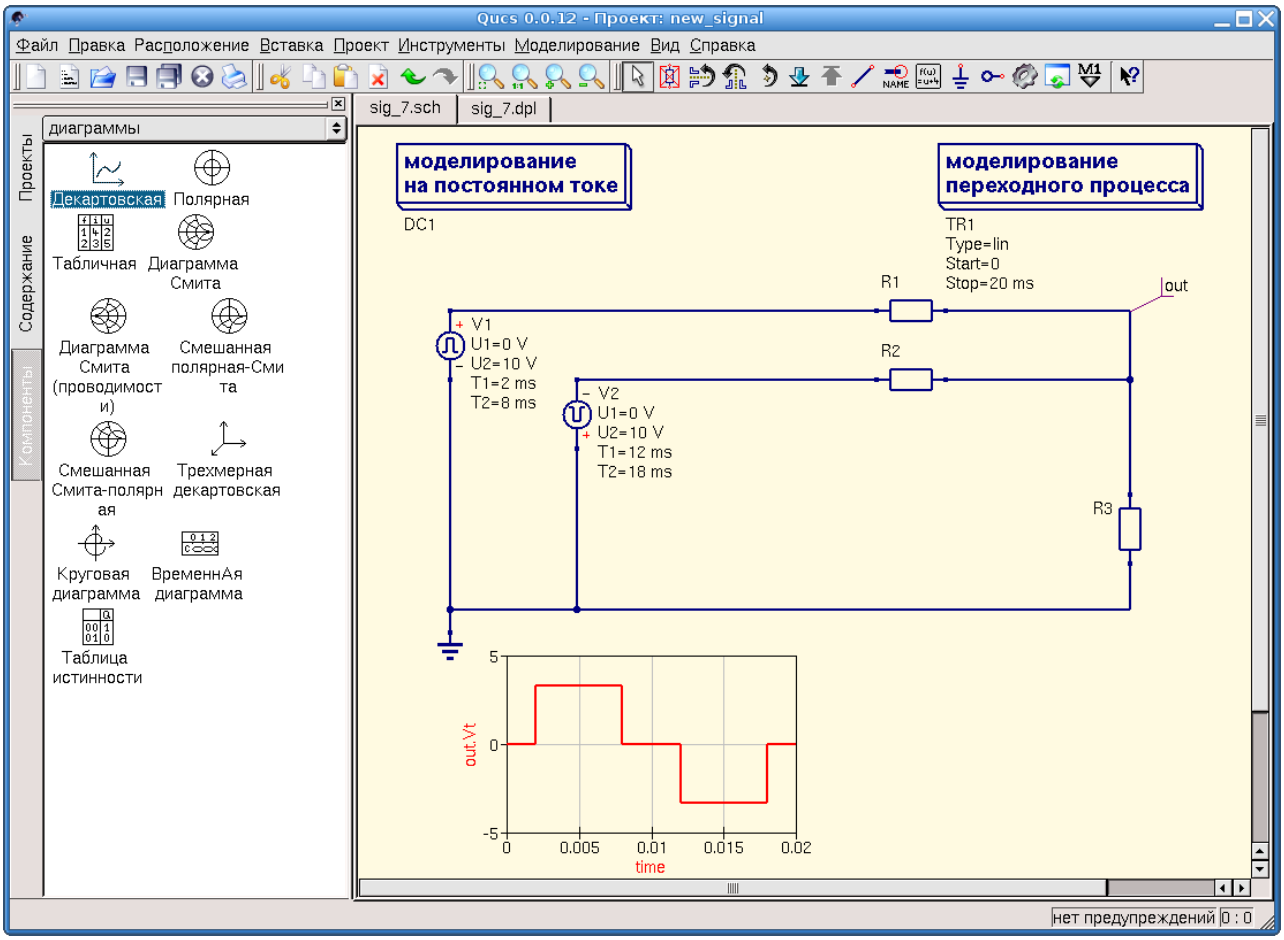


Рис. 10.9. Первая итерация силового напряжения

В таком решении оконечные транзисторы работают в импульсном режиме, и если импульсы имеют «хорошие» фронты, то мощность рассеиваемая выходными транзисторами существенно меньше, чем при работе с синусоидальным сигналом. Замена непрерывной работы транзисторов на импульсную и дает тот выигрыш, который позволит создать легкое и компактное устройство. Большая часть потребителей тока такого преобразователя имеет силовой трансформатор на входе блока питания, который в данном случае играет роль фильтра, преобразующего прямоугольные импульсы в синусоиду. Или подобный фильтр можно поставить на выходе конвертера, как это делается в бесперебойных устройствах питания большой мощности (UPS). При тех мощностях, на которые они рассчитаны, обычные преобразователи потребовали бы «перевозных» систем охлаждения.

Представьте себе ситуацию – вы купили для своего автомобиля преобразователь на ~220 В, чтобы включать дорогостоящий ноутбук без опасений, что вы сделали что-то не так. Вы человек осторожный, прежде, чем включить ноутбук, хотя продавец уверял, что именно так и следует поступить, вы решили посоветоваться с приятелем, большим любителем и знатоком.

Приятель мудро решил, что он принесет осциллограф, подключит к преобразователю лампочку, и все увидит (при вас!). Если ваш приятель не знаком с современными устройствами, он может убедить вас, увидев нечто похожее на диаграмму рисунка 10.9, в том, что вам подсунули бракованную вещь. Это лишь один пример, того, как важно иметь представление о том, что вы должны или можете увидеть на экране осциллографа. Прибор очень важный и полезный в практике и любителей, и профессионалов, но не способный заменить знания, хотя и способный помочь в приобретении оных.

Немного о сигналах и линиях

Буквально, чуть-чуть. За остальными сведениями лучше обратиться к великолепной книге Гоноровского И.С «Радиотехнические цепи и сигналы». Пусть вас не пугает математика, если вы еще не знакомы с ней в должном объеме — даже пропуская формулы, вы узнаете много нового для вас и полезного. В отношении математики я всегда вспоминаю, как мне в свое время казалось, что я лучше буду понимать теорию относительности, если пойму разницу между символами Кристофеля и тензорами. Освоив тензорный анализ, я убедился, что лучше стал понимать уравнения теории относительности, но не ее самое. Только постоянная и длительная практика могла бы оказать реальную помощь в этом, а без практики я счастливо забыл и тензорный анализ, и теорию относительности.

Радиолюбитель сталкивается с линиями часто. Любые два провода, идущие от одного устройства к другому, это, фактически, линия. Коаксиальный кабель, которым подключен ваш телевизор к антенне, это тоже линия. Любая из этих линий характеризуется рядом параметров, о существовании которых и влиянии на сигналы, необходимо знать.

Самый простой случай, когда знание вам нужно, это подключение с помощью двух проводов блока питания к вашему устройству. Любые провода, как вы знаете, имеют сопротивление, которое тем больше, чем тоньше провод. Любые провода, по которым протекает ток, создают падение напряжения на них равное (закон Ома)... И без учета этого падения напряжения нельзя сказать, какое напряжение должно быть на выходе блока питания, удаленного от устройства, напряжение питания которого должно быть строго определенным. Мало того, при протекании тока, и при наличии падения напряжения, на проводах будет рассеиваться мощность, выделяемая в виде тепла. Именно с учетом этого выделяемого тепла, с учетом допустимого падения напряжения рассчитываются все силовые провода, проложенные в вашей квартире, или в вашем доме. По причине того, что питающее напряжение очень низкочастотное, другие параметры силовой проводки в доме (кроме сопротивления изоляции) не слишком важны.

Вместе с тем, если вы возьмете радиокабель, то обратите внимание на его конструкцию. Я имею в виду самый обычный радиокабель. У него есть центральная жила и металлическая оплетка. Такой кабель, как любой провод, имеет сопротивление. Но его конструкция еще и явно указывает на то, что он будет иметь некоторую емкость! Его можно представить, такое представление еще называют линией с распределенными параметрами, в виде набора равномерно расположенных по всей длине сопротивлений и конденсаторов, примерно так:

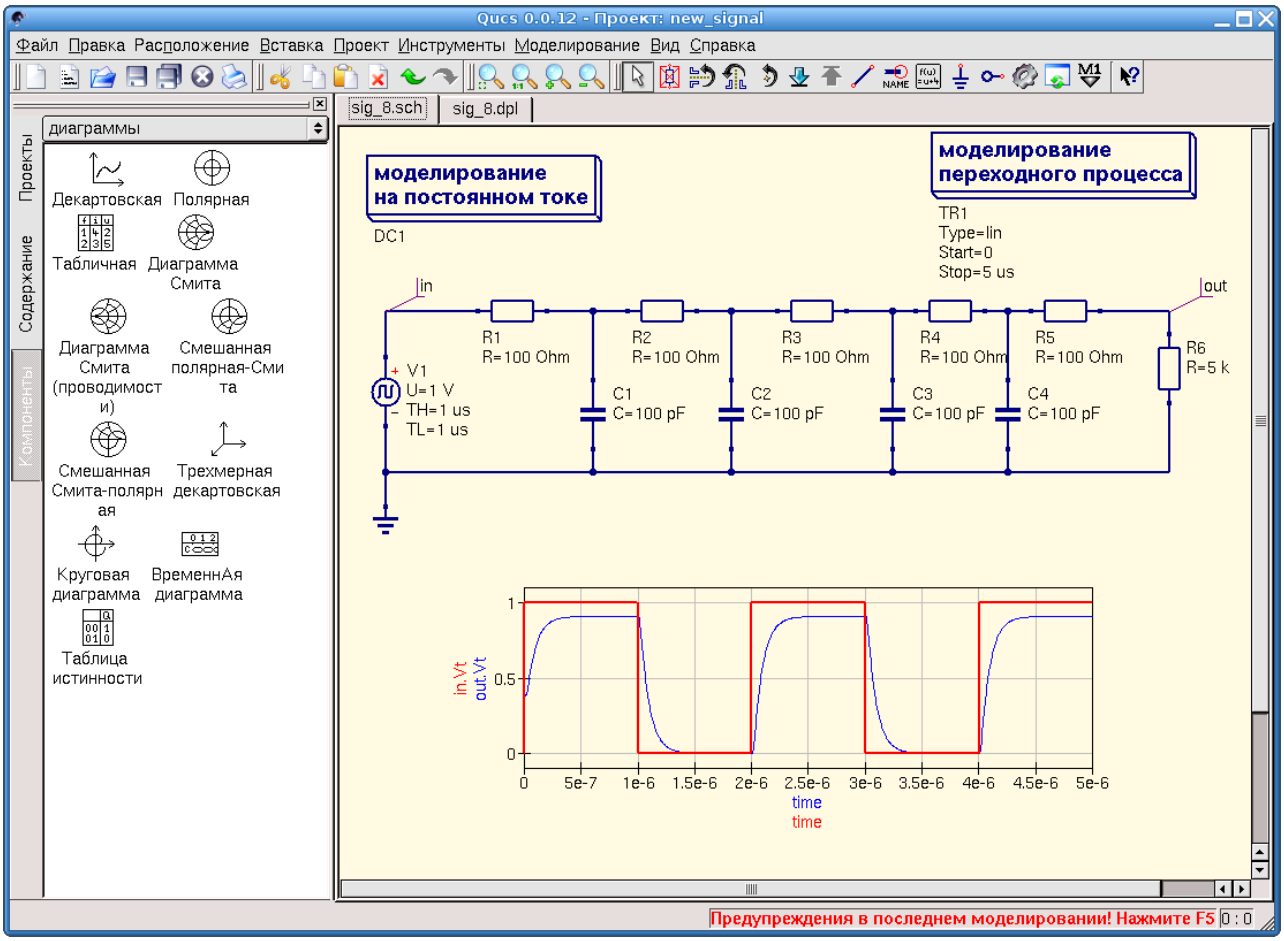


Рис. 10.10. Представление линии в виде RC цепей

Количество резисторов и конденсаторов, а так же их номиналы, я взял произвольно. Но ясно, что суммарно все эти сопротивления дадут общее сопротивление кабеля, а конденсаторы его общую емкость.

На диаграмме видно, во что превратился наш «чудненький» прямоугольный сигнал после того, как ему удалось «пролезть» сквозь эти дебри. А если добавить, что кроме очевидных сопротивления и емкости, радио-кабель может иметь «распределенную» подобным же образом индуктивность...

Искажения сигнала, как мы уже знаем, означает, что амплитудно-частотная характеристика кабеля имеет неравномерность и, видимо, ограничения в области верхней границы. Это достаточно очевидно. Менее очевидно, например, почему радиокабель маркируется как РК-50 или РК-75. Для сигналов линия, подобная кабелю, является «средой обитания», как для звука средой обитания будет помещение, в котором мы его воспроизводим. Когда я говорил об отражении звука от стен, я приводил пример того, что в результате интерференции отраженный звук может складываться и вычитаться из прямого. Проводя, может и не вполне корректную, но на мой взгляд наглядную аналогию, можно сказать, что сигнал, распространяясь в линии, подобно звуку может отражаться. Самым неприятным отражением будет для него отражение от противоположного конца линии. Как звук имеет длину волны, так и сигналы в линии имеют свою длину волны. И кабели принято характеризовать волновым сопротивлением. Именно волновое сопротивление кабеля отражено в его названии. Если сопротивление нагрузки линии, R6 на рисунке выше, равно волновому сопротивлению кабеля, то кабель ведет себя наилучшим образом по отношению к

сигналу в части отражения от другого конца. В этом случае через линию передается максимальная мощность сигнала.

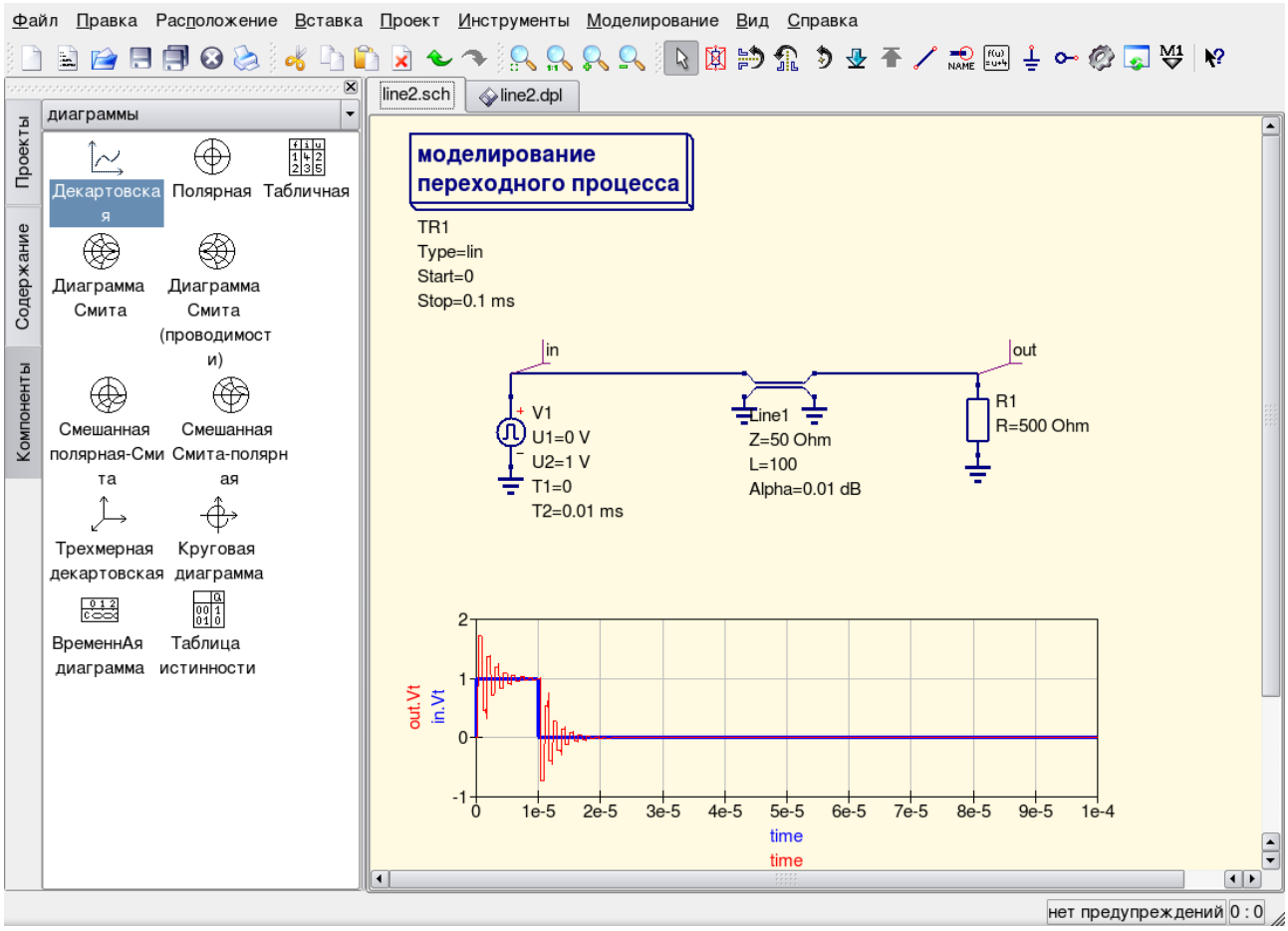


Рис. 10.11. Импульс в несогласованной линии

Таким образом, при использовании линии в своих разработках вам обязательно нужно иметь представление о том, как линия повлияет на переносимые через нее сигналы, будь то постоянный ток, радиосигнал или импульсная последовательность. Даже щуп вашего осциллографа, вместе с кабелем, соединяющим его с осциллографом, имеет некоторую емкость. Если вы касаетесь точки на схеме таким щупом, вы вносите эту емкость в схему, которую изучаете или исследуете. Вносимая вами емкость может в корне изменить работу схемы. Вы увидите на осциллографе то, чего нет в правильно работающей схеме. Не забывайте об этом!

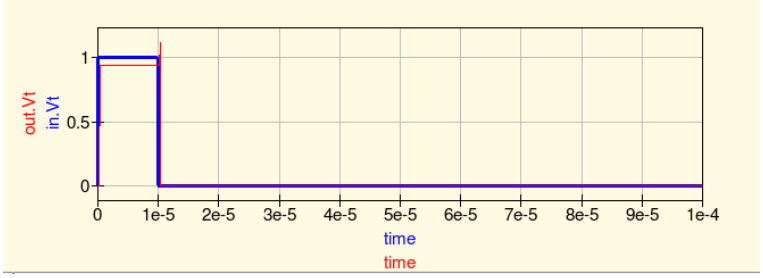


Рис. 10.12. Импульс в согласованной линии

Отражение сигнала от дальнего конца линии находит очень интересное применение, его

используют для исследования линии. Подавая в линию редкие и очень короткие импульсы, по времени прихода отражения можно судить о длине линии. Собственно прибор, используемый для этой цели, устроен так, что вам не нужно «судить-рядить», вы получаете точное значение.

Кроме того, отражение происходит не только от другого конца линии. Как вы помните из школьного курса физики свет отражается и преломляется на границах раздела сред. Или другими словами, в местах неоднородности среды распространения света. Но свет, как мы договорились, это тоже радиоволна, но очень большой частоты. Значит и любая волна может иметь те же свойства. Я имею в виду, что, отправляя короткий сигнал в линию для определения ее длины по отражению от другого конца, мы можем получить не одно отражение, а ряд отражений во всех местах, где линия перестает быть однородной. Это и места соединений линии (пайка, скрутка и т.д.), и места повреждения линий: обрыв, короткое замыкание. Как радиолокатор (еще одно применение отражения, теперь уже радиоволны) ловит отражение от летящего самолета, так и прибор для исследования мест повреждения кабеля ловит отражение импульса и по времени его прихода вычисляет расстояние до повреждения, а по виду отраженного импульса дает нам представление о характере повреждения.

Вот так знание свойств линий, знание того, что мы увидим на экране осциллографа, может помочь отыскать повреждение в кабеле, закопанном глубоко в землю. И исправить его!

Больше об амплитудной модуляции

Когда речь заходит об амплитудной или частотной модуляции, то предмет разговора невольно связывается с радио. Некогда в радиовещании применялась только амплитудная модуляция. Позже ей на смену пришла частотная модуляция. Но в диапазонах длинных, средних и коротких волн амплитудная модуляция радиосигнала не сдала своих позиций.

Пример того, как выглядит амплитудно-модулированный радиосигнал приводился в начале книги. Напомню, что в таком радиосигнале есть несущая частота – переносчик и разносчик информативной составляющей радиосигнала, последнюю еще называют «огибающей». Аналитическая форма записи для несущей частоты выглядит примерно так:

$$f_{радио} = A*sin(\acute{\omega}_p t)$$

Здесь $f_{радио}$ – это функция, изменяющаяся по закону синуса с частотой $\acute{\omega}_p$. Но функция синуса, о чем мы прекрасно осведомлены, изменяется от -1 до +1, тогда как сигналы могут иметь разные значения амплитуды. Эта неприятность устраняется с помощью постоянной, которая и будет амплитудой результирующего сигнала – A.

Построить такую радиостанцию не представляет большого труда: генератор на частоту $\acute{\omega}_p = 2\pi f_p$ и усилитель мощности, но для ее использования потребуется получить разрешение. Однако главная проблема не в этом. Что услышит некто, настроившись на волну нашей радиостанции, скажем «Полная Свобода!»? Правильно, полную свободу фантазировать о содержании. Это как в восточной мудрости – увидеть белого дракона на белом полотне.

Чтобы добавить хоть какое-то содержание к излучаемой нами свободе, мы можем с точки зрения математики записать нашу «радио» функцию в таком виде:

$$F_{радио} = A(t)*sin(\acute{\omega}_p t)$$

Я заменил амплитуду A на новую функцию времени, которая и должна нести содержание. Например, это будет сигнал с частотой 400 Гц, который можно часто слышать по телевизору при передаче технической таблицы для настройки телевизора. Это не Скрябин, да, но хоть

какая-то польза.

Сигнал с частотой «содержания» воздействует на нашу несущую частоту, заставляя ее амплитуду меняться (он сам стал амплитудой!). Как может выглядеть такой процесс? Попробуем. Но начнем с простого. С ответа на вопрос, как мы поступим с информационным низкочастотным сигналом.

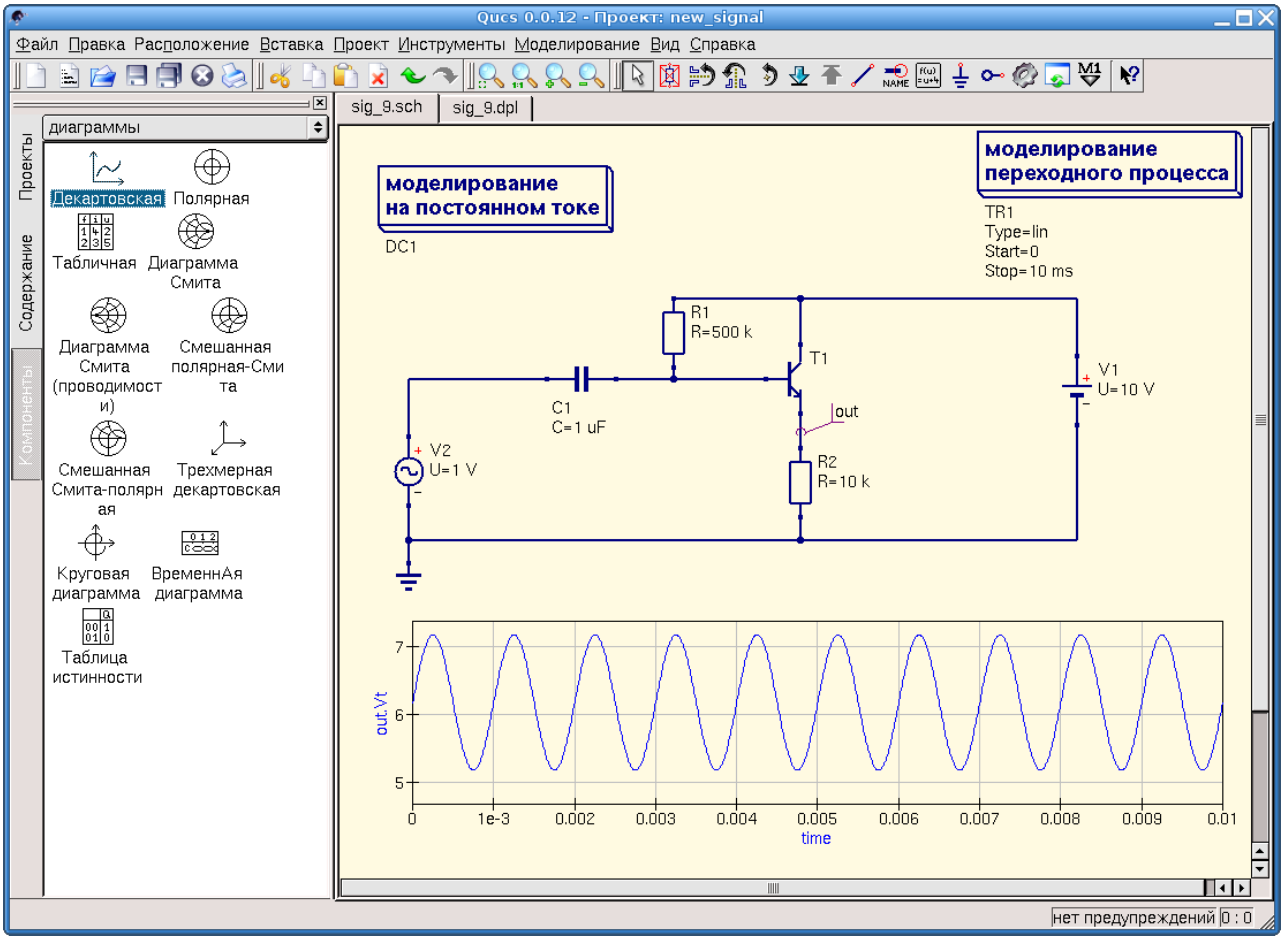


Рис. 10.13. Низкочастотный усилитель радиосигнала

Представленная на рисунке диаграмма должна определять амплитуду несущей частоты в любой момент времени, но так, чтобы последняя не была отрицательна. Теперь мне приходит в голову идея использовать наш низкочастотный усилитель в качестве источника питания для усилителя высокой, несущей, частоты. Ведь сигнал усилителя может оказаться чувствителен к изменениям питающего напряжения. Недаром принимают меры к стабилизации этого напряжения. А мы поступим наоборот!? Возможно, это авантюра, но намерения у меня добрые.

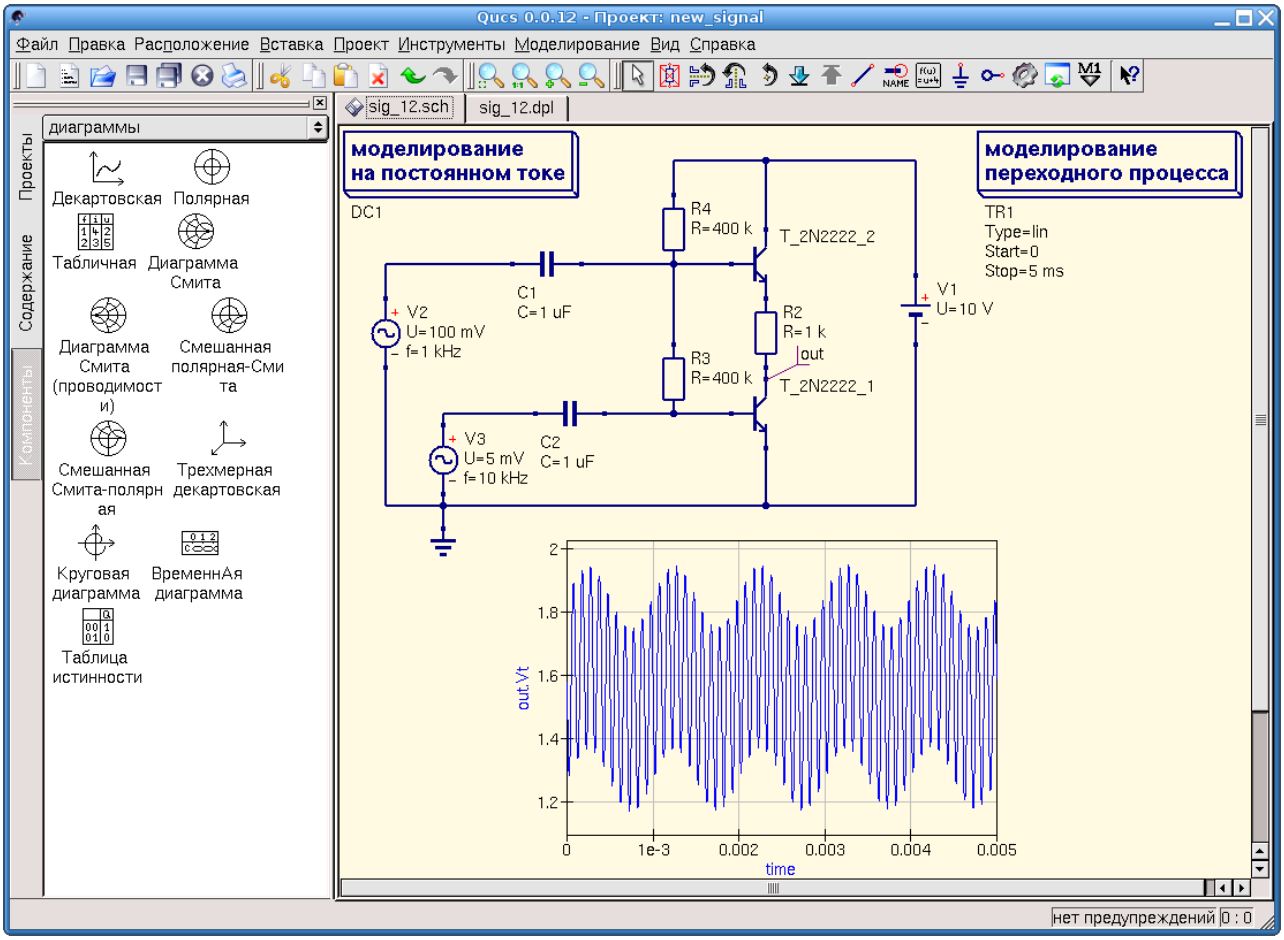


Рис. 10.14. Попытка создания амплитудно-модулированного сигнала

Увы, хотя есть два сигнала, но результат не похож на модуляцию.

Вместо формирования амплитуды несущей частоты низкочастотный сигнал перемещает рабочую точку со своей частотой, не меняя величины амплитуды несущей. Идея была хороша, но попытка неудачна. А попробовать стоило.

Впрочем, и последующие попытки, как я ни старался, не дали лучших результатов, хотя диаграмма при одной из них была совсем похожа на настоящий успех, но схема, с которой она была получена, явно получилась неработающей.

Для достижения эффекта амплитудной модуляции информационный сигнал должен так воздействовать на усилитель несущей частоты, чтобы изменять коэффициент усиления пропорционально своему закону изменения $A(t)$ (в формуле для радиочастоты).

Я иногда жульничаю, сделаю это и сейчас. Я запишу уравнение, которое и использую для получения диаграммы. Идея уравнения в том, что синусоидальный низкочастотный сигнал меняется от +1 до -1, а амплитуда результата не должна быть отрицательной, следовательно, нежно добавить единицу. В остальном все будет соответствовать уравнению для радиосигнала.

Вот и результат, который можно показать, чтобы продолжить разговор об амплитудной модуляции. Попробуйте в уравнении ниже изменить добавленную единицу на 2, например, и вы получите другую *глубину модуляции*.

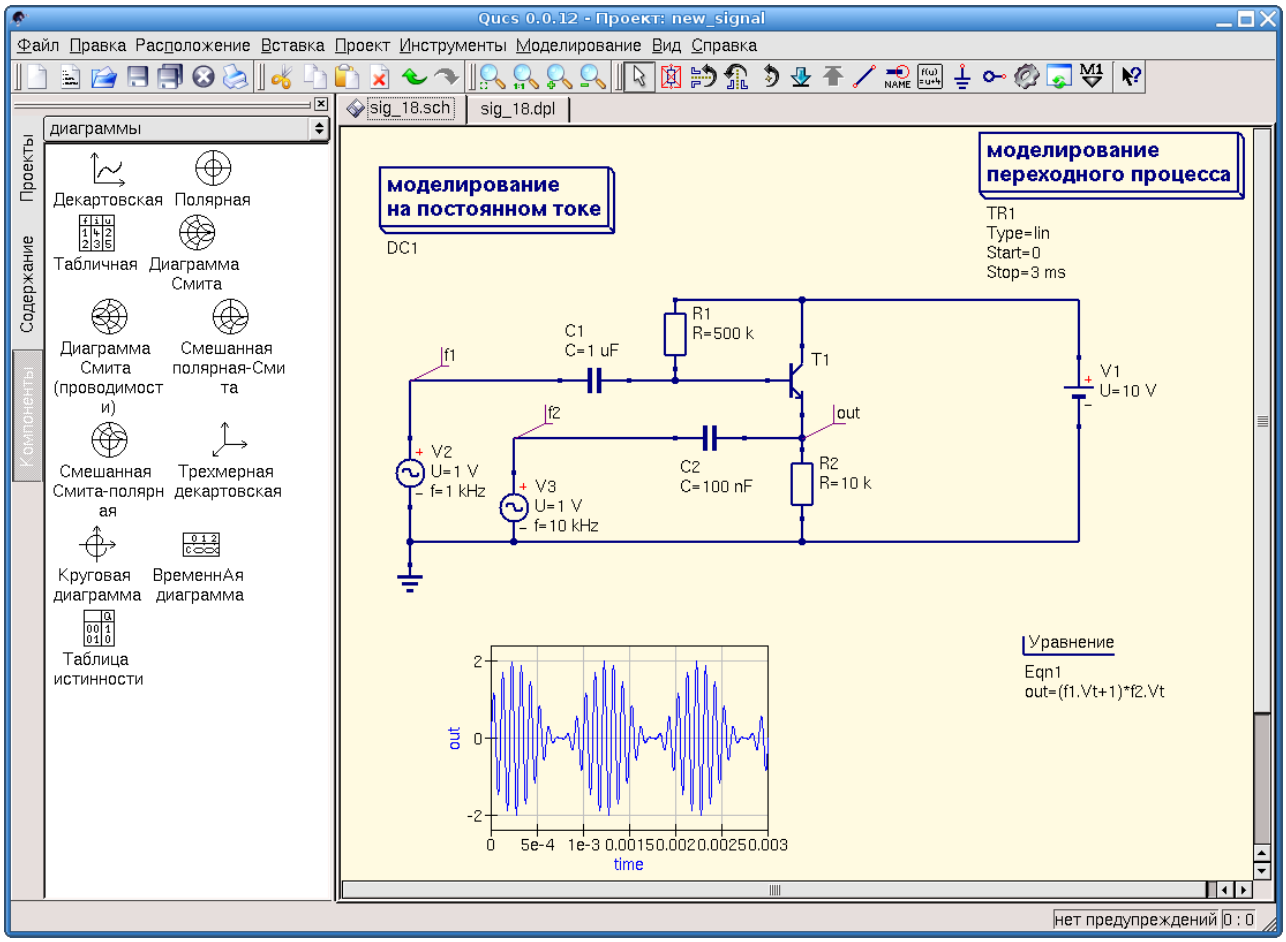


Рис. 10.15. Получение амплитудно-модулированного сигнала

На рисунке достаточно отчетливо можно увидеть и «несущую», и «огибающую», которые сформированы двумя источниками переменного напряжения V2 и V3. Что и требовалось в данный момент продемонстрировать, чтобы с легким сердцем перейти к тому, на чем я, собственно, хотел остановиться.

Амплитудная модуляция все реже применяется в радиовещании, ее теснит частотная модуляция – воздействие информационным сигналом не на амплитуду несущей, а на ее частоту. Есть и другие виды модуляции, например, фазовая модуляция: если полностью записать функцию синусоидального сигнала, то в синус войдет кроме частоты еще и фаза, о которой я время от времени упоминаю.

А еще я хочу сказать, что самый простой способ амплитудной модуляции в настоящее время используется совсем не в радиотехнике, а в дистанционном управлении бытовыми приборами: телевизорами, CD- и DVD-проигрывателями и т.д. Речь идет о пультах управления. Каким образом организована работа обычного пульта управления телевизором? Мы знаем, что для управления используется инфракрасное излучение. Самая простая схема может выглядеть так – посылаем ИК луч, который улавливает фотоприемник, и выполняем включение прибора. Повторяем эту команду и выключаем прибор.

Подобное решение можно реализовать, но у него два существенных недостатка. Для создания ИК луча достаточной силы, чтобы преодолеть несколько метров и быть еще способным воздействовать на фотоприемник, необходимо использовать оптику, фокусирующие линзы. И второе, ИК излучение исходит не только от пультов управления, но от любого нагретого предмета, в частности, и от человека, температура которого выше

температуры воздуха в комнате. Телевизор включался бы и выключался беспрерывно под воздействием этих излучений. Поэтому в пультах дистанционного управления давно уже использую амплитудную модуляцию, где несущая – это последовательность импульсов с частотой в несколько килогерц (на практике это значение может меняться от 20 до 500 кГц), а огибающая сформирована информационной последовательностью импульсов. Получить амплитудную модуляцию при таких «вводных», мне гораздо легче.

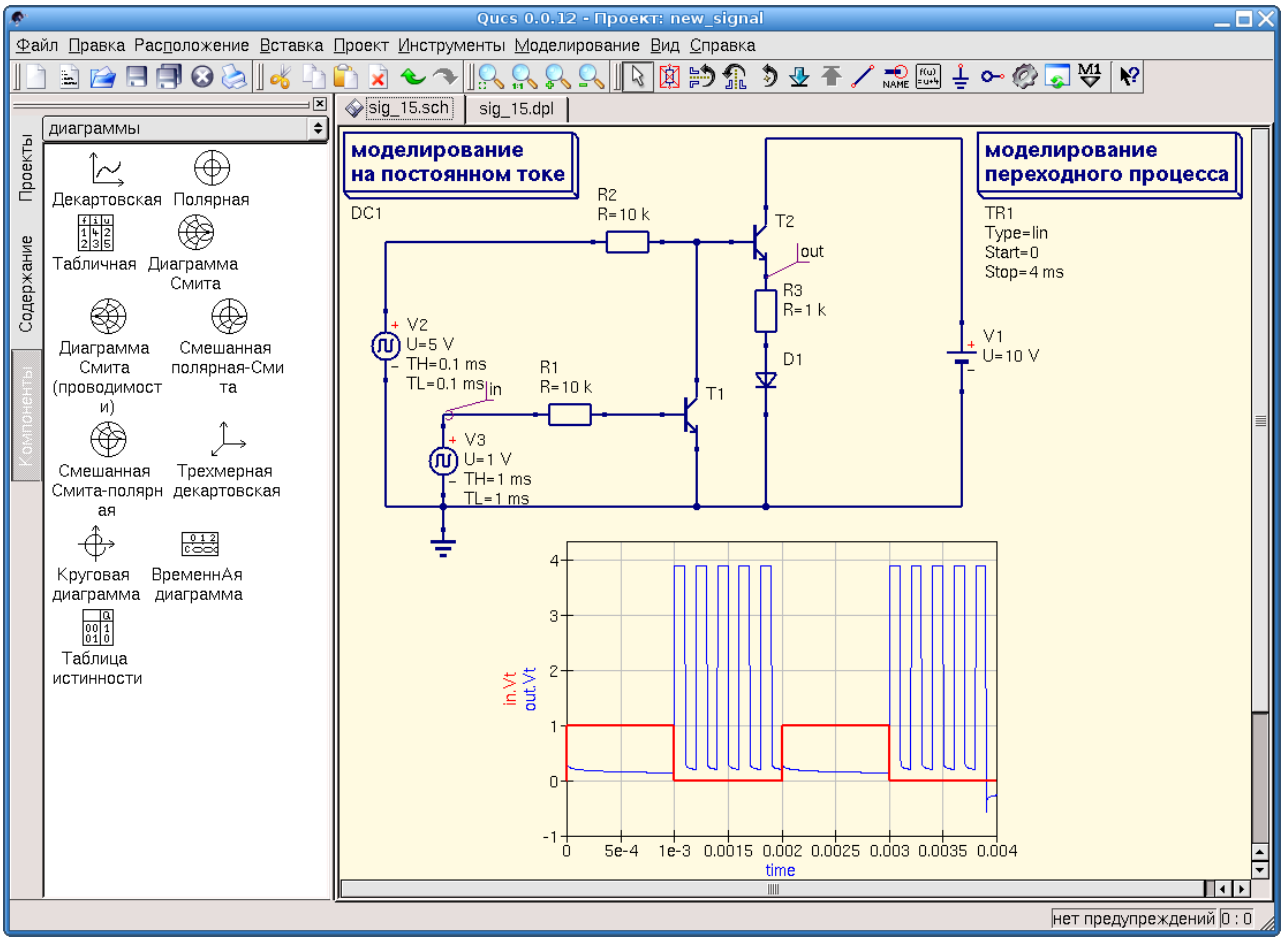


Рис. 10.16. Существо амплитудной модуляции в ИК пультах управления

Это только только одно из возможных решений, где информацию несут импульсы (*in*) от источника V3, а несущая формируется генератором V2. На диаграмме, в отличие от реальных, все информационные импульсы одинаковы. В действительности они сформированы из весьма, порой, сложной последовательности, определяемой либо только изготовителем, либо изготовителем в соответствии с рекомендациями некоторых стандартов. Но даже такой простой сигнал, как на диаграмме выше, может успешно включить и выключить устройство. Фотоприемник управляемого устройства следует включить так, чтобы медленно изменяющееся ИК излучения отсеивались, а распознавались только импульсы ИК излучения с заданной частотой, на диаграмме около 10 кГц. Кроме задачи лучшего распознавания сигнала, такое построение амплитудной модуляции часто позволяет решить и вторую задачу – применение достаточно маломощных ИК излучателей (светодиодов ИК диапазона) без оптики. Для этого в качестве импульсов несущей частоты используют короткие импульсы с большой скважностью похожие на такие:

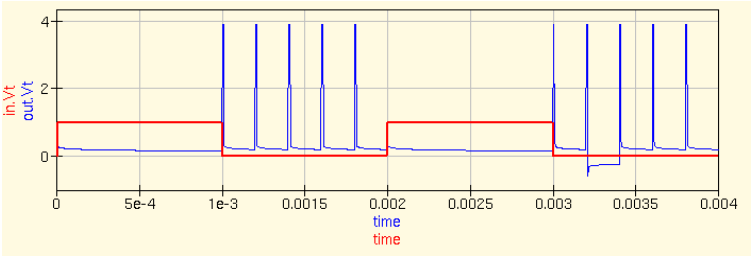


Рис. 10.17. Вид импульсов несущей частоты для маломощных излучателей

Время, когда через светодиод протекает ток, уменьшают, а время, когда светодиод выключен, пропорционально увеличивают. Дело в том, что все светодиоды позволяют существенно увеличить рабочий ток при условии, что это короткий импульс, за которым следует долгая пауза. В итоге вспышки «света» становятся гораздо ярче, что и фиксируется фотоприемником, установленным в управляемом приборе.

И еще одно. Если вместо ИК управления использовать радиоуправление на разрешенных для этого частотах, то можно, исключая несущую частоту, которая должна быть синусоидальной, использовать последовательность информационных импульсов такую же, как и в пульте ИК управления.

Вообще, принцип модуляции используется гораздо чаще, чем это можно предположить. Не только традиционные области радиовещания, но и средства коммуникации, измерения и автоматика активно используют его. Компьютерные телефонные модемы, еще не полностью вытесненные другими видами подключения к глобальным компьютерным сетям, в своем названии, образованном от слов «модуляция-демодуляция», увековечивают использование этого принципа. Очень часто для наблюдения за медленно меняющимися процессами, чтобы не быть голословным, приведу пример температуры, для наблюдения за ней часто используется датчик в виде термочувствительного сопротивления. Напряжение на этом сопротивлении меняется медленно и может и само быть небольшим, а, главное, изменяться незначительно. На практике можно использовать усилитель постоянного тока для наблюдения за этим напряжением. Но усилители постоянного тока оказываются очень чувствительны к колебаниям окружающей среды, вносящей изменения в выходные параметры, да сами такие усилители склонны к произвольным, хотя и незначительным, изменениям своих параметров. Чтобы избежать влияния этих эффектов на результат работы устройства применяют модуляцию постоянного тока (или напряжения), превращающую сигнал постоянного тока в переменный, который можно усиливать, используя усилители переменного тока (или напряжения), параметры которых гораздо легче стабилизировать. А то, что информационная функция перестала быть, например, звуком, став почти постоянным напряжением, то с точки зрения общего подхода мы и не оговаривали вид модулирующей функции.

Другие преобразования напряжений

Закончив предыдущее рассмотрение преобразованием постоянного (или почти постоянного) напряжения в переменное, мы коснулись еще одного очень интересного вопроса, связанного с сигналами. Тесно соприкасаясь с компьютерной техникой, с новыми коммуникационными и информационными технологиями, эта тема может быть названа «Оцифровывание». С развитием цифровых компьютеров появилась возможность и необходимость в оцифровывании таких ранее абсолютно аналоговых (непрерывных) сущностей, как звук и изображение.

Рассмотрим это на примере звука. Сегодня есть цифровая запись. Есть цифровые музыкальные диски. Есть цифровая передача звука по проводам и радиоканалу. Есть обработка цифрового звука на компьютере. Что же представляет собой оцифровка звука?

Конечный результат этого процесса мы как-то можем представить – некоторый набор чисел, сопоставленный звуку. Понятно, что компьютер, предназначенный для обработки чисел, может эти числа обрабатывать (складывать, вычитать и т.д.), но какое это имеет отношение к звуку как таковому?

Мы уже готовы к тому, чтобы под звуком понимать преобразованное в напряжение, скажем с помощью микрофона, звуковое давление, меняющееся по некоторому, может быть, очень сложному закону. Не будем слишком усложнять себе жизнь и примем, что нам безразличен вид функции этого напряжения. Мы возьмем простой – синусоиду, тем паче, что знаем, любой сложный сигнал мы можем представить как совокупность синусоидальных сигналов. Итак, мы незаметно для себя осуществили первое преобразование, превратив звук (тихий свист) в переменное напряжение. Обратимся к новому преобразованию.

Основа этого преобразования состоит в том, что при передаче сигнала вовсе не обязательно передавать весь сигнал, достаточно передавать его отдельные (дискретные) значения, производимые периодически. То есть, сигнал превращается в ряд значений амплитуды сигнала, по которым мы в любой момент сможем восстановить вид сигнала. То, сколько значений требуется для полного восстановления сигнала, определяется математическими методами. Не буду, как обычно, задерживать вас с математикой, при всем моем к ней уважении и любви, сошлюсь только на «Теорему отсчетов», которая в конечном счете гласит, что частота, с которой следует измерять амплитуду сигнала, должна вдвое превышать максимальную частоту сигнала. Иными словами, если мы хотим получить качественное воспроизведение музыки в полосе звуковых частот 40 Гц — 15 кГц, то мы должны с частотой 30 кГц измерять амплитуду звука при исполнении музыкального произведения. Точнее измерять амплитуду сложного переменного напряжения, в которое превращается звук.

Для нашего испытания синусоиды, теперь забудем, что это звук, а рассмотрим процесс, используя синусоиду в качестве примера, для преобразования синусоиды разобьем ось времени (ось «x») на равные интервалы, через которые и будем определять величину амплитуды.

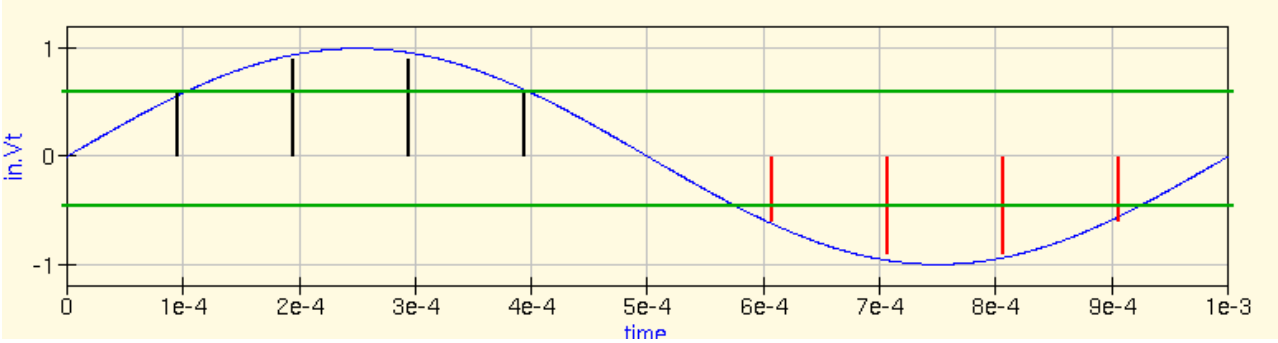


Рис. 10.18. Принцип дискретизации звука

Как смог, я нарисовал через равные промежутки времени 100 мкС амплитуды. Если горизонтальные лини, которые я пририсовал к диаграмме, мы примем за возможности нашего измерителя амплитуд (то, что линия разделяет вертикальную ось не в середине промежутка между 0 и 1, заслуга только моя, считайте, что это середина), то мы можем выразить эти амплитуды через числа. Первая амплитуда равна нулю – начальная точка

отсчета. Вторая амплитуда равна одной единице (половина единицы напряжения). Третья двум и т.д. Для амплитуд второй полу-волны значения будут отрицательными. Мы можем даже выразить последовательность этих числе в двоичной форме: 0, 01, 10, 10, 01 и т.д.

Вот такую работу осуществляет электронная микросхема, которая называется аналого-цифровой преобразователь или АЦП. Мы подаем на преобразователь тактовые импульсы, периодичность которых определяется теоремой отсчетов, преобразователь производит отсчеты в заданные моменты времени, а в перерывах между этими отсчетами может передать нам полученные значения в двоичном, например, виде. Многие современные микросхемы АЦП имеют в своем составе и некоторое количество памяти, где могут запоминать результаты, и средства передачи результатов измерений на значительные расстояния. Разрядность АЦП (количество выходных бит) определяет точность, с которой могут быть измерены амплитуды.

Имея в своем распоряжении значения амплитуд, зная периодичность отсчетов, мы можем восстановить вид сигнала. Соедините вертикальные линии на предыдущем рисунке хотя бы отрезками прямых линий и вы получите вполне правдоподобный вид синусоиды. Конечно, если вы попробуете восстановить этот вид по тем данным, которые я смог воспроизвести, напомню: 0, 01, 10 и т.д., – вы получите очень приблизительный вид функции.

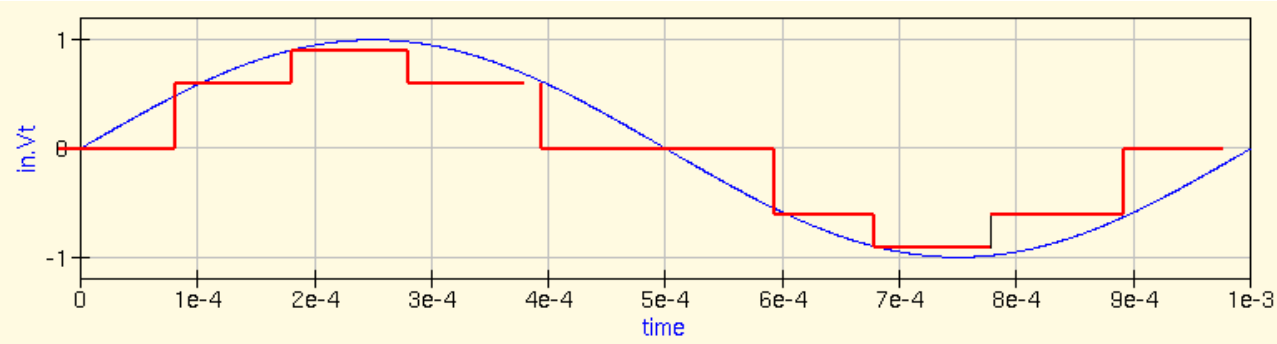


Рис. 10.19. Вид восстановленной функции синуса

Думаю, вы сообразили, что этот «прекрасный» рисунок, плод моих стараний, ни что иное, как ступенчатая функция, о которой мы уже говорили. Работу, которую я только что проделал, с гораздо большим успехом выполняет другой преобразователь, который называется цифро-аналоговый преобразователь или ЦАП. Получая числа от АЦП он строит результат, соединяя прямыми полученные значения амплитуд.

Если сетку диаграммы выше сделать гораздо мельче, что потребует большего количества измерений, то результат получится гораздо лучше. Чем выше разрешающая способность подобного построения, тем труднее обнаружить «подлог», труднее решить, с чем мы имеем дело — с реальным непрерывным процессом, или его числовым представлением.

Ярким примером, насколько это «обстоит именно так» может служить ваш монитор, если вы работаете за компьютером. Его разрешающая способность определяется величиной точки на мониторе, которую он может передать. Все эти точки с их параметрами запоминаются в видео-памяти. Это, в сущности, цифровая память ничем не отличающаяся от оперативной памяти компьютера, и сегодня объем видео-памяти такой, что превосходит оперативную память компьютеров двадцатилетней давности, которые и тогда проделывали очень много полезного, тогда как вы можете даже не заметить всей прелести воспроизводимых на мониторе изображений.

Запоминающий цифровой осциллограф, пусть это будет приставка к компьютеру,

содержит в своем составе аналого-цифровой преобразователь. Последний измеряет тестируемый сигнал, передает полученные значения в компьютер, а программа, обслуживающая осциллограф, воссоздает вид сигнала подобно цифро-аналоговому преобразователю. Частота, с которой может работать встроенная в такой осциллограф микросхема АЦП, определит верхнюю граничную частоту наблюдаемых сигналов.

Кроме осциллографа метод «оцифровывания» давно используется для записи изображения, скажем в фотоаппарате. Используется в телефонии и вещании, и мало-помалу вытесняет аналоговые способы хранения и передачи информации.

К работе преобразователей, я думаю мы вернемся позже, может быть, после рассмотрения работы цифровых процессов и микроконтроллеров, а сейчас, немного расширив свои знания о сигналах, вернемся ненадолго к усилителям.

Глава 11. Немного больше об усилителях

Принимая решение о создании своего устройства, вам не следует забывать, что конечной целью для вас могут быть только две вещи: знание и удовольствие. В первом случае вы можете сами придумывать устройство или искать готовую схему для того, чтобы исследовать работу устройства, понять эту работу, и в процессе экспериментов пополнить свои знания предметной области. Во втором случае вы делаете тоже самое, но ваша цель получить удовольствие от результата работы: сделать хорошее устройство, которого ни у кого нет, или сделать устройство, потратив только немного денег, а другие должны были заплатить за него полную цену. Как мне кажется, самое лучшее это использовать обе цели, вначале первую, а затем вторую. Вы наверняка получите вдвое больше от своего занятия!

Почему я заговорил о цели, так только потому, что она часто определяет выбор решения. Если говорить об усилителях вообще, не соотнося это с конкретной целью, то невозможно сделать выбор из множества возможных схем. Только конкретная цель, конкретная задача позволяют сделать это.

Усилители можно было бы сразу разделить на две большие категории, усилители постоянного и переменного тока. И когда-то это было определяющее деление. Но с появлением транзисторов схемы усилителей все чаще стали делать с непосредственными связями (без разделительных конденсаторов), что позволяет применять их и для усиления постоянного и для усиления переменного тока. По таким схемам построены все операционные усилители.

Усилители можно было бы сразу разделить на высокочастотные и низкочастотные. Но изменение технологии производства транзисторов привело к их существенному удешевлению, и сегодня одни и те же транзисторы можно применить для построения высокочастотной части радиоприемника, равно как и низкочастотной.

Кстати о радиоприемниках, рассмотрим схему приемника прямого усиления.

Усилители в радиоприемнике

Подобная схема радиоприемника обозначалась как 1-V-1, что означало один каскад усиления высокой частоты, детектор и один каскад усиления низкой частоты. Я не застал те времена, когда происходило становление таких схем, но полагаю, что широкое распространение детекторных приемников привело радиолюбителей к желанию улучшить схему детекторного приемника сначала за счет усиления низкочастотного сигнала после детектора, а затем и к желанию усилить радиосигнал.

Приведенная ниже схема условна и не предназначена к повторению. Ее можно было бы изобразить в виде структурной, но мне хотелось показать вид сигналов, которые можно увидеть в разных точках схемы.

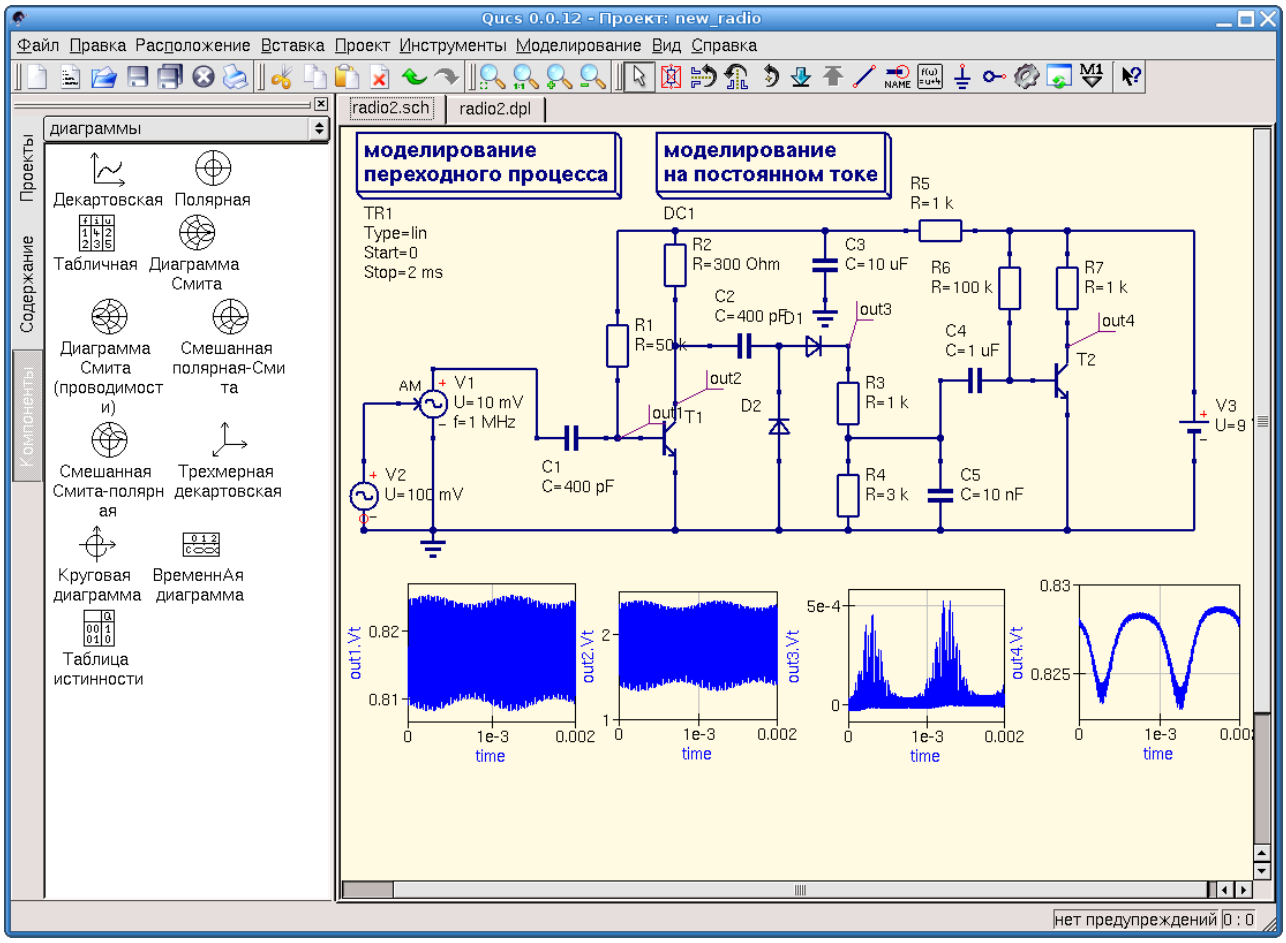


Рис. 11.1. Почти настоящая схема радиоприемника

Сразу хочу сказать, что гораздо нагляднее, гораздо лучше вы можете получить результаты, если исследуете подобную схему в программе Qucs покаскадно. Отдельно входной контур, отдельно усилитель ВЧ и т.д.

Вместо антенны и входного контура радиоприемника использован генератор амплитудно-модулированного сигнала. Такой сигнал приносит радиоволна, излучаемая радиостанцией. Величина сигнала 10 мВ скорее завышена, чем занижена. На входе радиоприемника сигнал попадает на входной LC-контур, который выделяет ту частоту, которая соответствует его резонансной частоте. Обычно конденсатор этого контура делают переменным, но можно изменять и индуктивность. Если вас интересуют радиоприемники, то можно найти много схем подобных приемников и выбрать любую из них. Не думаю, что сегодня есть смысл ставить целью получение готового устройства, но в качестве испытательного стенда, предмета для проведения ряда экспериментов, любая из схем принесет много пользы.

Итак, амплитудно-модулированный сигнал, каким он изображен на левой диаграмме, приходит на вход усилителя высокой частоты, выделенный из множества других входным LC-контуром. Величина сигнала маленькая, поэтому расчет каскада усиления высокой частоты зачастую производится с использованием методов для малосигнальных параметров. Напряжение питания этого усилителя тоже выбирается, как правило, небольшим, а транзистор используется с высокой граничной частотой или из серии высокочастотных. Я думаю, вы встречали схемы, где используются транзисторы КТ315. Можете попробовать использовать их и в усилителе высокой частоты, и в усилителе низкой частоты.

И вы, и я знаем, что есть самый простой вид приемника – детекторный. Он не содержит

усилителя, поэтому возникает вопрос, а зачем нужен усилитель?

Детекторный приемник, принимая сигнал радиостанции, сможет работать только с самыми мощными, близко от вас расположенными радиостанциями, способными обеспечить обработку сигнала. Когда мы говорили об n-p переходах, о диодах, мы пришли к выводу, что для нормальной работы диоды ему нужно падение напряжения на переходе от 0.1 до 0.5 В. Поскольку детектор в данном случае не более, чем выпрямитель, сигнал, который он может выпрямить, не должен быть меньше этих значений. Таким образом, усилитель высокой частоты в радиоприемнике нам нужен в первую очередь для того, чтобы обеспечить нормальную работу диода.

В реальных приемниках прямого усиления вы можете не увидеть детектора (диода). Куда он пропал? Он не пропал, в качестве диода используется переход база-эмиттер транзистора, который, как n-p переход, прекрасно справляется с задачей выпрямления высокочастотного сигнала. А транзистор используется и как детектор, и как усилитель, благо на его переходе создается небольшое падение напряжения, которое помогает процессу выпрямления (детектирования). Иногда каскад усиления ВЧ (высокой частоты) используется для создания положительной обратной связи, охватывающей и входной LC-контур. Зачем вводится положительная обратная связь, которая может привести к «самовозбуждению» усилителя, и которая ухудшает его параметры. Ведь мы говорили, что отрицательная обратная связь улучшает многие параметры усилителя, а положительная, от обратного, должна их ухудшать. Это так. Но ухудшая одни параметры, она может улучшить другие. Например, усиление по напряжению каскада с положительной обратной связью будет больше. То есть, наш усилитель будет усиливать лучше, не надо будет добавлять второй каскад. Кроме того, положительная обратная связь в фильтре повышает добротность фильтра, а наш входной LC-контур, как мы и говорили, это фильтр. После введения положительной обратной связи он становится круче в буквальном смысле слова, спад вне частоты резонанса происходит значительно быстрее. Приемники, использующие такое построение, называют сверхрегенеративными. Они сложнее в настройке, действительно склонны к самовозбуждению, но могут улучшить работу приемника прямого усиления.

Вернемся к рисунку. На следующей диаграмме показан усиленный сигнал на выходе каскада ВЧ, а следом выпрямленный сигнал. Он несет в качестве огибающей, как и входной сигнал, нужную нам информацию, а высокочастотное наполнение (несущая), которое было так нужно радиоволне, нам уже ни к чему. Поэтому в схему добавлен конденсатор C5 (на рисунке), сопротивление которого для сигнала звуковой частоты очень большое, а сопротивление для частоты несущей очень маленькое. Благодаря частотно-зависимому делителю напряжения R3C5 мы сохраняем амплитуду информационной части и уменьшаем амплитуду несущей. Или иначе, мы говорили о выпрямлении высокочастотного сигнала, в этом случае, как в любом выпрямителе, конденсатор C5 можно рассматривать как сглаживающий для высокой частоты, но он почти не сглаживает, из-за малой емкости, низкочастотный сигнал. Последняя диаграмма на рисунке показывает усиленный информационный сигнал с остатками высокой частоты.

Низкочастотный усилитель, располагающийся за детектором, в данном случае призван в первую очередь «умощнить» сигнал. Слабый продетектированный сигнал плохо будет слышен даже в наушниках. Его мощность следует увеличить. Это и выполняет усилитель низкой частоты. Он работает с большим сигналом, его расчет ведется соответствующими методами, а вы можете применить любые схемы усилителей НЧ. Часто эти схемы имеют чувствительность порядка 250 мВ, что соответствует стандартной для них чувствительности. Этого может не хватить для работы с сигналом, снимаемым с детектора. Тогда вам потребуется предварительный усилитель.

Предварительный усилитель НЧ

Раньше в зоне практического внимания находилось больше разновидностей предварительных усилителей НЧ: предварительный усилитель для проигрывателя грампластинок с электромагнитной головкой звукоснимателя, усилитель магнитофона, обязательные регуляторы тембра. Все они выполняли требуемое формирование амплитудно-частотной характеристики.

В настоящее время остаются, быть может, если говорить о звуковоспроизведении, эквалайзеры – многополосные фильтры, с помощью которых можно в довольно широких пределах менять результирующую АЧХ усилителя, хотя и их все чаще заменяют формирователи режима прослушивания.

Собственно, схема предварительного усилителя, если это не активный фильтр, ничем не отличается от той, что на рисунке. Иногда, правда, следует принять меры к тому, чтобы минимизировать шумы предварительного усилителя. Если сигнал на входе предварительного усилителя мал, а вы хотите сохранить большой динамический диапазон всего тракта, то без этого не обойтись. В отношении шумов я хочу в первую очередь обратить внимание на два фактора: для малошумящих транзисторов (есть такой параметр) шумовые свойства нормируются и уменьшение коллекторного тока уменьшает шумы. Немного больше об этом.

Малошумящие транзисторы не панацея от бед с шумами. Дело в том, что выбирая именно малошумящий транзистор, вы получаете только заведомо известную норму его шумовых свойств, за которую он не выходит. Но это не означает, что остальные транзисторы шумят сильнее. Они могут иметь и меньшие шумы, но те, просто, не определены для всей серии, то есть, могут встретиться и очень шумные экземпляры. И еще. В некоторых готовых устройствах указывают соотношение сигнал/шум, но не приводят ссылок на метод измерения шума. Очень часто шумы измеряют с использованием «взвешивающего» фильтра. Что он из себя представляет? Он имеет амплитудно-частотную характеристику, отражающую реальную «слышимость» шумов, реальное восприятие нами шумов, как мешающего фактора. Выбрав схему малошумящего усилителя вы можете долго пытаться настроить его, чтобы получить данное в описании значение шумов, но не достичь этого. Не стоит сразу отказываться от схемы, возможно, ее испытания проводились именно со взвешивающим фильтром.

И еще одно пояснение к шумам. При снижении коллекторного тока в предварительном усилителе можно уменьшить шумы. Следует только не забывать, что с уменьшением этого тока уменьшается статический коэффициент усиления по току, что может привести к уменьшению усиления по напряжению. В этом смысле лучше использовать транзисторы с большим усилением при малых токах коллектора, такие как КТ342, КТ3102 или КТ3107. Последние в некоторых случая предпочтительней. И процесс уменьшения тока коллектора для получения минимума шумов имеет некоторое экстремальное значение — дальнейшее уменьшение тока уже не приводит к уменьшению шумов.

Когда мы определяли ток, как направленное движение электрических зарядов, мы оговаривали, что это направленное движение в целом, вообще. Тогда как по отношению к каждому из зарядов нельзя сказать, будет ли он следовать этому направлению. То есть, часть зарядов в целом движется направленно, но отдельные заряды могут двигаться хаотически, почти не следуя заданному направлению. Чем больше электродвижущая сила, тем более «стройными рядами» движутся заряды. Однако хаотическое движение для них более естественно. Именно это хаотическое движение, существующее во всех материалах, обусловленное хотя бы температурой, и создает шумы. Шумят резисторы, шумят полупроводники. Все шумят, что нельзя не учитывать, работая с малыми сигналами.

Не слишком последовательно и не совсем верно, но поскольку речь шла о

радиоприемниках, давайте немного остановимся на смесительном каскаде и каскаде усиления промежуточной частоты супергетеродинного радиоприемника.

Чем плох радиоприемник прямого усиления. Он достаточно понятен, его проще сделать. Зачем все усложнять? В первую очередь проблемы с приемником прямого усиления в его усилительных возможностях. Можно попытаться увеличить количество каскадов усиления высокой частоты. Но я упоминал о сверхрегенеративной схеме, где положительная обратная связь вводится специальным, контролируемым образом. В многокаскадном усилителе, особенно высокочастотном, многочисленные конструктивные элементы – дорожки печатной платы, корпуса элементов – вносят паразитные емкости и индуктивности, которые трудно поддаются учету, а, следовательно, могут создавать трудно контролируемые обратные связи. Многокаскадные усилители высокой частоты из-за этого склонны к самовозбуждению. Раньше, когда основными активными элементами приемников и усилителей были радиолампы, каждый каскад приемника тщательно экранировался, цепи питания тщательно прокладывались с учетом возможных обратных связей по питанию. Но даже эти меры не спасали при работе со слабыми сигналами. Именно тогда возникла идея, лежащая в основе работы супергетеродинного приемника. Гораздо проще, чем уберегать широкополосный усилитель от паразитных связей, все усиление, необходимое для приемника, осуществить на одной рабочей частоте, используя «одночастотные» усилители. Эту частоту, для радиоприемников диапазонов длинных и средних волн, как правило, 465 кГц, назвали промежуточной частотой. Усилитель на одну частоту при достаточно большом усилении будет работать гораздо устойчивее, и гораздо легче учесть все вредные параметры конструкции. Чтобы получить эту промежуточную частоту из радиосигнала, выделенного входным LC-контуром, производится преобразование частоты: несущую частоту с помощью смесителя меняют на 465 кГц. Такое преобразование возможно в нелинейной цепи при одновременной работе двух источников переменного напряжения с близкими частотами.

В результате смешивания двух сигналов получаются сигналы с суммарной и разностной частотой при сохранении «огибающей», нашего информационного сигнала. Каскад усиления, в котором производится смешивание сигналов, называется смесителем и нагружается он обычно на контур, настроенный на промежуточную частоту. Так в смесителе образуется четыре сигнала: принятый радиосигнал, сигнал от перестраиваемого высокочастотного генератора, который называется гетеродином, и результат смешивания с разностной и суммарной частотой. Почему чаще используют разностный сигнал, а не суммарный, и тот, и другой, вполне подходящие кандидаты на роль промежуточной частоты, а потому, что чем ниже частота, тем легче с ней работать, но ее и нельзя слишком занижать по причине того, что она будет несущей частотой для звукового сигнала. Попробуем, на этот раз без жульничества, получить такое смешивание сигналов.

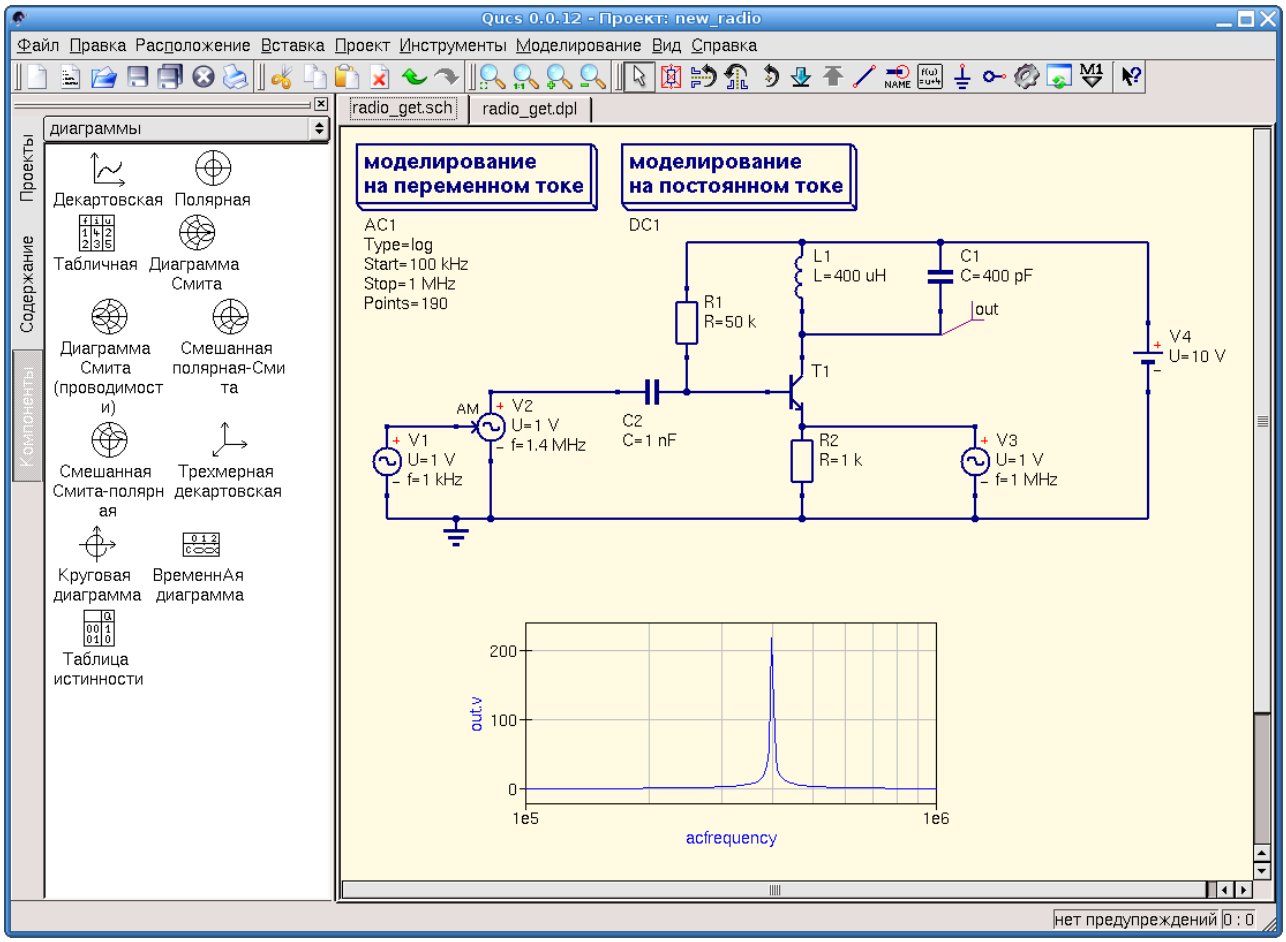


Рис. 11.2. Подбор LC-контура на частоту 400 кГц

Первым делом, уж если без жульничества, благо программа это позволяет, я без дополнительных расчетов пытаюсь подобрать (подогнать) параметры колебательного контура в коллекторной цепи транзистора. Я выбрал параметры генераторов так, чтобы получить промежуточную частоту 400 кГц (округлив это значение для простоты). Моделирование на переменном токе в программе Qucs дает мне АЧХ каскада, на которой видно, как частота 4*10e5 резко выделяется при усилении.

Для обеспечения синхронной подстройки частоты гетеродина к частоте принимаемой радиостанции в радиоприемниках часто применяют сдвоенный блок конденсаторов переменной емкости, с подвижными пластинами, расположенными на одной оси. При вращении ротора такого конденсатора емкость обоих блоков меняется синхронно, а значит синхронно меняется настройка входного и гетеродинного LC-контуров, с тем, чтобы разность частот сохранялась постоянной.

В первом эксперименте мне хотелось показать, что при этом преобразовании огибающая радиосигнала сохраняется. Поэтому все элементы схемы выбраны для получения достаточно наглядной картинки. Если сравнивать на рисунке ниже первую диаграмму выходного напряжения и вторую, которая относится к исходному радиосигналу, можно увидеть, что огибающая действительно сохраняется.

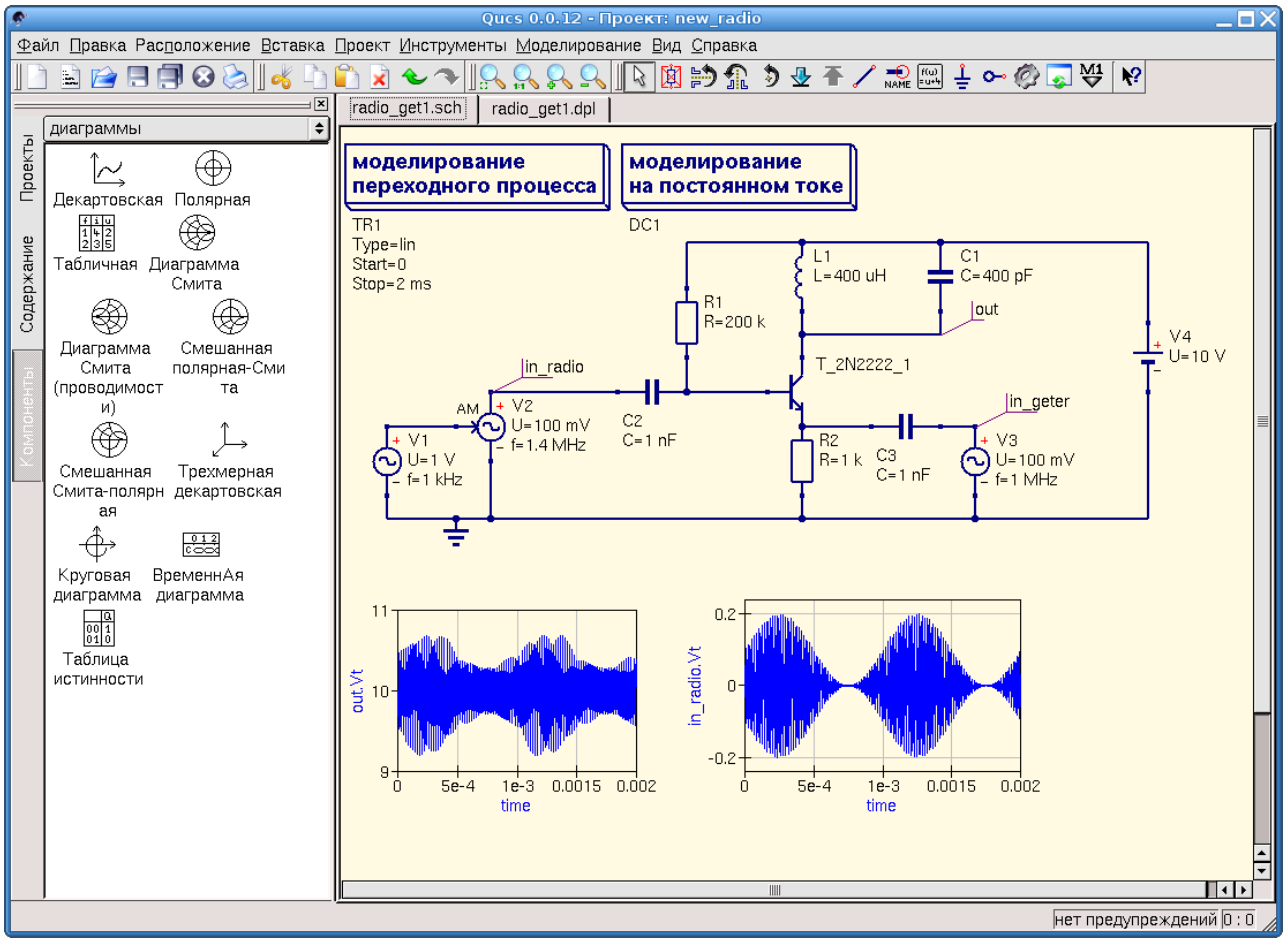


Рис. 11.3. Пример работы гетеродина

А сейчас, несколько изменив схему, я хочу заменить генераторы V1 и V2 обычным с рабочей частотой 1.4 МГц, посмотрим, действительно ли на выходе смесителя мы получим сигнал промежуточной частоты. Удалив генератор, формирующий амплитудно-модулированный сигнал, мы на входе смесителя получаем обычную синусоиду. Такой же сигнал, но другой частоты, приходит в цепь эмиттера транзистора. Если смешивания не произойдет, то на выходе каскада мы можем получить только очень слабые, из-за подавления LC фильтром, сигналы этих двух частот, вид которых не должен отличаться от вида сигналов, подаваемых на вход и в цепь эмиттера.

Если же смешивание сигналов будет иметь место, вид сигнала (*out* на схеме) каким-то образом должен нам это показать.

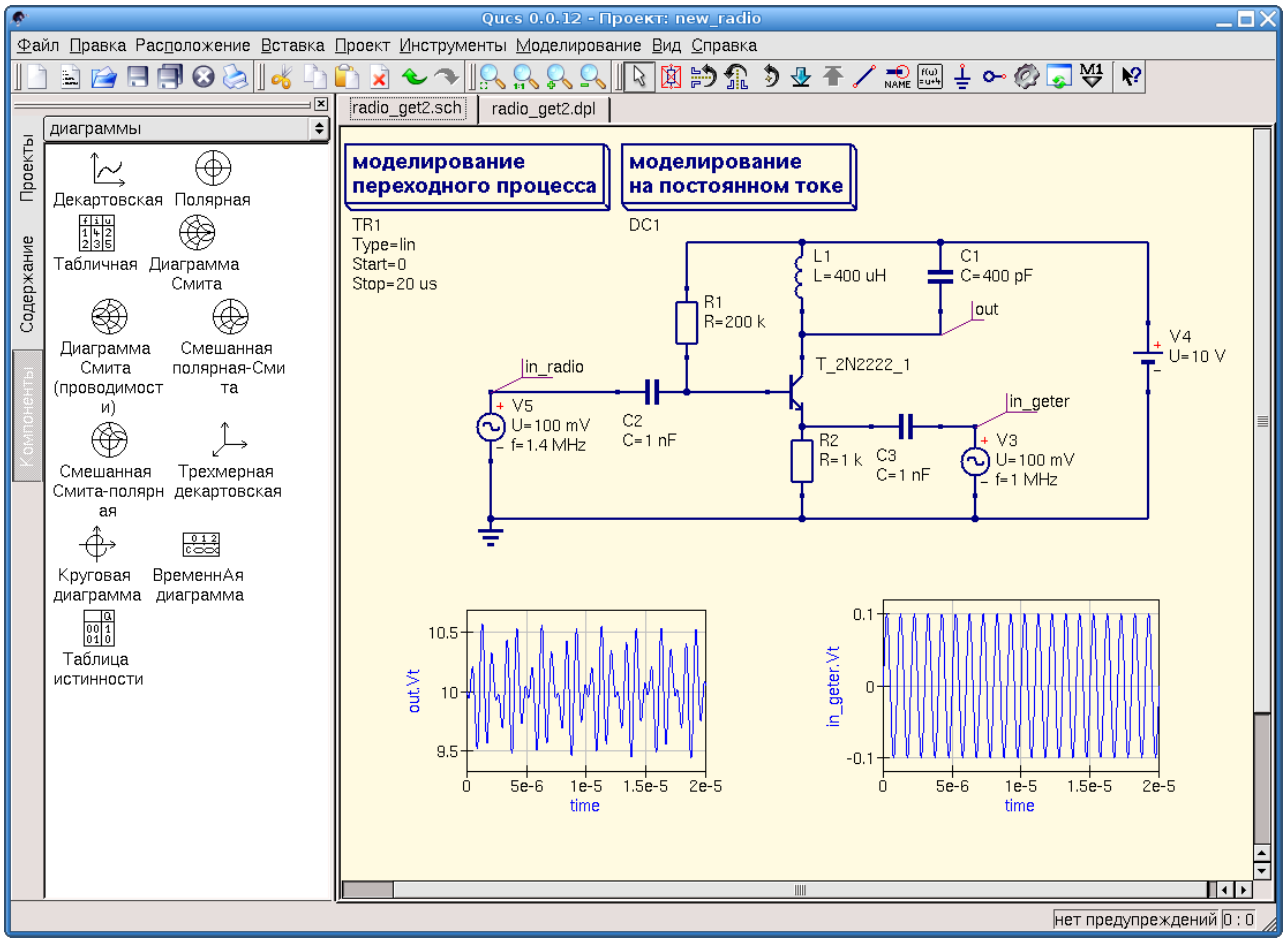


Рис. 11.4. Вид смешанного сигнала

Хотя результат не слишком впечатляет, остались сигналы с частотой гетеродина, но сигнал «биений» двух генераторов, который похож на огибающую в амплитудно-модулированном сигнале радиочастоты, в данном случае и будет сигналом промежуточной частоты, имеющей период, примерно, вдвое ниже периода сигнала гетеродина.

Каскады усиления промежуточной частоты могут незначительно отличаться от того, что на рисунке, исключая, конечно, и генератор, имитирующий радиосигнал, и генератор, имитирующий гетеродин. В качестве фильтра раньше использовались в основном LC-контуры (как правило, более сложные LC-фильтры), позже стали применять фильтры, скажем, на основе поверхностных акустических волн (ПАВ-фильтры).

Детектор в таком радиоприемнике может не отличаться от выше описанного, оставаясь выпрямителем радиоволны.

Сделав экскурс в схемы обработки радиосигнала, вернемся к предварительным усилителям НЧ. Многие радиоприемники имеют в своем составе регуляторы тембра низких и высоких частот.

Это, в сущности, активные или пассивные фильтры. В последнем случае схема представляет собой некоторый набор резисторов и конденсаторов. Например, такой:

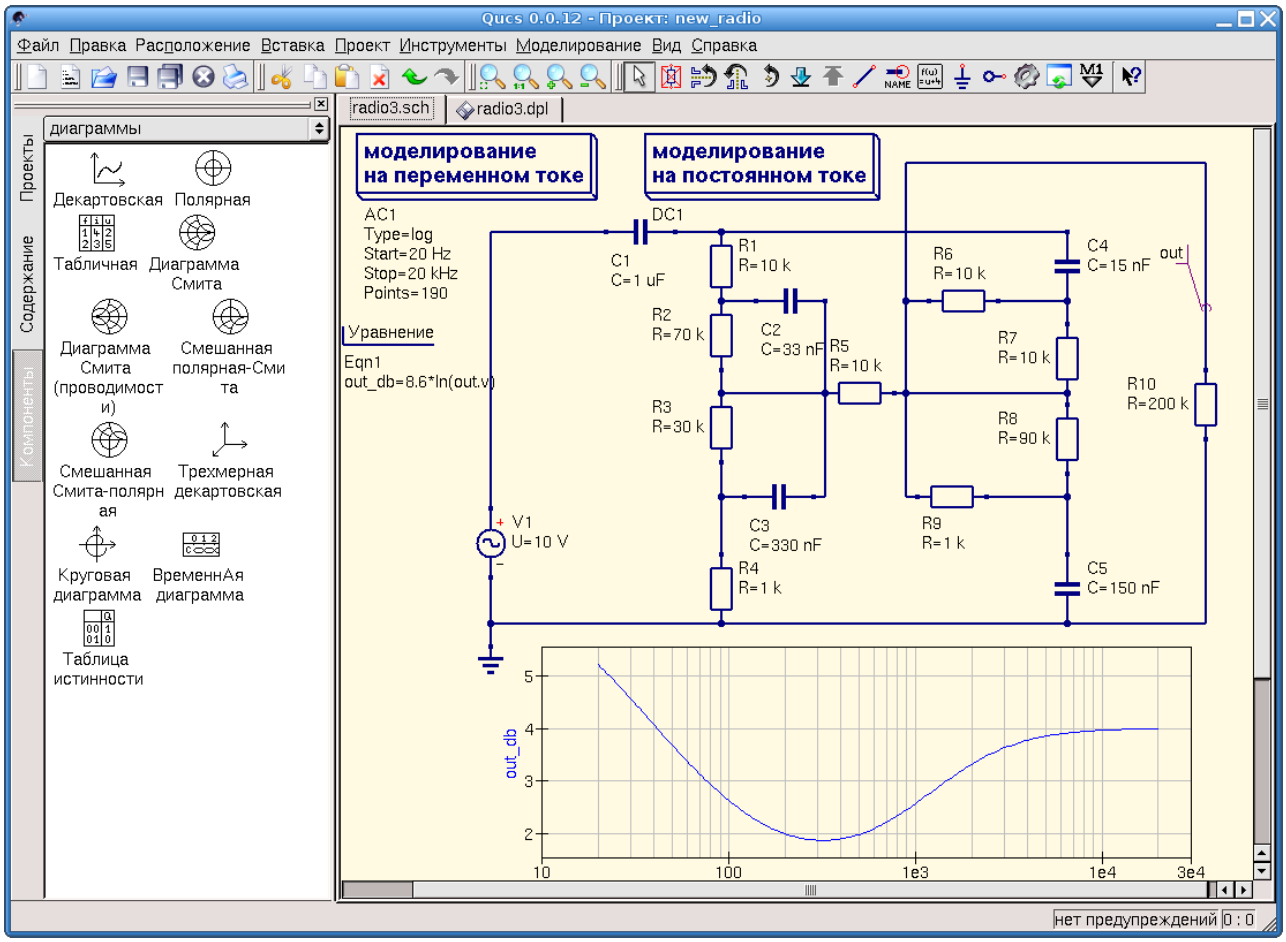


Рис. 11.5. Пассивный регулятор тембра

Уравнение, напомню, добавлено для выражения амплитудно-частотной характеристики (в точке *out*) в децибелах. Регулятор может поднять уровень высоких и низких частот, например, для соответствия их уровня кривым равной громкости при снижении общего уровня громкости.

В схеме делители напряжения R2R3 и R7R8 в действительности переменные сопротивления с логарифмическим характером изменения сопротивления в зависимости от угла поворота.

Изменяя значения резисторов делителей напряжения, не забудьте, что сумма должна равняться 100 кОм, можно проверить поведение фильтра при разных «углах поворота» переменных резисторов. Можно проверить и влияние, например, входного конденсатора C1 и сопротивления нагрузки R10 на результирующую амплитудно-частотную характеристику. То же можно сказать и об остальных элементах цепи. Даже если полученные данные будут несколько отличаться от того, что вы увидите, собрав схему на макетной плате, характер этих изменений не изменится.

Применение пассивных корректоров АЧХ бывает неудобно хотя бы из-за уменьшения уровня сигнала. Чтобы этого не происходило, можно применять активные регуляторы тембра. Часто цепи коррекции добавляют в цепь отрицательной обратной связи усилителя. Пример такого построения регуляторов тембра приведен ниже.

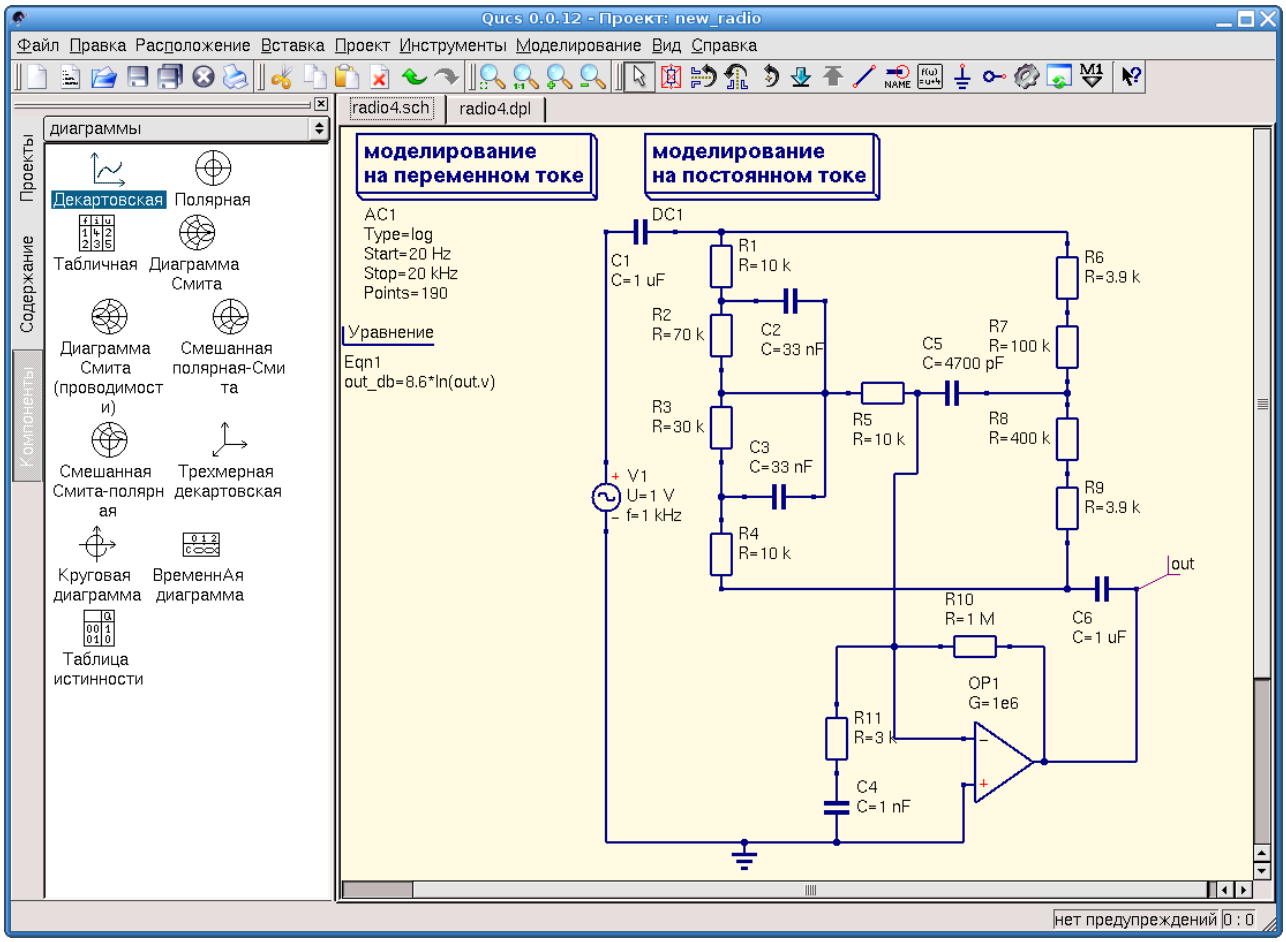


Рис. 11.6. Активный регулятор тембра

Немного «аляповатый» рисунок схемы не позволил мне добавить диаграмму, приведу ее отдельно. И обратите внимание, один из регуляторов выполнен на потенциометре 100 кОм, второй 500 кОм, и оба должны иметь линейную характеристику регулировки.

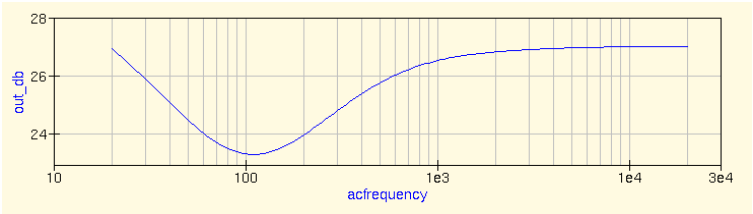


Рис. 11.7. Диаграмма АЧХ регулятора тембра

Применение операционных усилителей в предварительных усилителях вполне удобно и оправданно, но не следует забывать, что верхняя граничная частота операционного усилителя зависит от его коэффициента усиления, определяемого элементами отрицательной обратной связи. И еще одно, операционные усилители требуют двух-полярного источника питания. Во всяком случае, с таким источником питания их значительно удобнее применять. А многие устройства, особенно переносные, используют одну батарейку для питания всего устройства. В этом случае можно использовать приемы получения искусственной средней точки для «общего провода» операционного усилителя, или использовать преобразователи для получения второго источника питания.

Фильтры, подобные вышеприведенным, можно с успехом применять и в своих разработках, и в своих экспериментах. Например, при экспериментах с громкоговорителями можно использовать простые регуляторы тембра, которые можно и упростить в некоторых случаях. Программа позволит вам понять, что нужно (и как этого добиться), если вы хотите получить определенные изменения в работе регулятора.

Получение двух-полярного напряжения из одно-полярного мы рассмотрим ниже. А то, как включить операционный усилитель в схему с одним источником питания, рассмотрим сейчас.

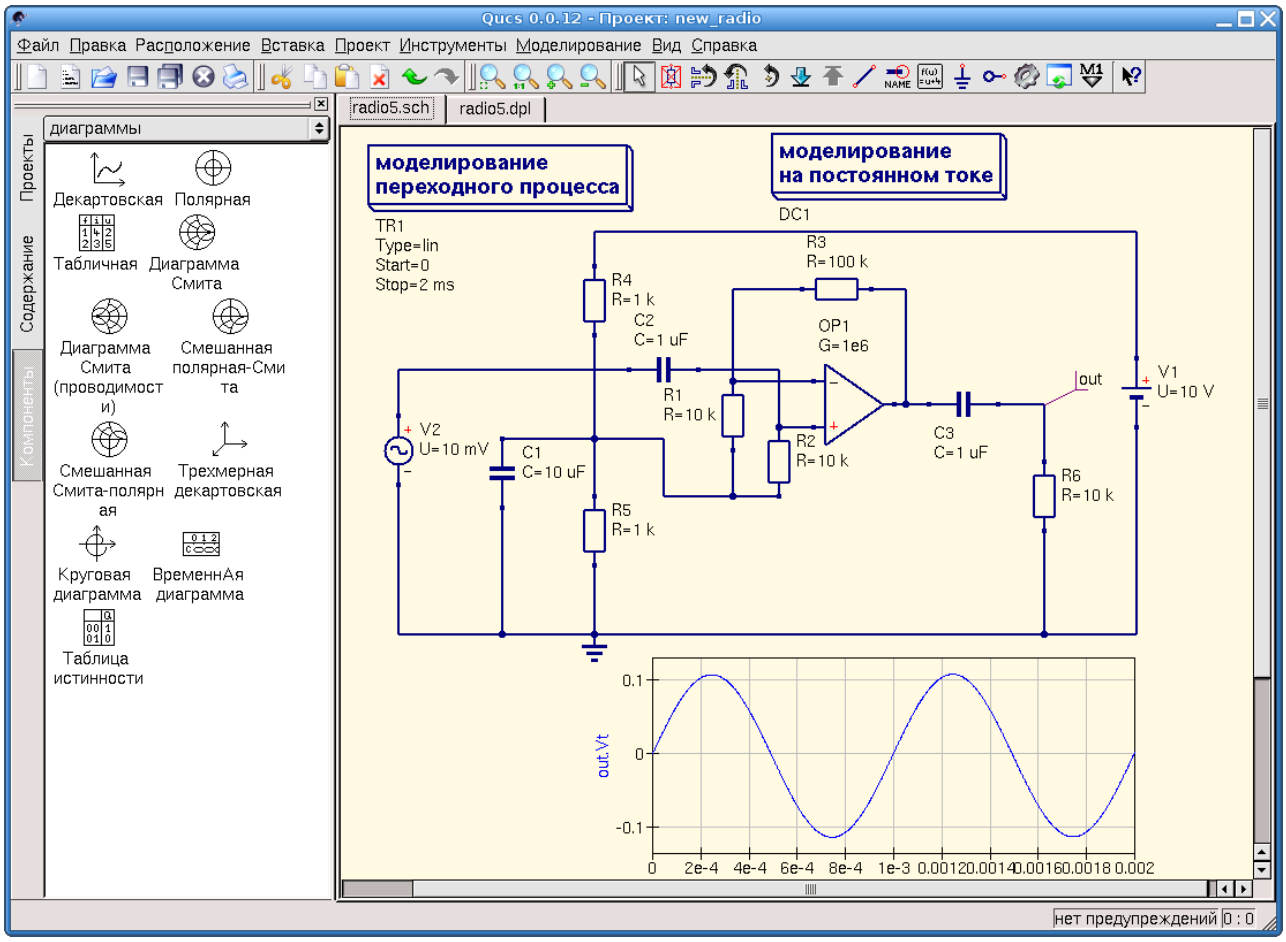


Рис. 11.8. Включение операционного усилителя с одно-полярным питанием

Применять операционный усилитель для усиления постоянного тока с таким включением неудобно, но для усиления переменного тока это вполне подходящий вариант. Можно делитель напряжения R4R5 заменить парой подходящих стабилитронов. Схема включения операционного усилителя получается достаточно «классической».

Снижение шумов и автоматическая регулировка

Самый надежный способ избавиться от вредных привычек – это никогда их не приобретать. Это же можно отнести и к проблеме шумов. Если схема достаточно хорошо продумана, если приняты все меры по уменьшению шумов, то применять какие-либо устройства, снижающие уровень шумов, нужды не будет. Исключения составляют те случаи, когда вы получаете шумы в «обязательном» порядке, а их количество вас явно не устраивает.

Во времена повсеместного применения магнитофонов борьба с шумами магнитной ленты шла не шуточная. Большим трудом давалось снижение шумов на несколько децибел. Наиболее успешно этот процесс проходил на Dolby-фронте. Шумоподавители этого класса работали по компадерному принципу. Перед записью на ленту сигнал сжимался с помощью схемы компрессора, что позволяло записать его «над уровнем» шума ленты, а при воспроизведении сжатый сигнал «расширялся» с помощью экспандера, работающего с «зеркальными» параметрами расширения по отношению к компрессору. За счет этого шумы отодвигались на задний план достаточно существенно. Существовало несколько вариантов построения схем шумоподавления, которые отличались, например, тем, что исходный сигнал проходил через несколько каналов, каждый из которых работал в своей полосе частот. С появлением цифровых методов записи все эти устройства отошли в сферу профессиональной деятельности, а любители, если и используют эти схемы, то, надеюсь, применяют для этих целей готовые микросхемы.

Вместе с тем некоторое представление об устройствах снижения уровня шумов по моему мнению должен иметь каждый любитель. Никогда не знаешь, с чем ты столкнешься в своей практике. Даже в профессиональной деятельности, которая, согласитесь, гораздо жестче регламентирована, решение тех или иных задач, время от времени уводит далеко в сторону от проторенной дороги. А любитель с его неподдельным интересом ко всему на свете видит проторенные дороги только во сне.

Сейчас я совершенно уверен в двух вещах: при необходимости вы всегда найдете подходящую схему, и при желании сами сможете ее собрать и настроить. Поэтому не буду приводить схем, даже функциональных, а вместо этого предлагаю рассмотреть некоторые элементы подобных схем в предположении, что вам известны базовые идеи (где-то слышали, или прочитали, или придумали сами).

Что вы могли об этом услышать? В первую очередь об особенностях восприятия человеком мешающего воздействия шума, что особенно характерно для верхней части звукового диапазона. К особенностям этого восприятия можно отнести эффект маскирования шума полезным сигналом. Если в звуковом фрагменте присутствуют сигналы достаточной громкости с частотами выше 2-3 кГц, то человек не слышит сопутствующих шумов в этой части звукового спектра. Они становятся слышны только при тихих звуках или в их отсутствии. С другой стороны, все тихие звуки, как правило, не имеют ярко выраженных обертонов. И если мы в отсутствии громких звуков и звуков высоких частот ограничим частотный диапазон усилителя, скажем, выше нескольких килогерц, то шумы будут подавляться и не будут нам мешать. Такова идея, она может оказаться несостоятельной, но я хотел бы показать, как можно подступиться к ее проверке, используя то немногое, что было написано в предыдущих главах.

В первую очередь посмотрим, как мы могли бы управлять полосой воспроизводимых частот. Представим, что мы собрали одну из схем регулятора тембра, описанных выше. У нас есть регулятор высоких частот, выполненный на переменном резисторе. Мы включаем музыку, и когда звук тихий и нет ни колокольчиков, ни тарелок, мы «убираем» высокие частоты, а с появлением высоких звуков прибавляем их. То есть, появляется две задачи: первая требует выделить высокие звуки с оценкой их громкости, вторая «покрутить» регулятор высоких частот.

В качестве базы для построения схемы используем предварительный усилитель. Решая первую задачу, сигнал со входа этого усилителя мы отправим на фильтр, который срезает низкие частоты, скажем, от 6 кГц и ниже. После фильтра мы поставим выпрямитель (похожий на детектор приемника). Теперь выпрямленное напряжение на выходе выпрямителя будет пропорционально громкости высоких частот в исходном сигнале. Проверим это.

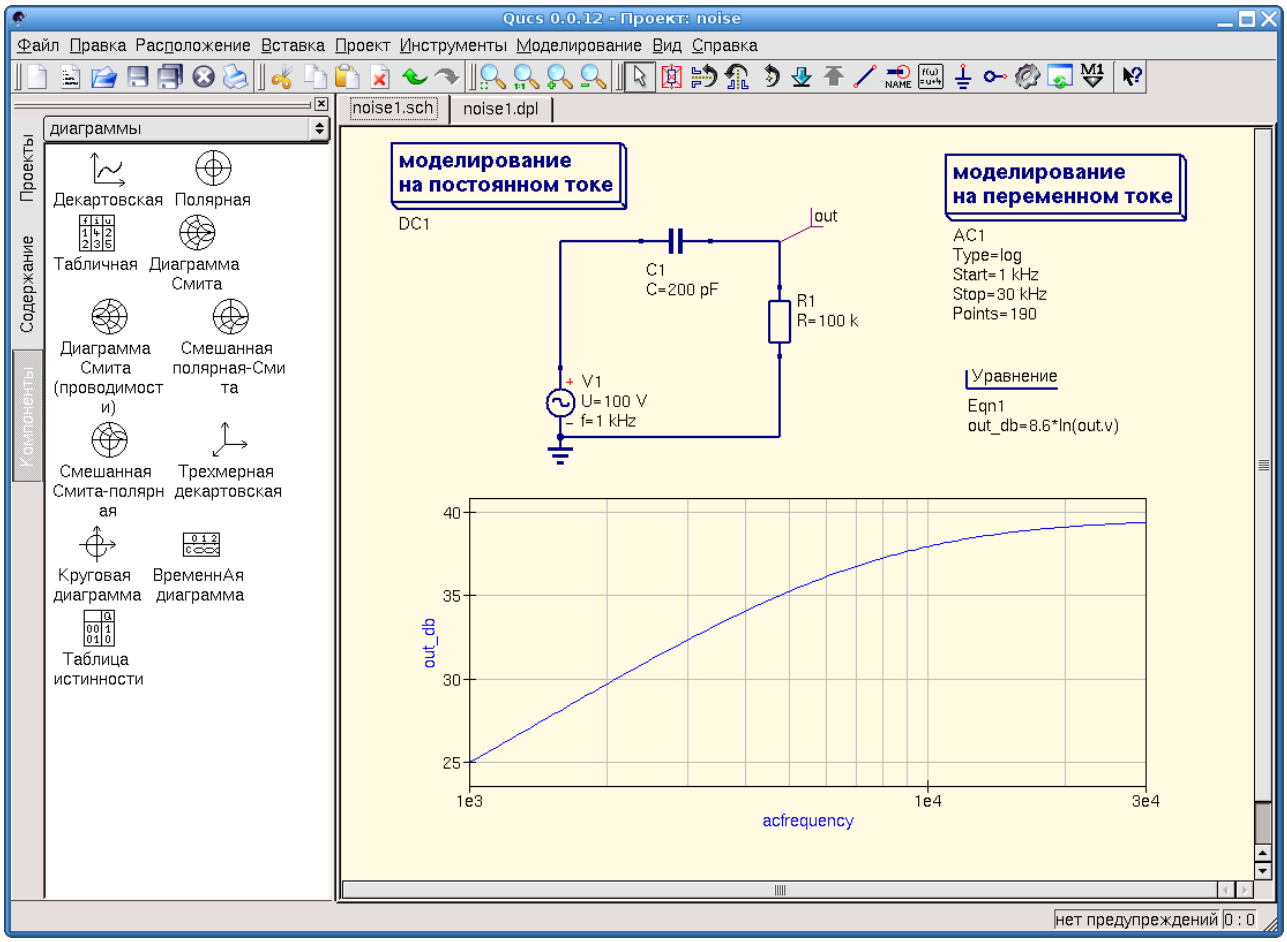


Рис. 11.9. RC-фильтр

Как видно на диаграмме, низкие частоты не будут присутствовать на выходе фильтра. Теперь добавим выпрямитель, диод, и посмотрим, что у нас происходит на частоте 10 кГц.

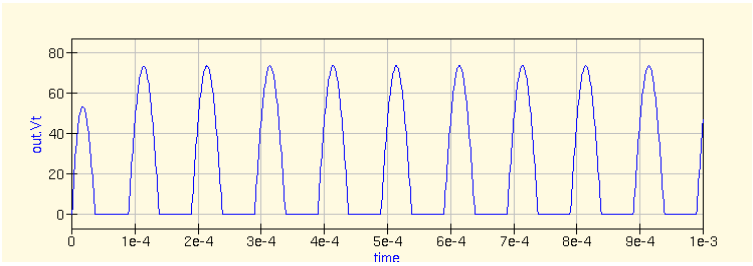


Рис. 11.10. Сигнал с фильтра после выпрямления

Как и следовало ожидать, мы получили «пульсирующее» напряжение на выходе. Оно не вполне подходит для цели управления, но мы знаем, что пульсации на выходе выпрямителя устраняются добавлением сглаживающего конденсатора.

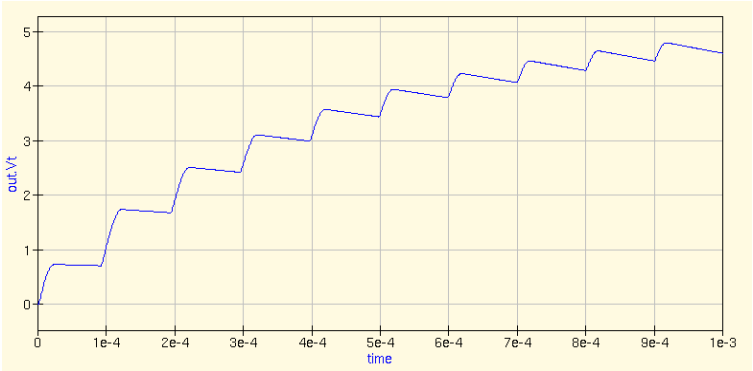


Рис. 11.11. Сглаженное управляющее напряжение

Впрямь, конденсатор помог. Можно увеличить его значение, что приведет к еще лучшему «сглаживанию». Но... отчего-то напряжение увеличивается странным образом. Оно не сразу принимает нужное значение, а плавно поднимается к нему по образовавшимся «ступенькам», оставшимся от пульсаций. Чем больше мы будем увеличивать конденсатор, тем дольше придется ждать, когда напряжение достигнет своего значения. Применяя такую схему для получения управляющего напряжения мы сталкиваемся с некоторыми проблемами. С одной стороны, мы хотели бы использовать в качестве управляющего не очень «дергающееся» напряжение, с другой стороны, мы хотели бы, чтобы схема достаточно быстро реагировала на изменение ситуации. Чтобы оценить реакцию схемы на происходящие изменения, я попробую заменить в предыдущем эксперименте источник переменного напряжения с частотой 10 кГц на источник прямоугольных импульсов с частотой, скажем, 4 кГц (период 500 мкС), оставив остальную часть схемы без изменений. Быть может, страхи мои беспочвенны, а сомнения напрасны, тогда в качестве конденсатора можно использовать электролитический конденсатор большой емкости.

Почему мне приходит в голову использовать источник прямоугольных импульсов? Думаю, потому, что резкий переход от низкого уровня напряжения к высокому, и наоборот, более всего ассоциируется у меня с понятием реакции схемы на внешнее воздействие. С помощью генератора прямоугольных импульсов легче заметить наличие собственных колебаний в параллельном LC-контуре. С его помощью легче было оценить влияние интегрирующей RC цепи. Не помогут ли прямоугольные импульсы и сейчас?

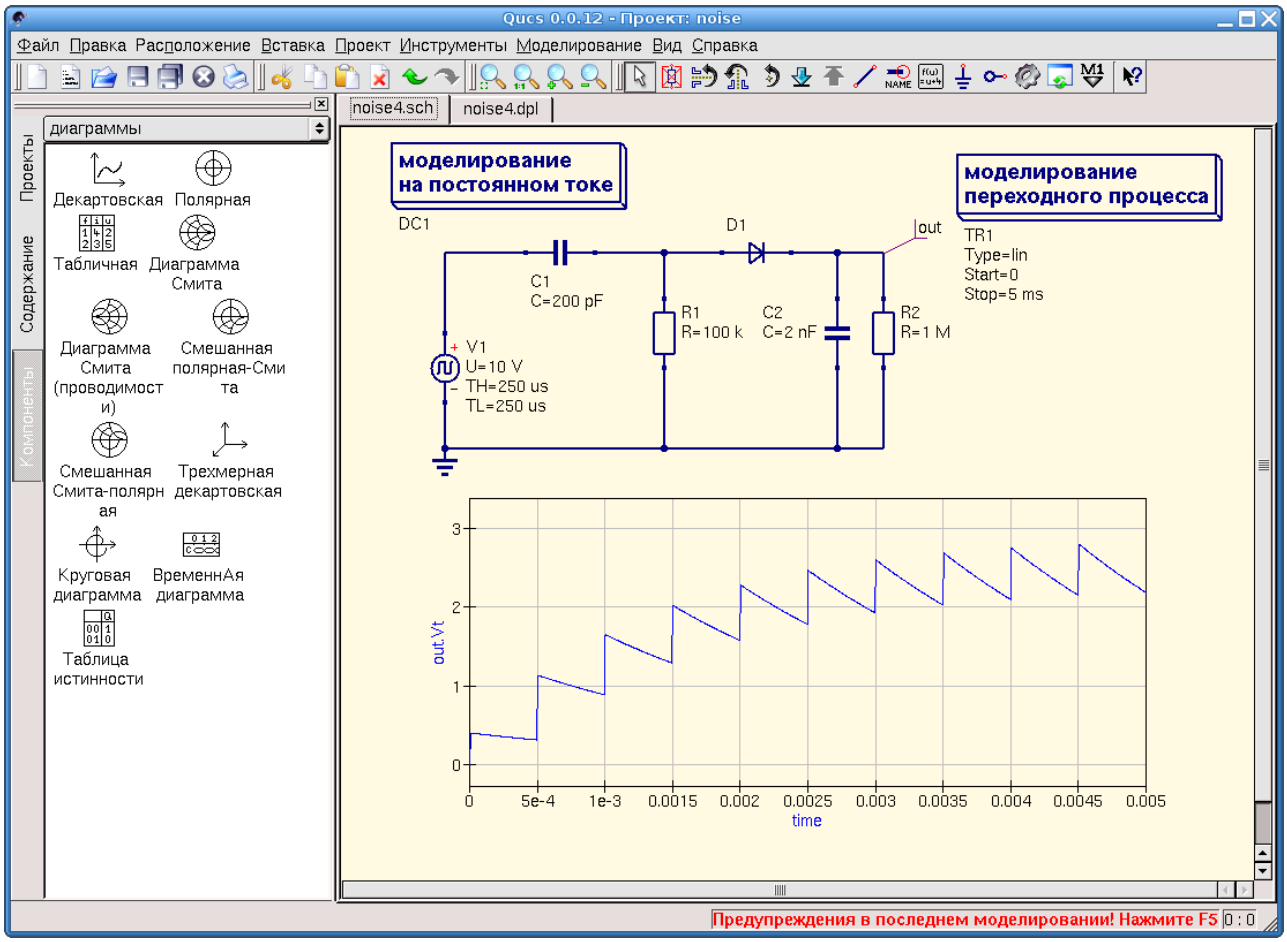


Рис. 11.12. Оценка динамических качеств схемы

Что я могу сказать, каждый импульс быстро заряжает конденсатор, после каждого импульса конденсатор медленно разряжается, не успевая разрядиться полностью, и за время около 3 мС он успевает зарядиться почти полностью. Достаточно этого или нет? Не знаю. Я хочу попробовать заменить источник прямоугольных периодических импульсов на источник одиночного прямоугольного импульса, но даст ли мне это больше?

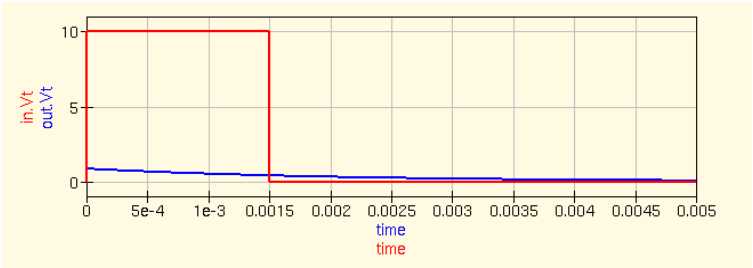


Рис. 11.13. Реакция схемы на одиночный импульс

И что? Пока ничего нового. Да, каждое изменение уровня будет приводить к появлению управляющего напряжения, которое быстро появляется, которое затем медленно убывает, и за время в несколько миллисекунд все заканчивается. Достаточно ли этого?

Не знаю, но пока оставим и будем считать, что у нас есть управляющее напряжение (что-то управляющее нам нужно в любом случае).

Рассмотрим работу схмы тембра высоких частот.

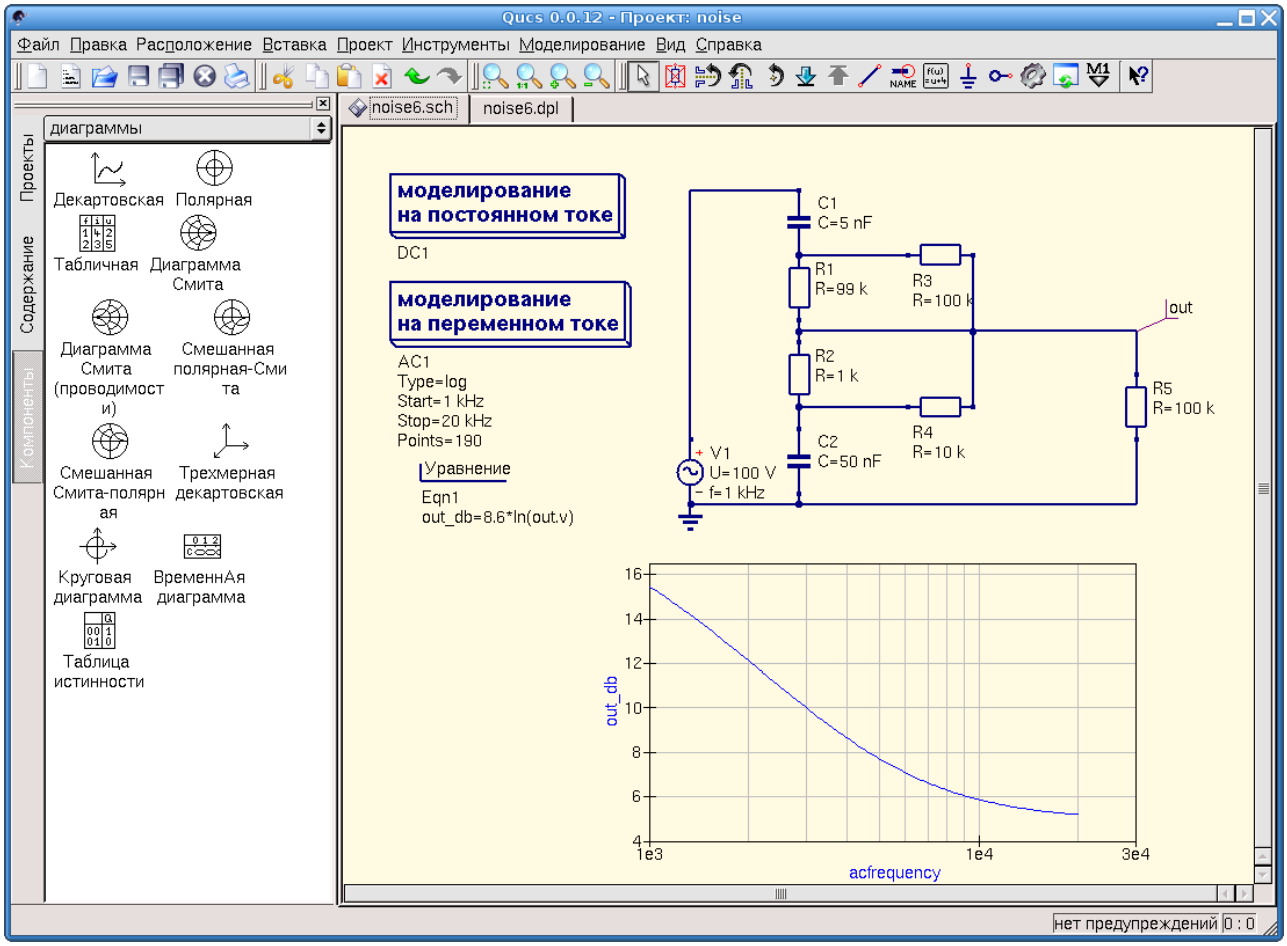


Рис. 11.14. Фильтр, обрезающий высокие частоты

Не самый-самый, но остановимся на этом фильтре, который при значениях R1 = 99 кОм и R2 = 10 кОм (R2 = 10 кОм!) имеет следующую амплитудно-частотную характеристику:

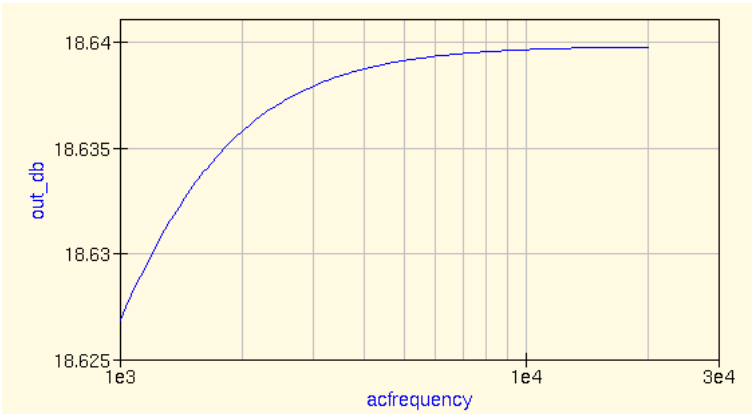


Рис. 11.15. АЧХ при втором положении регулятора

Такую АЧХ вполне можно считать линейной. Теперь наша задача формулируется так: какой из известных нам элементов мы можем применить для изменения его сопротивления от 10 кОм до 1 кОм при изменении управляющего напряжения?

Выше я уже говорил, что в усилителе мощности транзистор в некотором смысле можно рассматривать как регулируемое сопротивление, почему бы мы мне не повторить эту «крамольную» мысль еще раз! Транзистор, на базу которого мы подадим управляющее напряжение, мы используем для замены R2. Еще лучше, это я знаю, будет работать в этом качестве полевой транзистор. Не знаю только, получится ли эта замена в программе Qucs. Уж очень много я от нее хочу.

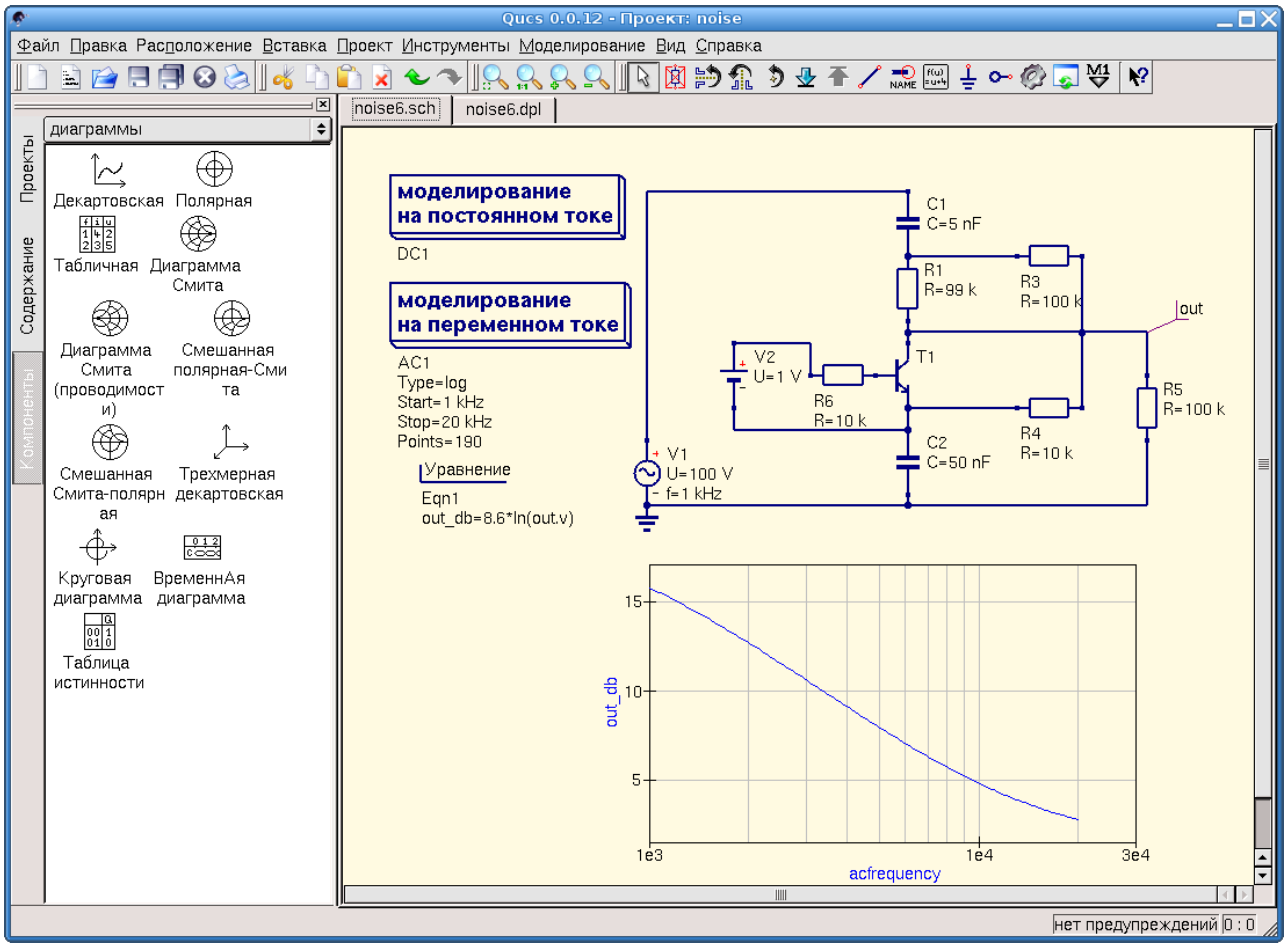


Рис. 11.16. Применение транзистора в качестве управляемого резистора

Диаграмма напоминает мне рисунок 11.14. А вот другая диаграмма, полученная при изменении напряжения V2 с 1 В на 10 В.

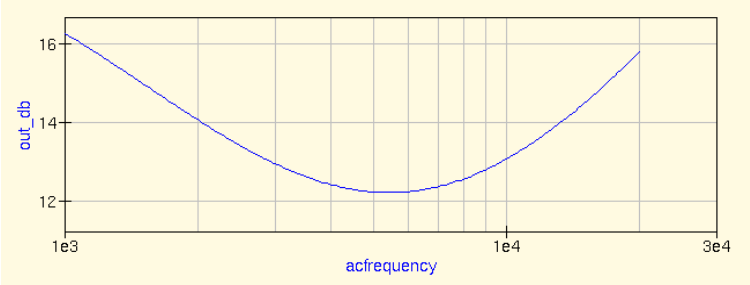


Рис. 11.17. АЧХ при изменении напряжения в базовой цепи транзистора

Не совсем, возможно, то, что нужно, но что нужно, я и сам еще не знаю, однако нет

сомнений, изменение базового тока меняет характер АЧХ фильтра. Что и требовалось.

Итак, шумоподавитель, изобретением которого мы занимались, в общих чертах «нарисовался»: фильтр для выделения высоких частот из сигнала, выпрямитель для получения из этих частот управляющего напряжения, и регулятор АЧХ тракта с применением транзистора, управляемого от выпрямителя.

Я не вижу особого смысла продолжать построение шумоподавителя, при необходимости лучше взять готовую микросхему, а все приведенные эксперименты нужны были только для того, чтобы понять сущность работы схемы, получить дополнительный опыт работы со схемами, но более всего мне хотелось показать, как транзистор играет роль управляемого резистора.

А это мне нужно для рассказа об еще одной разновидности предварительного усилителя — с автоматической регулировкой уровня, что мне нужно для плавного перехода к разговору об автоматическом регулировании и автоматике вообще. Но вначале автоматическое регулирование уровня громкости (впрочем, чего угодно).

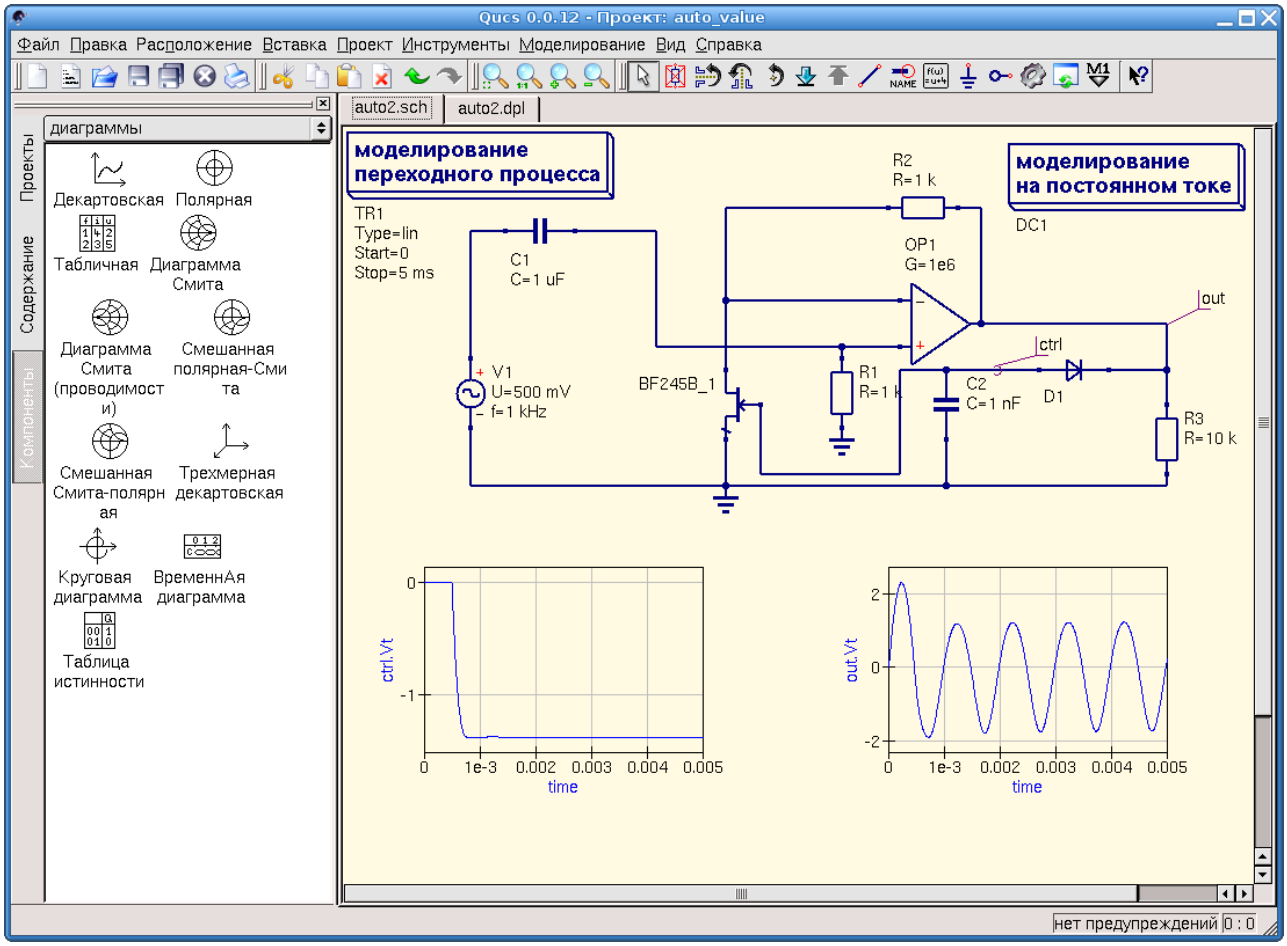


Рис. 11.18. Предварительный усилитель с автоматической регулировкой усиления

На схеме полевой транзистор входит в делитель напряжения, образующий отрицательную обратную связь. Отношение сопротивления резистора R2 к сопротивлению исток-сток транзистора определит коэффициент усиления каскада по напряжению. В свою очередь, сопротивление полевого транзистора определится напряжением на его затворе, полученном за счет выпрямления выходного напряжения диодом D1.

На первой диаграмме видно, что напряжение выпрямителя (*ctrl*) первоначально равное нулю возрастает до -1.3 В, закрывая транзистор, что увеличивает сопротивление его канала. А на второй диаграмме видно, что пока напряжение на затворе равно нулю, выходное напряжение (амплитуда) около 2 В. С появлением запирающего напряжения на затворе (время на диаграмме 1 мС) напряжение на выходе усилителя падает. Если в схеме менять входной сигнал от источника V1 в пределах от 300 мВ до 600 мВ (изменение в два раза), то выходное напряжение изменится очень незначительно. Что и демонстрирует возможность автоматической регулировки усиления. Реальные схемы, применяемые для этих целей, отличаются не столь разительно от приведенной, а автоматическая регулировка усиления используется и в телевизорах для оптимизации уровня радиосигнала, и в радиоприемниках, и в телефонах, словом, везде где можно или нужно автоматически регулировать сигнал. А зачем его вообще регулировать автоматически, зачем нужна автоматика?

Глава 12. Автоматика, и зачем она нужна

Осень приносит не только ненастье и дожди, появляются новые версии операционных систем, установленных на моем компьютере. Меня давно устраивают и прежние, но укоренившаяся привычка ожидать перемен в графической оболочке, правильнее в Linux ее называть «оконный менеджер», заставляет меня обновлять текущую версию. Графическая оболочка, буду называть ее так, больше остального интересует обычного пользователя, каким являюсь я. Linux предлагает ее в разных вариантах, временами я просматриваю их, но пользуюсь, как правило, одной, которая устанавливается по умолчанию. Что я хотел сказать, когда заговорил об операционной системе? Она тоже, как и чайник, как и обогреватель, подвержена автоматике. Есть ее раздел, который называется «автоматическое обновление». Для меня автоматика больше ассоциируется с физическими предметами: датчиками, управляющими устройствами, исполняющими устройствами и т.д. Даже в том, что я предпочитаю называть оконный менеджер графической оболочкой, сказывается мое пристрастие к «железу». Для меня название «оконный менеджер» больше подошло бы устройству, которое автоматически отслеживает количество углекислого газа в комнате, и при его превышении заданной нормы открывает окно для проветривания или включает вентиляцию. И пусть этот менеджер следит за освещенностью. Когда становится темно, закрывает шторы на окне. Вот такой «оконный менеджер» мне понятен.

Как некогда паровые машины избавили людей от монотонного физического труда, так автоматика избавляет от монотонного «умственного» труда. Наверняка у вас дома много устройств, которые делают что-то автоматически. По случаю осени у меня в комнате стоит электрический обогреватель. Старые электрические обогреватели (калориферы) включались в розетку, грелись, и вам, время от времени, приходилось их выключать, затем по мере остывания воздуха в комнате вновь включать. Современные обогреватели избавляют вас от необходимости следить за температурой в комнате. По тому же принципу действует сегодня и электрический утюг, и электрический чайник, и, что у меня порой вызывает удивление, электрическая гирлянда, которую в скором времени предстоит водружать на елку. Что же изменилось в обогревателе с «древних» времен?

Обогреватель-автомат или простые устройства автоматики

Сам обогреватель – это большой резистор, который, как мы давно знаем, при протекании по нему тока нагревается. Мы можем даже сказать, что на нем рассеивается: $P = U^2 /R$, мощность. И этому сопротивлению безразлично, переменный это ток с действующим значением напряжения U, или постоянный с тем же напряжением. Мы это давно знаем.

Нам не составит труда нарисовать схему из выключателя и сопротивления. Это будет схема «старинного» обогревателя. И, думаю, не составит труда заменить выключатель устройством управления, для которого мы готовы составить функциональное описание:

- измерять температуру воздуха;
- при ее значении ниже величины А, включать обогреватель;
- при ее значении выше величины Б, выключать обогреватель.

Дополнительно можно определить, что значения А и Б должны допускать изменение, то есть, устанавливаться вручную. Это добавит функциональной гибкости нашему устройству.

Кроме того, мы уже знаем, что на изменение температуры реагируют многие электрические элементы: резисторы, проводники и полупроводники. Последние реагируют очень хорошо, заставляя нас выбирать схему включения транзисторов с учетом возможных

температурных «передряг». Выберем в качестве чувствительного к температуре элемента разновидность резистора, которая называется терморезистор. Скорее всего он будет изготовлен из полупроводникового материала, но для нас представляет интерес не это, а то, что его сопротивление меняется при изменении температуры гораздо интенсивнее, чем сопротивление обычного резистора. В качестве сигнала, несущего информацию о температуре, выберем падение напряжения на этом терморезисторе, который включим последовательно с обычным резистором в цепь источника постоянного тока. Получается обычный делитель напряжения. И мы твердо знаем, что напряжение на датчике (терморезисторе) пропорционально его сопротивлению.

Если бы можно было параллельно этому терморезистору включить реле, предполагая, что оно включится, когда напряжение достигнет нужной величины, то мы могли бы контактами реле включать нагревающий элемент. Схема была бы простой. Но так, скорее всего, не получится. Посмотрим, что нам мешает.

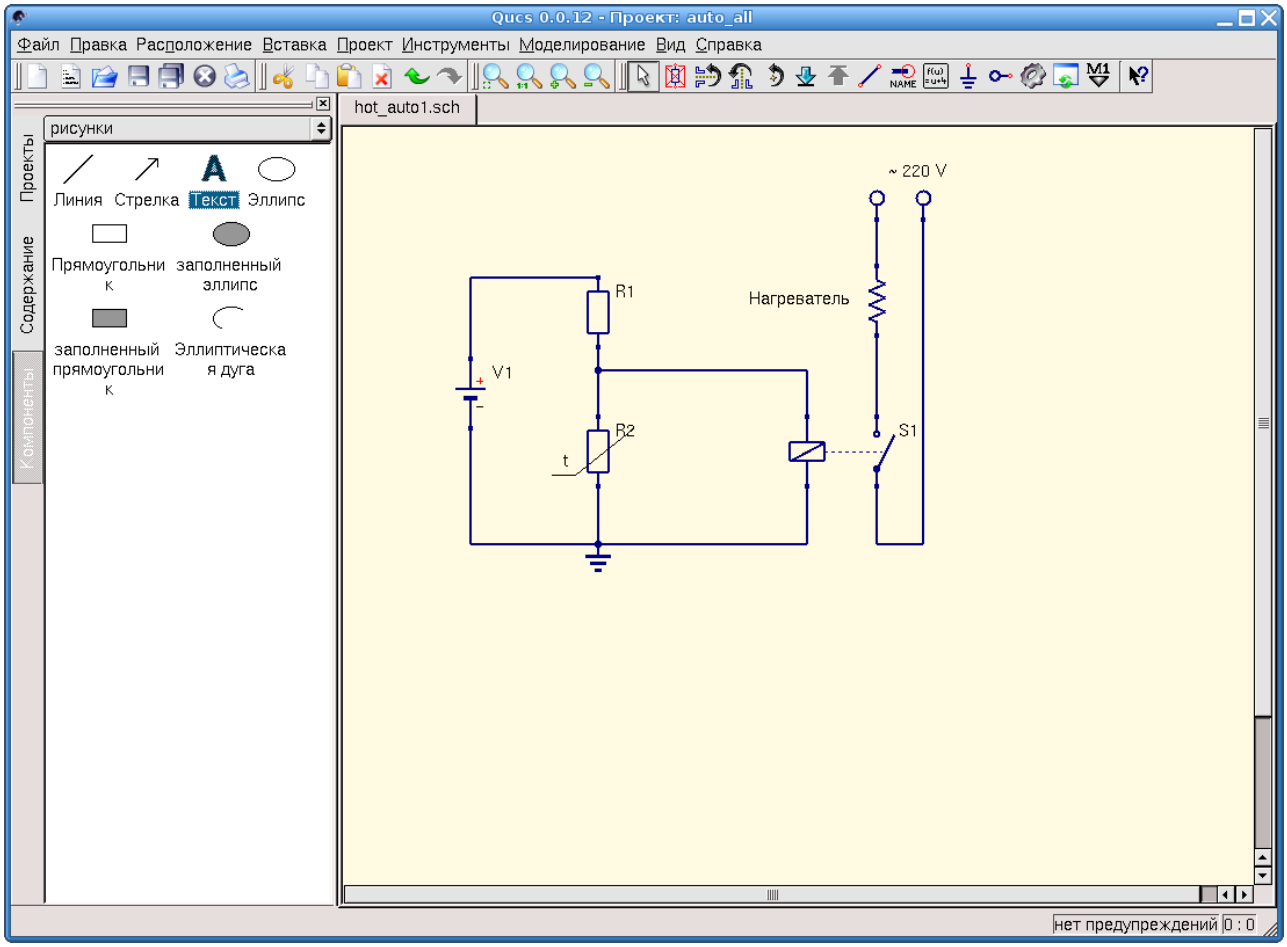


Рис. 12.1. Самая простая схема обогревателя-автомата?

Пока температура не достигла нужной величины, напряжение на резисторе R2 было бы ниже напряжения срабатывания реле, его контакты (нормально замкнутые) замыкали бы цепь нагревателя, а нагреватель нагревал бы воздух в комнате и, вместе с ним, терморезистор. Когда температура воздуха достигает нужной величины (то есть, сопротивление R2 увеличивается достаточно), реле включается и отключает нагреватель. Если бы такая схема работала, было бы очень хорошо (вообще-то, это не невозможно), но что вызывает сомнения? Во-первых, сопротивление терморезистора при нагревании может не увеличиваться, а уменьшаться. В этом случае говорят, что ТКС отрицателен (температурный коэффициент

сопротивления). Но это не главное.

Термосопротивление приходится выбирать с учетом параллельного его включения с сопротивлением обмотки реле, при этом желательно, чтобы его величина была много меньше сопротивления обмотки. Мало того, его сопротивление должно меняться очень сильно, иначе разница в падении напряжения, скажем, при 18 и при 25^0C будет почти неизменной. Попробуем обойти возникшие проблемы, но прежде хочу сознаться, что решение, даже проще нарисованного мной, есть, и давно известно, и давно используется. Это биметаллические контакты. Две плоские пластины из разных металлов с разным температурным коэффициентом расширения соединены своими плоскостями так, что при нагреве этот «бутерброд» изгибается в одну сторону. При остывании эти пластины восстанавливают первоначальную геометрию. Таким образом, при нагревании, снабдив один конец контактом, можно отключать нагреватель, а при остывании включать. Думаю, что так устроен автомат утюга. Знаю, что так работает прерыватель в елочной гирлянде, где биметаллическая пластина нагревается протекающим по ней током.

Такие простые решения вызывают у меня восторженное удивление человеческой изобретательностью, но, как правило, такие решения не бывают универсальны. Давайте найдем решение, которое способно решить и проблему нагревателя, и, если понадобится, проблему включения вентиляции, и включения освещения, и т.д. Я не исключаю, что мы не найдем наилучшее решение в данной главе, тогда будем искать его в следующих.

Предыдущие главы мы посвятили рассмотрению усилителей. Применим полученные знания к решению задачи. Очень удобный усилитель получается из операционного усилителя, хотя и не обязательно использовать его. Здесь выбор может быть продиктован разными обстоятельствами, в частности, стремлением к унификации элементной базы, стоимостью компонент, опытом работы. Остановимся пока на операционном усилителе.

Хотя об операционном усилителе я выше упоминал достаточно часто, я мало говорил о том, как с ним работать в программе Qucs или аналогичных программах. Как правило, эти программы создаются зарубежными авторами, которые используют модели элементов, производимых в их стране. Сегодня любителю доступны эти компоненты, и если у него возникают сомнения, он может использовать именно те компоненты, которые имеются в наличии в программе. Кроме того, есть справочники по элементной базе, в которых можно найти варианты замены зарубежных операционных усилителей, как и других элементов, на отечественные. И, наконец, большая часть экспериментов, которые приводятся в книге, предназначена для получения качественных, а не количественных, результатов. Что не исключает и возможности получения количественных результатов, но в этом случае нужно проводить не только моделирование схемы за компьютером, но обязательное повторение на макетной плате. Сравнение результатов даст тот опыт применения компьютерного моделирования, который поможет определять в каждом конкретном случае, требуется ли макетирование схемы. Только долговременный опыт работы с конкретной программой может дать все ответы на все вопросы.

Но вернемся к схеме обогревателя.

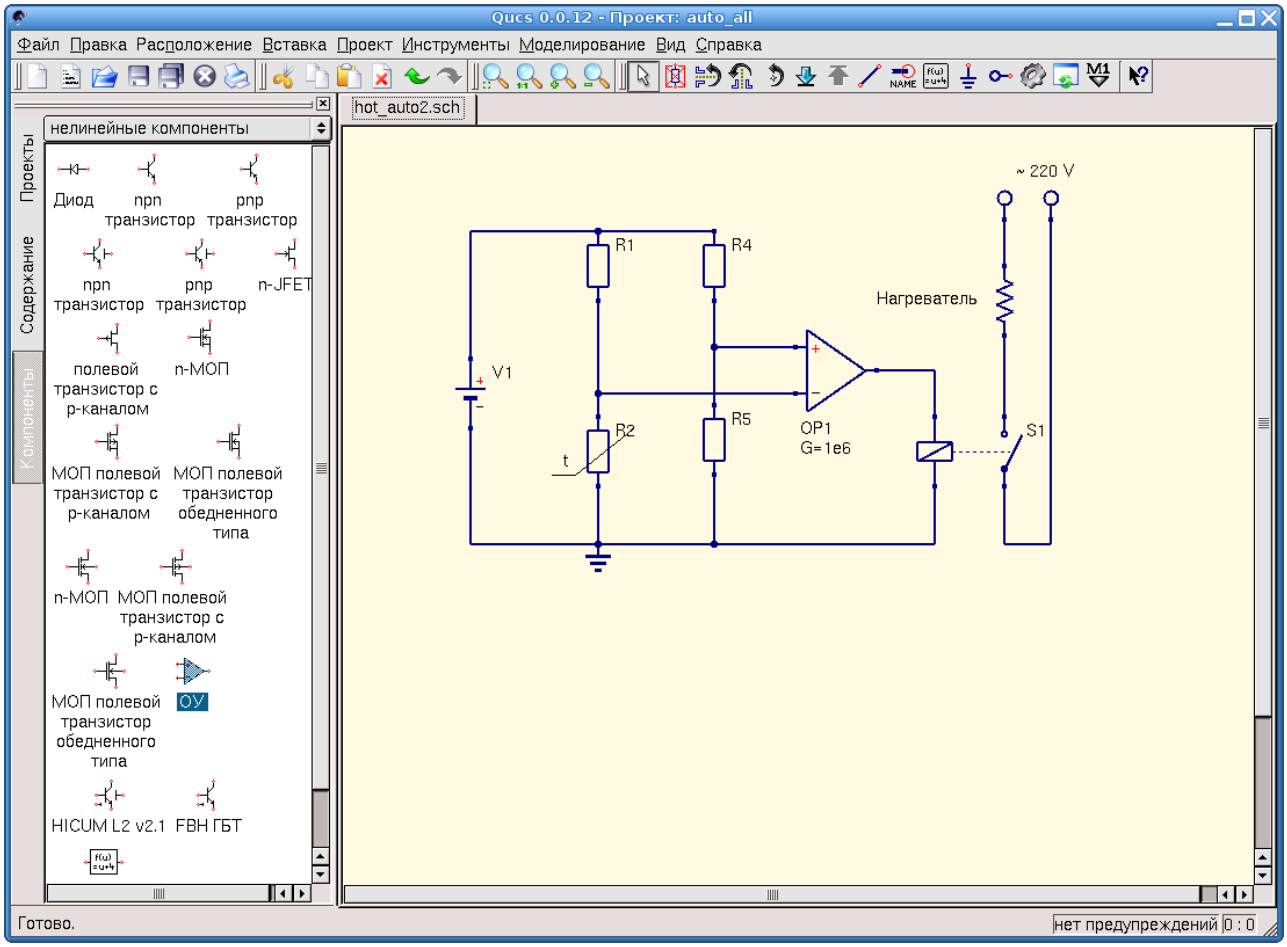


Рис. 12.2. Применение операционного усилителя в автомате-нагревателе

Операционный усилитель, включенный в мост, образованный двумя делителями напряжения R1R2 и R3R4, дает нам избавление от нескольких проблем. Подбором резисторов мы можем задавать нужные температурные значения, при которых происходит срабатывание. Даже небольшие изменения сопротивления датчика, благодаря большому коэффициенту усиления напряжения, могут использоваться для управления реле. Меняя подключение входов мы можем инвертировать работу реле по отношению к направлению изменения сопротивления. И начальное значение сопротивления датчика повлияет только на величину сопротивлений моста.

Какие еще преимущества дает применение усилителя? Большую универсальность (или гибкость решения). В данном случае мы «строим» обогреватель-автомат. Но если в качестве датчика мы используем не терморезистор, а фоторезистор – элемент, сопротивление которого зависит от освещенности, то, не меняя схемы, я даже не исключаю, что сопротивления моста могут сохранить свои значения, мы можем включать контактами реле не нагреватель, а светильник. В этом случае мы получим автоматическое управление освещением. Применив датчик, меняющий сопротивление при возросшей концентрации углекислого газа, мы получим автоматическое управление вентиляцией.

Проведем два эксперимента, которые позволят оценить возможности полученного решения.

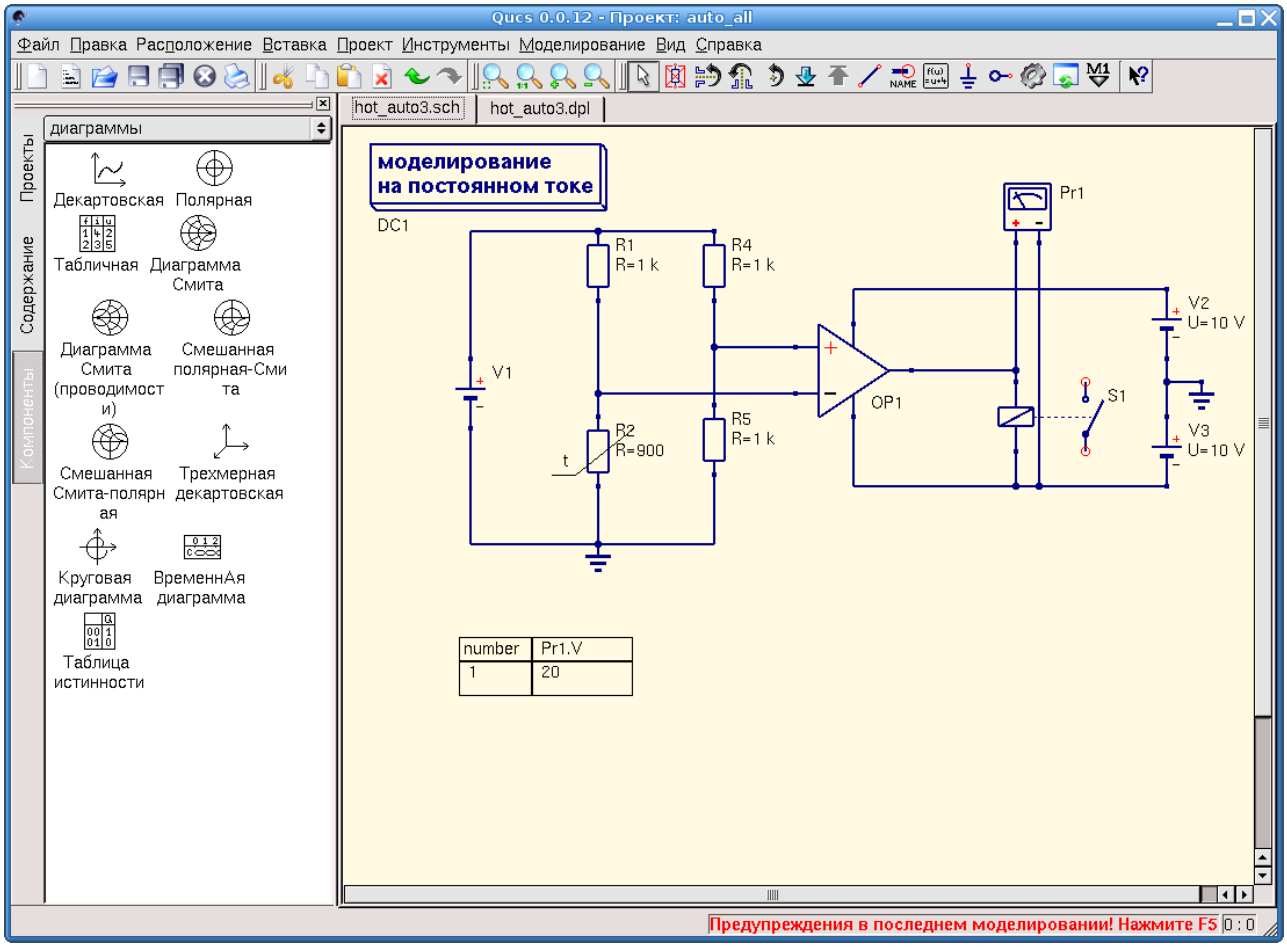


Рис. 12.3. Первый эксперимент со схемой автомата

Напряжение на реле 20 В. Изменим значение терморезистора с 900 Ом на 1.1 кОм.

number	Pr1.V
1	1.46

Рис. 12.4. Выходные значения при изменении температуры

Меняя значения резистора R2, не 900 Ом, а 950 Ом или 955 Ом, мы можем получить представление о чувствительности схемы, и сможем выбрать терморезистор, для которого в справочнике приводится параметр температурного коэффициента сопротивления.

Схема получилась достаточно универсальная, но решили ли мы все задачи? Не уверен.

За счет гистерезиса самого реле, разницы между напряжениями включения и выключения, мы могли бы получить некоторую устойчивость, но операционный усилитель нивелирует этот эффект. А проблема может выглядеть так: при достижении заданной температуры напряжение на датчике может колебаться около точки перевала, что заставит реле «вибрировать» вместо четкого включения, которое может и не произойти. Для температуры это может быть связано с движением воздуха, для других датчиков с колебаниями других параметров около точного значения.

Именно по этой причине лучше задаваться двумя температурами – включения и

выключения. Но как это реализовать?

Частично «погасить» проблему можно добавлением на выход операционного усилителя RC-цепи. Резистор включается последовательно с реле, а конденсатор параллельно реле. Для эксперимента мы изменим схему, заменив операционный усилитель генератором одиночного импульса. Генератор «смоделирует» поведение операционного усилителя при достижении заданной температуры с возможным «откатом» к исходному положению.

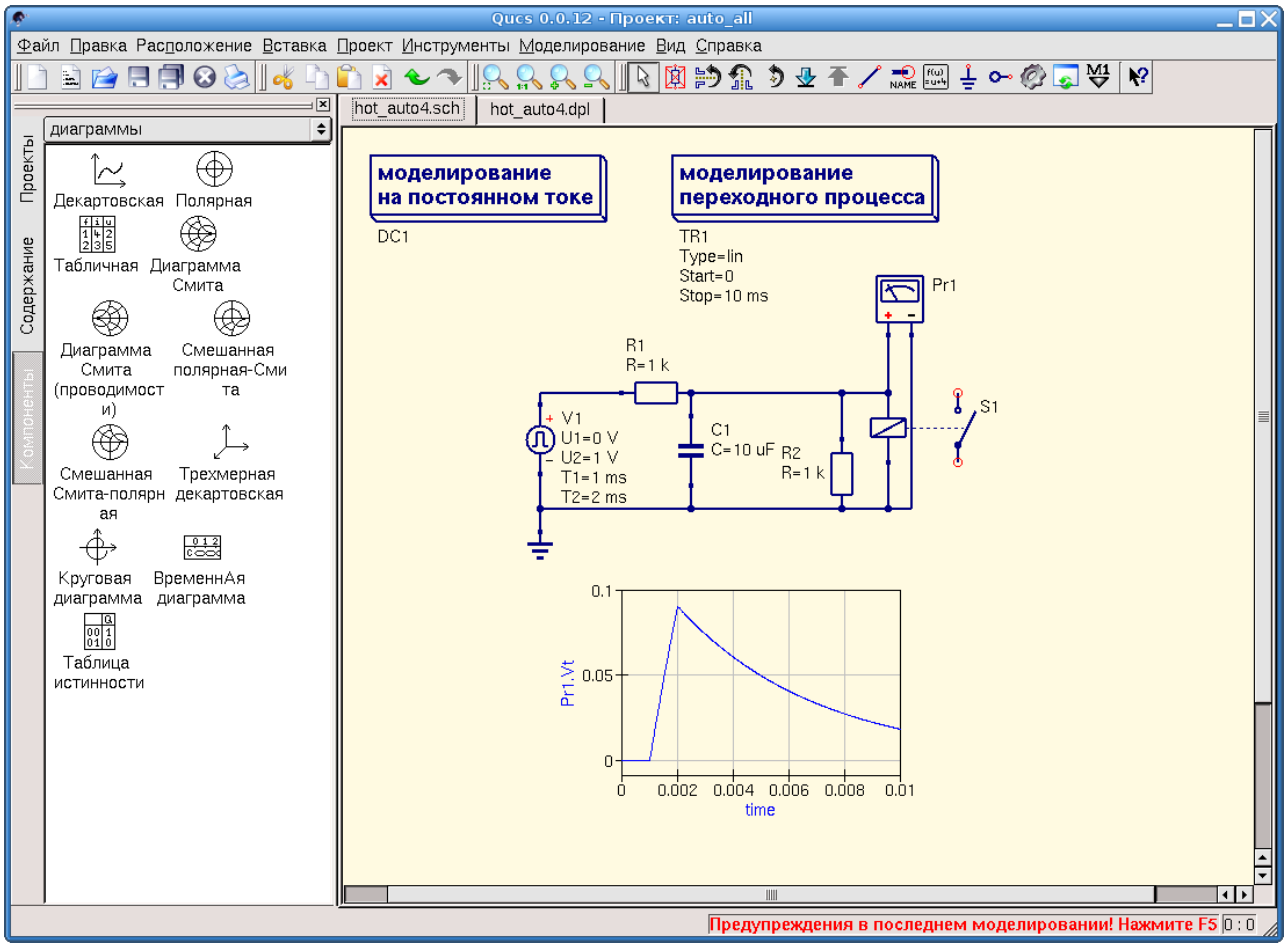


Рис. 12.5. Эксперимент с «затягиванием» переходов

Собственно, мы добавили фильтр на выходе усилителя, который «обрезает» высокие частоты (быстрые колебания напряжения) и пропускает низкие частоты (редкие перемены состояния). Резистор R2 – это сопротивление обмотки реле. Как видно из диаграммы, напряжение на реле возрастает не стразу, включение произойдет через некоторое время. И выключение (добавьте гистерезис реле) тоже.

Дальнейшее совершенствование схемы, направленное на стабилизацию ее работы, потребует некоторых изменений. Думаю, вы сами увидите разницу:

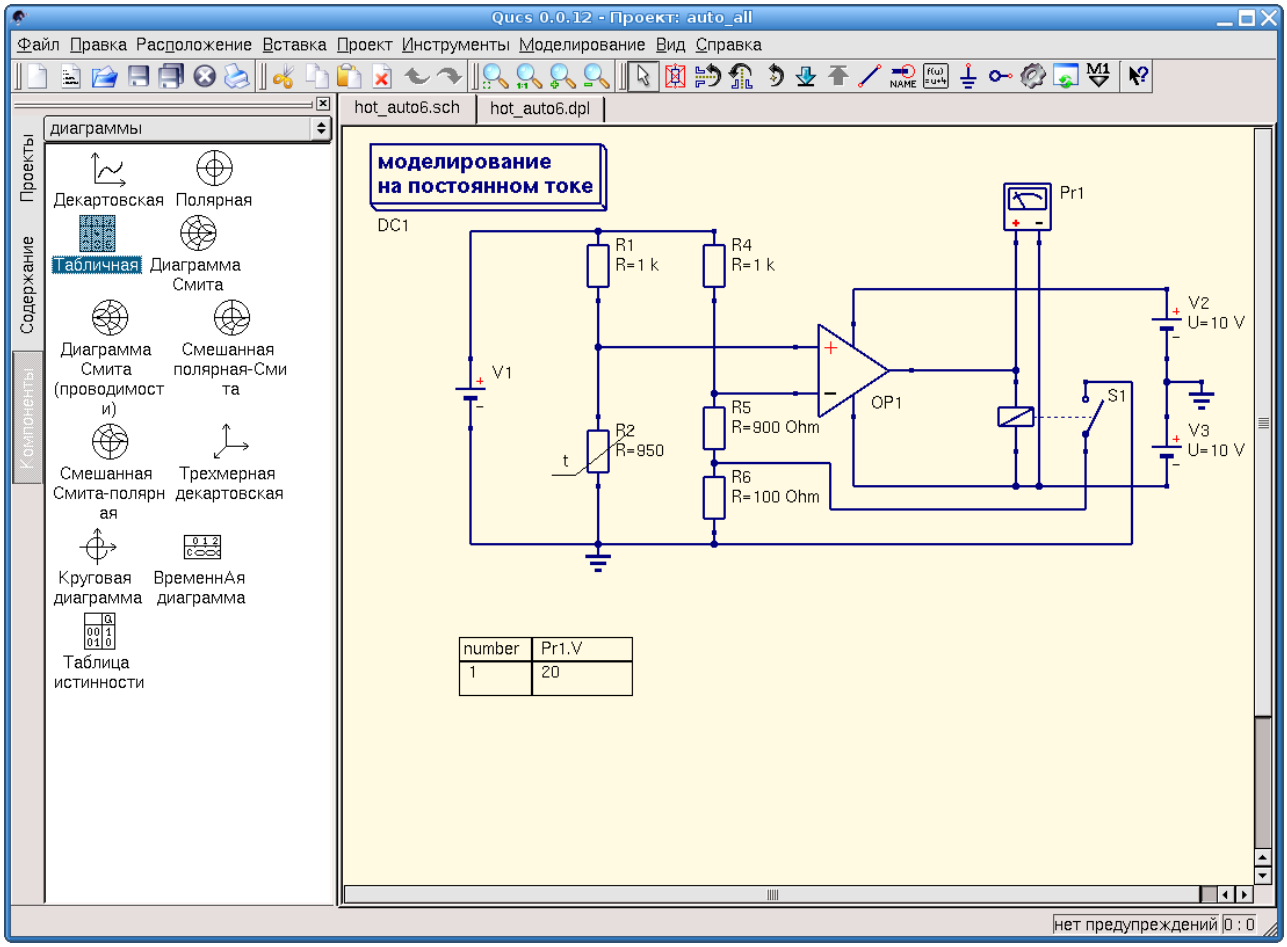


Рис. 12.6. Продолжение работы над схемой автомата-обогревателя

Большинство терморезисторов имеют отрицательный ТКС, то есть, их сопротивление уменьшается при увеличении температуры, поэтому входы операционного усилителя поменялись местами. Фильтр, показанный выше, я из схемы эксперимента убрал – его работа в данный момент не будет видна, а схему он загромоздит.

Я решил использовать дополнительную (нормально разомкнутую) пару контактов реле для достижения цели. Один из резисторов «разбит на две части», при включении реле его контакты замыкают нижнюю часть этого резистора, R6. В первоначальном варианте переключения происходили при значениях резистора R2 равных 995 и 1005 Ом. Это можно проверить еще раз, удалив соединение с контактами реле.

number	Pr1.V
1	1.54

Напряжение при 995 Ом

number	Pr1.V
1	20

Напряжение при 1005 Ом

Рис. 12.7. Значения величины резистора при переключении первоначальной схемы

Теперь, это видно на рисунке 12.6, при сопротивлении 950 Ом (более высокая температура) напряжение на выходе схемы все еще остается высоким, поддерживая реле включенным.

Хотя это простое устройство, но его простота продержится ровно до того момента, как попытаешься его реализовать.

Дальнейшее усложнение схемы может улучшить ее, в части надежности, точности, в части гибкости или функциональности, но это можно сделать многими способами, в частности применить другой подход. К этому мы, возможно, вернемся, а сейчас я предлагаю решить другую задачу – создать автомат для «рассеянных».

Автоматическая нянька для рассеянных

Собственно, не только для забывчивых людей, но для удобства даже людей собранных. А, если честно, то несколько примеров, включая этот, мне нужны лишь для того, чтобы показать, насколько проще все можно решить, используя современную элементную базу. Это относится и предыдущей задаче, и к проекту «Светофор», и, как мне кажется, ко многим случаям, с которыми вы можете столкнуться в своей практике. А пока о рассеянных.

Каждый вечер ваша семья собирается дома. Приходите вы, следом приходит с работы ваша жена, возвращаются ваши родители, а первым, возможно, приходит из школы ребенок.

Каждое утро ваша семья уходит из дома. Уходите ли последним вы, уходит ли ваш ребенок, всегда желательно, чтобы все электрические приборы были выключены, еще лучше, если будет перекрыта вода. Достаточно утомительно проделывать это каждый день. А если в вашей семье есть человек безответственный, вроде меня, который вспоминает о том, что надо проверить свет и воду только тогда, когда собирается с работы домой...

Хорошо бы, если бы: при входе в дом автомат подсчитывал, сколько человек вошло; а при выходе из дома подсчитывал, сколько вышло; и если вышли все, кто вошел, то выключал бы электричество. Есть электрические вентили, которые открывают воду при подаче на них напряжения, и перекрывают воду, когда электричество выключено. Вот бы это все собрать!

Попробуем.

Первое, с чем следует определиться, а выбирать будем из того, что уже знаем, так это с датчиком. Я не знаю датчика «подсчета посетителей». В голову приходит простой вариант – фиксировать открывание входной двери. Установить на дверь контактный датчик не сложно, он позволит подсчитывать, сколько раз дверь открывалась.

Возражения против такого датчика: открытая дверь не означает, что кто-то вошел и вышел, открывание двери не определяет, открыта она для входа или для выхода. Существенно.

Некогда я встречал решение в виде двух лучей, приходящих на два фотоэлемента. При пересечении лучей есть возможность определить наличие и направление движения. Но, посмотрев на входную дверь, я понял, что в домашних условиях трудно приспособить подобное устройство без существенных переделок в доме. А, с другой стороны, если поставить такое устройство внутри помещения, дополнить его контактным датчиком на двери, то не получим ли мы требуемый датчик «подсчета посетителей»?

Рассмотрим ситуацию прихода. Срабатывает датчик двери, затем срабатывает фотодатчик. Фиксируя эту последовательность, мы можем точно сказать, что кто-то вошел. Срабатывает фотодатчик, затем дверной датчик, значит, кто-то вышел. Сомнительная ситуация: я подхожу к входной двери, чтобы открыть гостю. Вначале срабатывает фотодатчик, затем дверной датчик, затем (я отошел от двери, приветствуя гостя) фотодатчик, затем фотодатчик срабатывает повторно (гость вошел в дом), затем срабатывает дверной датчик (гость захлопнул дверь), затем срабатывает фотодатчик (теперь уже я закрываю дверь на замок). Ситуация может осложниться еще больше, если гость только подошел к двери, но не вошел в

дом (сосед спросил, когда я уберу свою машину с его места парковки возле дома).

Такая «возня» у входной двери... не знаю, можно ли с этим разобраться... Идем ко входной двери, и сразу становится ясно, что прежде чем разбираться с запутанными ситуациями, следует разобраться с основными: стена, а значит и дверной проем (именно там можно установить фотоприемник и светоизлучатель), настолько тонкая, что я не пересекаю луч света, когда открываю дверь.

Вот так всегда! Сидя за компьютером или за письменным столом, все прекрасно придумываешь, обходишь все трудности, принимаешь решение, а выходишь в реальную жизнь, и все летит «под откос». Если вы думаете иначе, то подойдите к своей двери.

Есть вариант, который мне не нравится. У входной двери лежит коврик, на который наступаешь и когда выходишь, и когда входишь в дом. Снабдив его датчиком давления (???) можно обойти проблему, но придумывать такой датчик совсем нет желания, а покупка готового, большинство датчиков, о которых я сейчас думаю, пришли из систем охраны, которые можно назвать автоматизированными системами защиты от вторжения, покупка такого датчика может оказаться слишком дорогостоящей.

Есть еще одно решение, и оно тоже мне не слишком нравится. Если фотоприемник установить в зоне дверной ручки, там, где оказывается рука, когда открываешь дверь изнутри, то рукой фотоприемник будет «затеняться», подавая нужный сигнал. Не самое изящное решение. В качестве второго датчика можно использовать контактный датчик (герконовый датчик) на входной двери.

И, наконец, есть частное решение, на котором я, пожалуй, остановлюсь. Входные двери в моей квартире устроены так, что есть небольшой тамбур шириной в дверной проем, который разделяет наружную и внутреннюю входные двери. То есть, в наличии две двери, одна из которых открывается изнутри, а другая снаружи. Снабдив их дешевыми герконовыми датчиками, я могу решить проблему с направлением движения: если первым срабатывает датчик внутренней двери, а затем наружной, то это «уход из дома», иначе «приход домой». Мало того, разделяя события на «дверь открыта» и «дверь закрыта», я, как мне кажется, смогу решить проблему с «возней около входной двери». Посмотрим, какая последовательность срабатывания герконовых датчиков получается в трех случаях: приход домой, уход из дома, сосед зашел поздороваться.

Приход домой:

1. Размыкается датчик наружной двери.
2. Размыкается датчик внутренней двери.
3. Замыкается датчик наружной двери.
4. Замыкается датчик внутренней двери.

Уход из дома:

1. Размыкается датчик внутренней двери.
2. Размыкается датчик наружной двери.
3. Замыкается датчик внутренней двери.
4. Замыкается датчик наружной двери.

Зашел сосед поздороваться:

1. Размыкается датчик внутренней двери.
2. Размыкается датчик наружной двери (далее, милая беседа с соседом).
3. Замыкается датчик наружной двери.
4. Замыкается датчик внутренней двери.

Первая половина событий в последнем случае соответствует уходу, но вторая половина приходу домой, что, в принципе, позволит различить последовательность событий. Пожалуй, именно этот вариант устройства датчиков я возьму за основу. Но, поскольку это частное решение, прежде я немного времени уделю тому, как создать датчик из оптопары, элементы которой можно извлечь из старой «мышки», например, или, что лучше, использовать в качестве фотоприемника элемент с названием TSOP. Это микросхема с фотоприемником и фильтром, настроенным на одну из стандартных частот, наиболее часто используемых в ИК-пультах управления: TSOP1730 – 30 кГц, TSOP1736 – 36 кГц и т.д.

Для излучателя можно использовать светодиоды ИК диапазона, или попробовать красные светодиоды серии АЛ307, например. В первом случае можно получить лучшие результаты, а во втором, получить лучший контроль – когда светодиод включен, это видно. Сами по себе эксперименты с такой парой достаточно интересны. Разберем первый, скажем, вводный эксперимент.

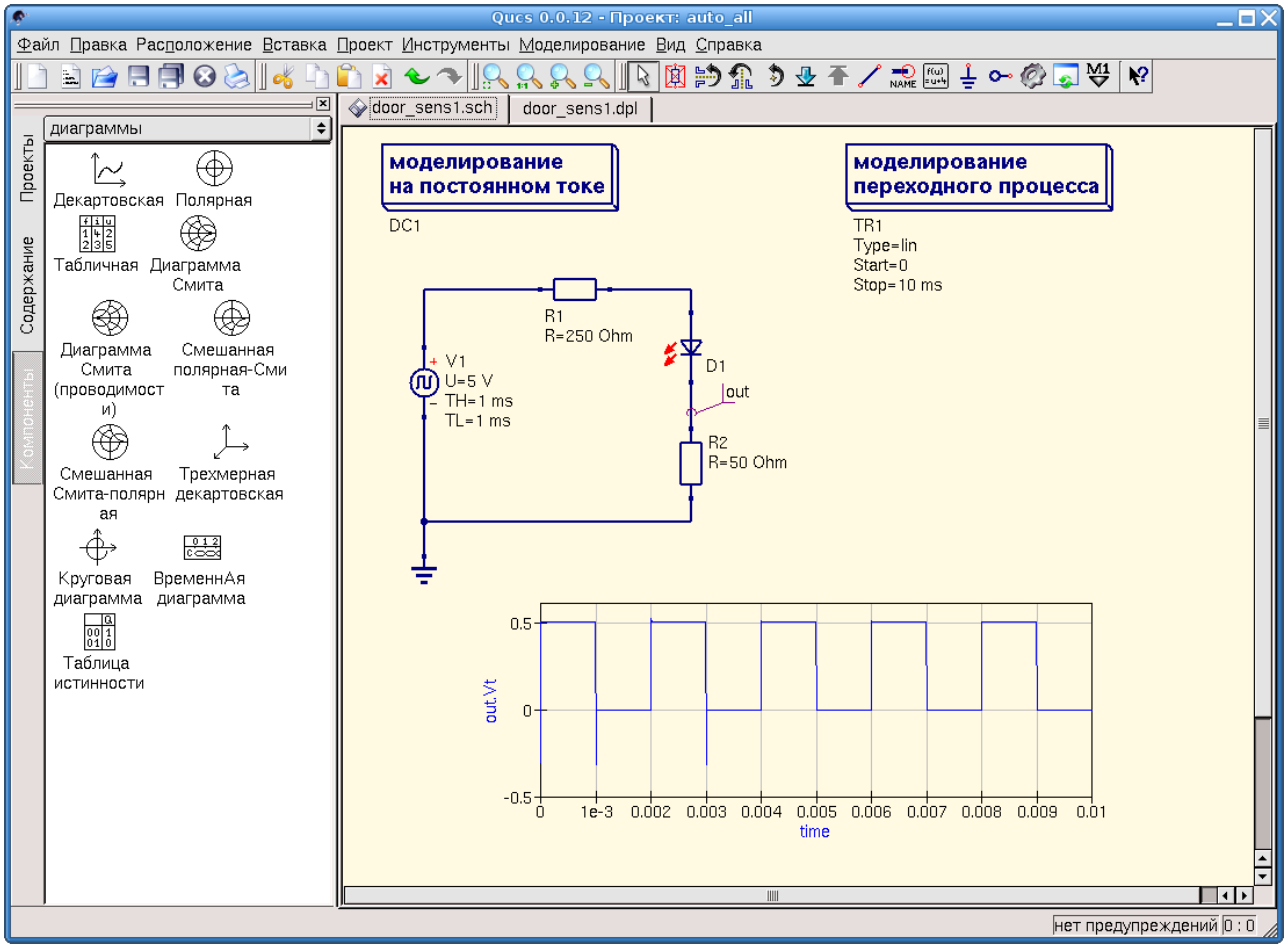


Рис. 12.8. Вводный эксперимент со светодиодом

Диаграмма показывает, как изменяется ток через светодиод. Поскольку в качестве генератора выбран источник прямоугольных импульсов, ток пульсирует с тем же периодом. Если теперь (на схеме не изображено) с помощью фотоприемника (обычного, а не TSOP) и осциллографа посмотреть на световые импульсы, то они будут похожи на вид, полученный средствами программы Qucs, хотя, видимо, будут не столь «красиво правильны». Светодиод, кстати, я взял из «Библиотеки компонентов», раздел меню *Инструменты*. Соотнести допустимый и полученный ток, я полагаю вы знаете как, полученный — средний за период.

Можно, раздвигая излучатель и приемник, оценить расстояние, на котором все будет успешно работать. Думаю, это расстояние окажется не больше полуметра. Теперь попробуем изменить схему, добавив всего один элемент.

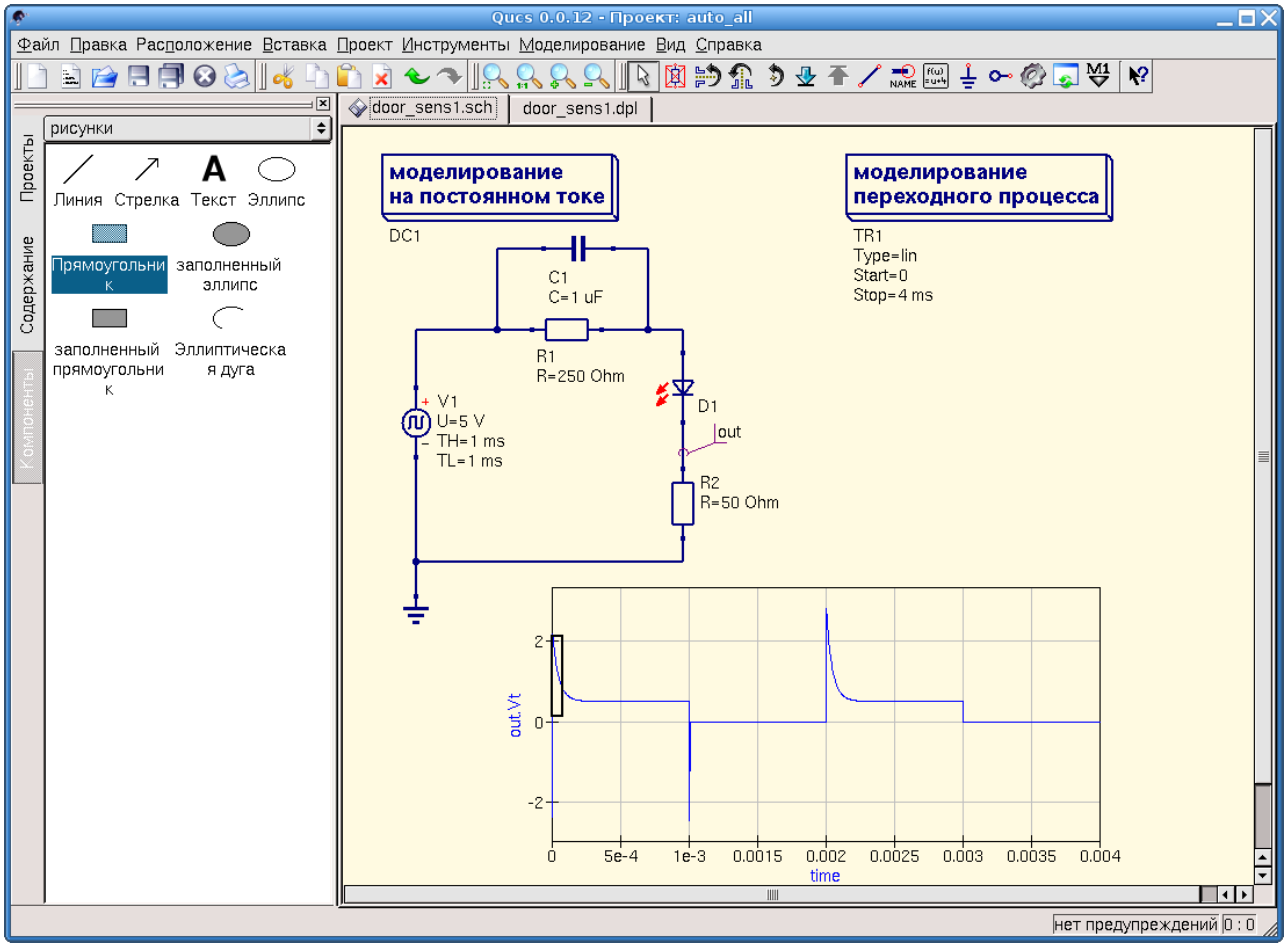


Рис. 12.9. Развитие вводного эксперимента

Если при таком включении вы постараетесь определить «живое» расстояние срабатывания, оно заметно возрастет. Происходит это по причине кратковременного увеличения тока через излучающий светодиод. Если в первом случае падение напряжения на резисторе R2 было 0.5 В, то во втором, выбросы напряжения достигают 2 В, то есть, ток увеличивается в 4 раза. Осталось привести параметры импульса в соответствие с допустимыми параметрами светодиода. Здесь можно использовать приближенную оценку, проведя (на диаграмме еще присутствует дополнительный ток, который больше, чем нужно из-за величин сопротивлений) прямоугольник подобный тому, что есть на рисунке, и считая величину импульса тока по длительности равной половине основания прямоугольника. Если датчик устанавливать в дверном проеме, то можно довериться положительным результатам при срабатывании фотоприемника на расстоянии в 1 м от излучателя. Впрочем, некоторый запас не помешает.

Отличие в работе обычного фотоприемника (из фотодиода) и микросхемы TSOP в том, что последняя будет поддерживать постоянное напряжение на своем выходе до тех пор, пока получает импульсы с частотой несущей светового сигнала. Пропадают импульсы, сигнал на выходе микросхемы принимает второе «логическое» состояние. Обычный же фотоприемник, а это может быть и фоторезистор, и фототранзистор, будет «честно» пытаться повторить те импульсы, которые он «видит». Соответственно следует устраивать обработку принимаемого

сигнала. В первом случае проще, есть один уровень – состояние ожидания, есть другой – событие. Во втором случае можно использовать детектор, как в радиоприемнике, предварительно усилив сигнал с помощью предварительного усилителя. Если вы в качестве фотоприемника используете фотодиод, напомню, что многие светодиоды могут работать в качестве фотодиодов, то включение светодиода можно произвести, как показано ниже.

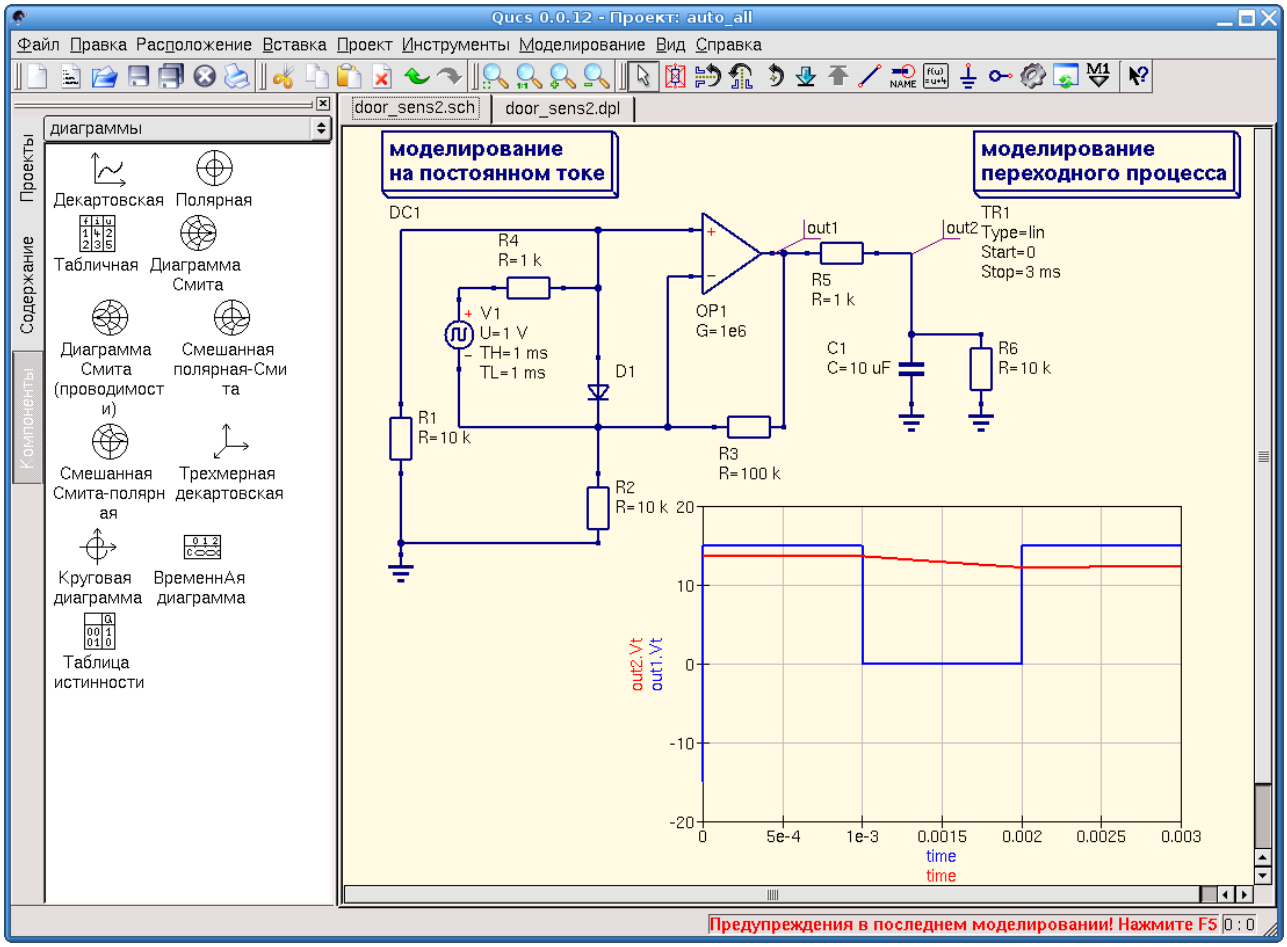


Рис. 12.10. Возможный вариант включения фотодиода

Диод D1 на схеме изображает фотодиод, а источник прямоугольных импульсов V1 «изображает» световые импульсы от излучателя, отнесенного на нужное расстояние. Детектор «как у радиоприемника» у меня получился без диода, который должен был бы стоять последовательно с резистором R5. Резистор этот, как вы догадались, ограничивает выходной ток операционного усилителя, и зачем это нужно, вам тоже, думаю, понятно. А диод я убрал, чтобы облегчить режим симуляции в программе Qucs. Но! Если на выходе операционного усилителя вы получите однополярный сигнал, как на диаграмме, то есть, прямоугольные импульсы от 0 до +15 В (напряжения положительного питания), то отчего бы не отказаться от диода? Или я неправ?

Если все получится, как получилось на рисунке, то в режиме ожидания пара светодиод и фотодиод будут формировать напряжение на резисторе R6 (нагрузка, реле, вход следующего каскада и т.п.), а когда фотоприемник укрыт рукой от излучателя, напряжение станет равно нулю. Не мгновенно, но это произойдет. Как быстро это случится, сколько времени пройдет от момента перекрывания луча до срабатывания, с этим попробуйте поэкспериментировать отдельно.

Будем считать, что с датчиками мы разобрались. На выход схемы рисунка 12.10 мы

включим реле, а его контакты будут передавать нам события, происходящие возле дверей. Конечно, реле следует выбирать из тех, что будут успешно работать (по току и напряжению) на выходе операционного усилителя. Или можно добавить на выход операционного усилителя транзистор, в коллекторную цепь которого можно включить реле.

Теперь пора перейти к построению схемы, которая может «обслужить» логику событий. Я не сомневаюсь, как во всех подобных случаях, что самый удобный вариант – это применение микроконтроллера. Но мы с ним не знакомы, увы. Будем «мудрить» с тем, с чем знакомы.

По первому порыву я хочу использовать RS-триггер, то есть, триггер, у которого два входа: S для установки выхода в активное состояние, и R для перевода его в пассивное состояние. Кроме прямого выхода у этого триггера есть инверсный выход, «на котором все происходит наоборот». Сигналы от двух датчиков должны прийти на эти входы, скажем, от внешней двери на вход S, мы «приходим», а от внутренней двери на вход R, мы «уходим». Помимо этой очевидной идеи я хочу использовать импульсы для управления входами, а не потенциалы. В целом триггер можно будет назвать «детектором направления движения». Для получения импульсов пригодится такая схема.

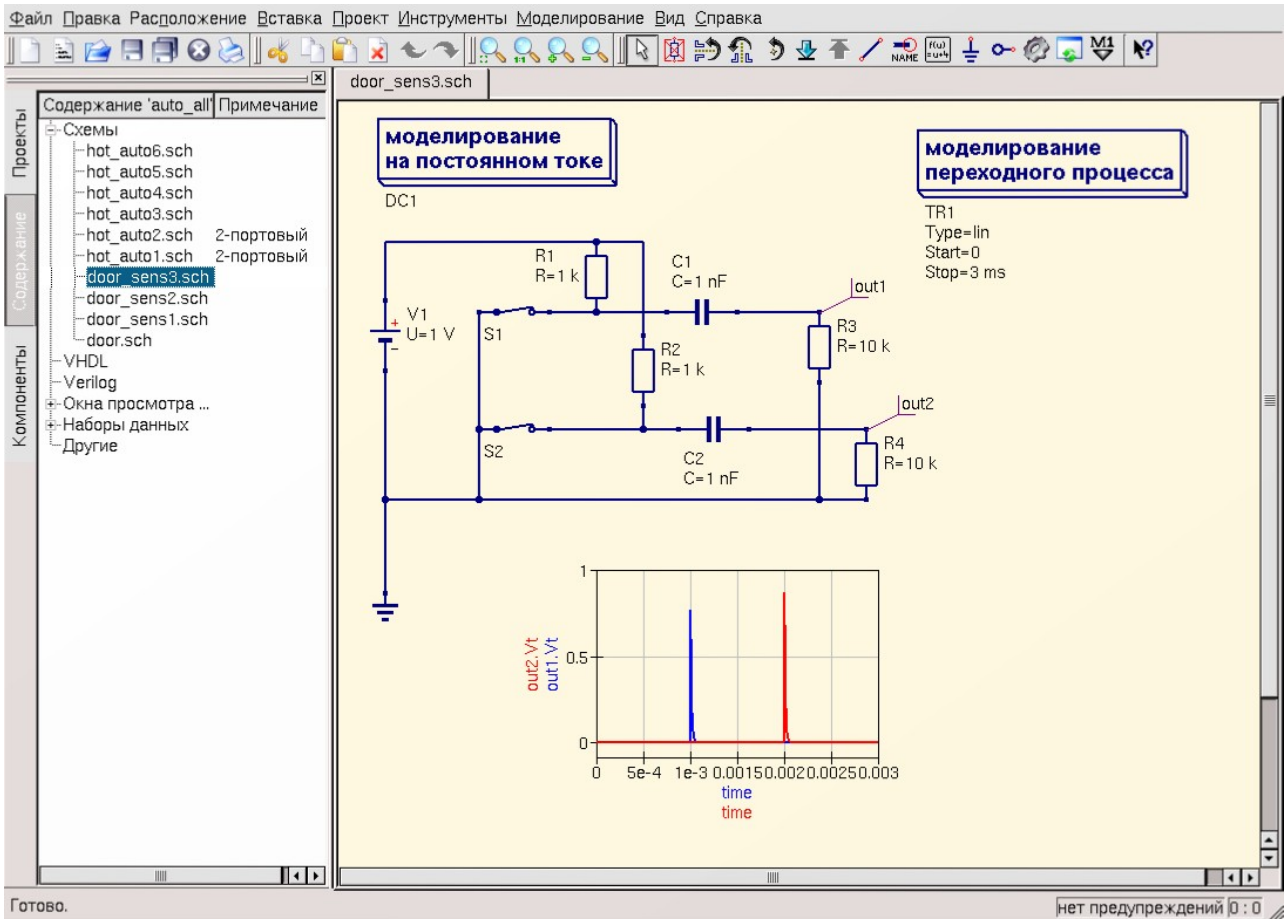


Рис. 12.11. Получение управляющих импульсов от датчиков

Два импульса на резисторах R3 и R4 (*out1* и *out2*) должны будут управлять RS-триггером. Так я предполагаю. Как получаются эти импульсы, вы уже разобрались – это импульсы тока, протекающие по резисторам в моменты, когда размыкаются контакты S1 и S2, за счет токов, протекающих через конденсаторы в переходных процессах. Когда контакты замкнутся, через резисторы R3 и R4 будут протекать обратные токи, формируя импульсы напряжения отрицательной полярности, от которых мы постараемся избавиться. А сейчас, предположив,

что с задачей формирования импульсов мы справились, разберемся с самим триггером.

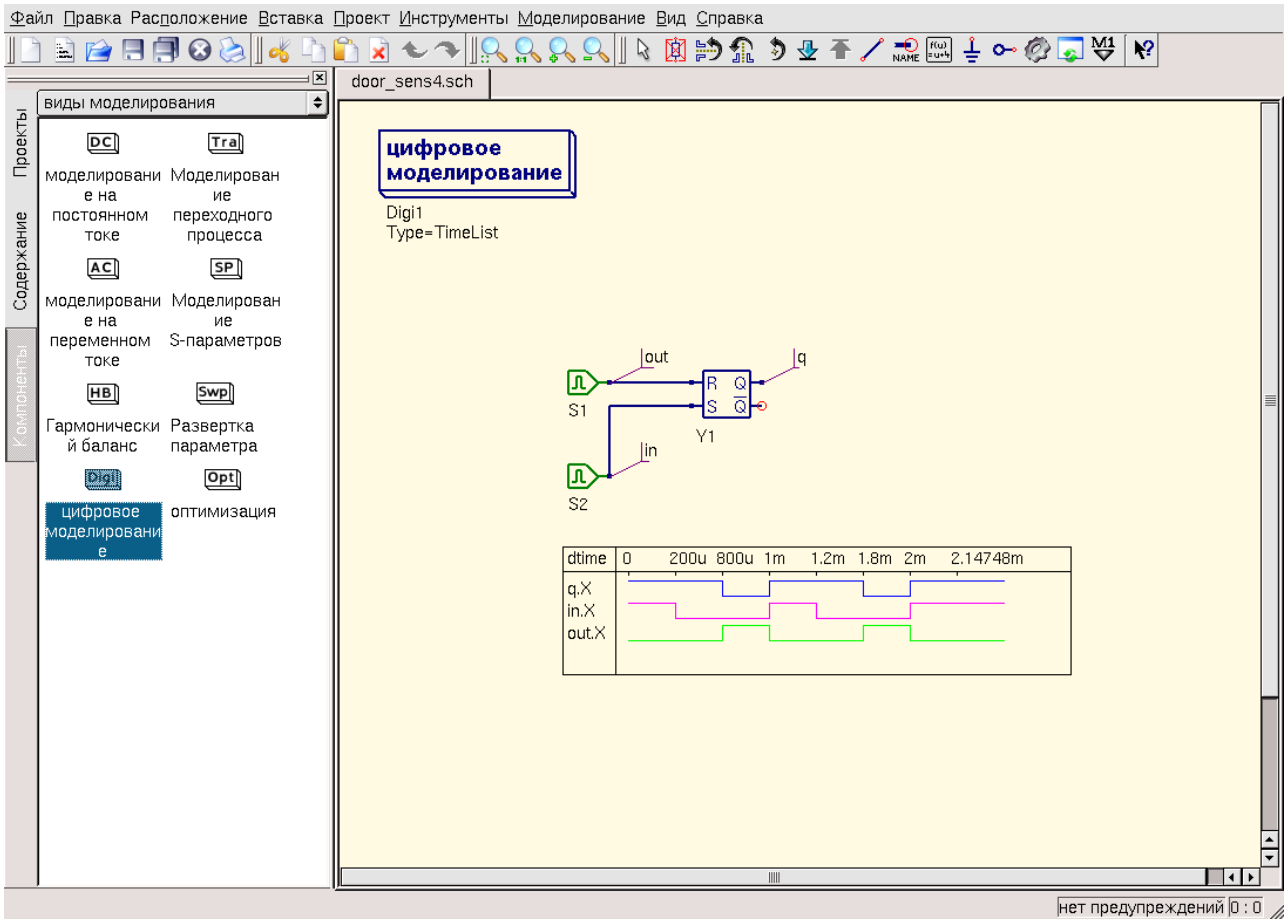


Рис. 12.12. RS-триггер, как детектор направления движения

В этом месте у меня возникло ощущение, что «детектор направления» обнаружил, что я двигаюсь не совсем «в ту сторону». Давайте разберемся. Что было задумано?

Сигналы от двух датчиков должны прийти на эти входы, скажем, от внешней двери на вход S, мы «приходим», а от внутренней двери на вход R, мы «уходим».

Как детектор, триггер должен принимать активное состояние на выходе Q, если «мы пришли», и иметь низкий уровень, если «мы ушли». Что получилось?

Рассмотрим диаграмму до момента времени, совпадающего с отметкой 1 мС (1m). Выход триггера изначально на высоком уровне (положим, кто-то приходил). Первым срабатывает датчик внешней двери, обозначенный на схеме меткой *in*, и остается разомкнут до временной отметки 200 мкС (200u), затем он замыкается – дверь закрылась – и на отметке 800 мкС размыкается датчик внутренней двери (на схеме метка *out*). Выход триггера, сигнал q.X на диаграмме, переходит в состояние с низким уровнем. А это, кажется, означает, что кто-то вышел! Вышел?.. Я все сделал наоборот. Хотел-то я сделать правильно, согласно логике моих рассуждений, что датчик наружной двери дает сигнал «кто-то входит», но триггер фиксирует свое состояние по последнему полученному им сигналу, а последим был сигнал от внутренней двери, а не от наружной. В этом нет ничего страшного – поменяю сигналы. Не было бы ничего страшного, даже если бы все было установлено на дверях, достаточно поменять местами провода от датчиков. Но даже на таком простом примере становится ясно, как легко можно ошибиться, а в более сложном устройстве выявить подобную ошибку было бы гораздо сложнее.

Я переобозначу датчики, как *in_door* и *out_door* (датчик внутренней и внешней двери). Теперь, если провести моделирование, то можно убедиться, при последовательности срабатывания датчиков внешний-внутренний триггер имеет высокий уровень на выходе, а при последовательности внутренний-внешний, низкий уровень. Таким образом, проверив состояние выхода триггера, мы можем сказать, каким было последнее событие: приход или уход. Уф!

Мы разобрались, как построить и установить датчики, поняли, как определить направление движения по состоянию RS-триггера. Теперь займемся счетом. Кому, как ни счетчику этим заниматься? Есть реверсивный счетчик, который при подаче импульсов на один из входов увеличивает число, отображаемое на его выходах, а при подаче импульсов на другой вход уменьшает это число. Когда на его выходах будет отображаться нуль, ушли из дома все, счетчик через контактор (реле) будет выключать электричество в доме, обесточив электрические вентили, которые при этом перекроют воду. Вот так. А то, уходя из дома, вечно ломаешь голову – а свет выключил? А на кухне? А в ванной? А воду закрыл?...

Гораздо приятнее мечтать о том, что ты сделаешь, чем делать то, о чем мечтаешь.

С формирователем импульсов разберемся попозже, а сейчас посмотрим, как можно соединить детектор направления и счетчик. В программе Qucs нет счетчика, поэтому я сделаю маленькие счетчики из D-триггеров в количестве двух штук (для двух направлений счета).

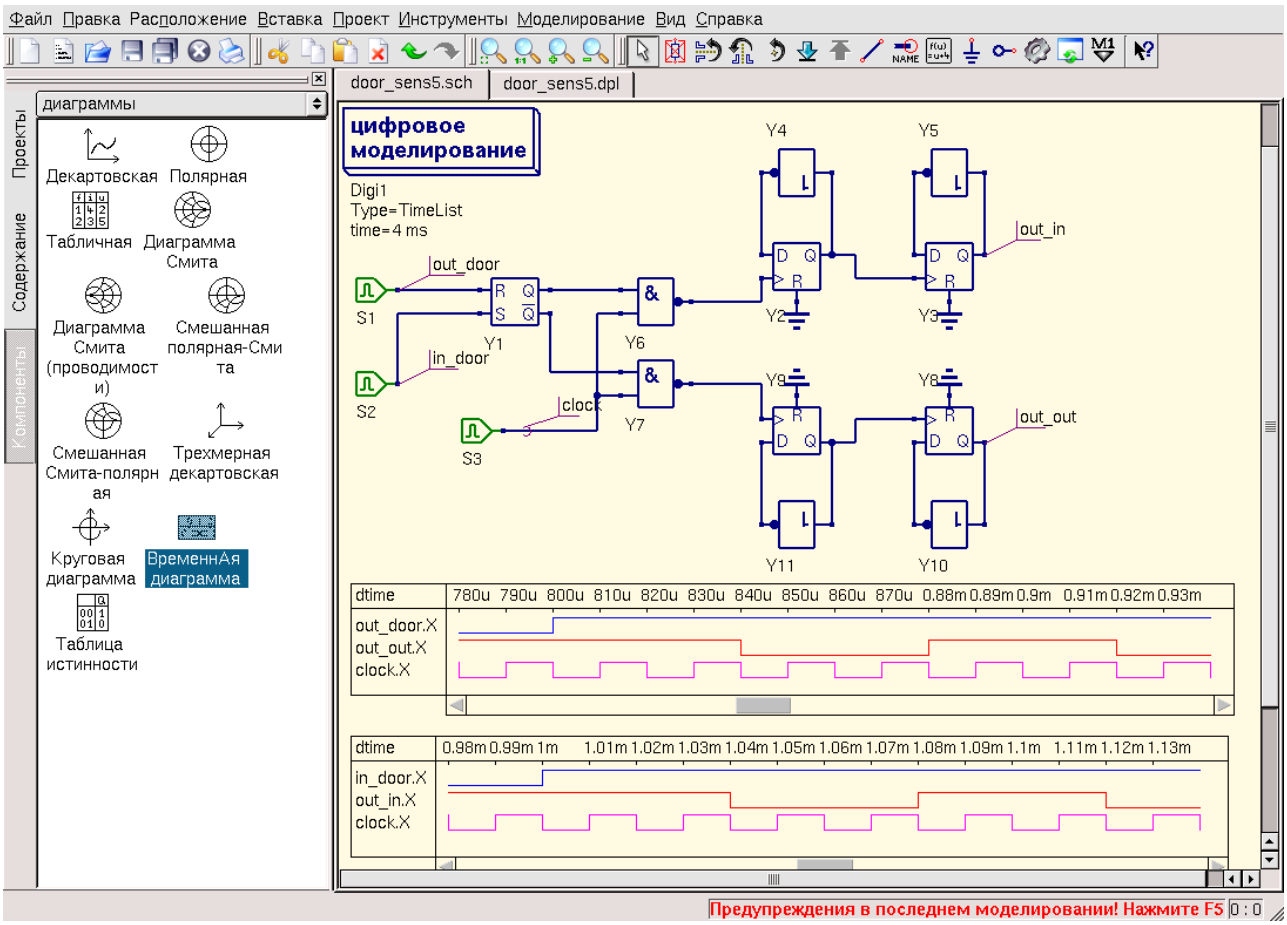


Рис. 12.13. Распределение счета после детектирования

Генератор импульсов счета с меткой *clock* иллюстрирует формирование направления счета

в счетчике. На верхней диаграмме видно, как счетчик начинает отсчет после срабатывания датчика *out_door*, на нижней, после срабатывания датчика *in_door*. Конечно, нам не нужно более одного импульса счета. Но эксперимент на данном этапе остается полезен, поскольку после него нам остается решить вопрос с формированием импульса счета, после чего можно собирать всю схему.

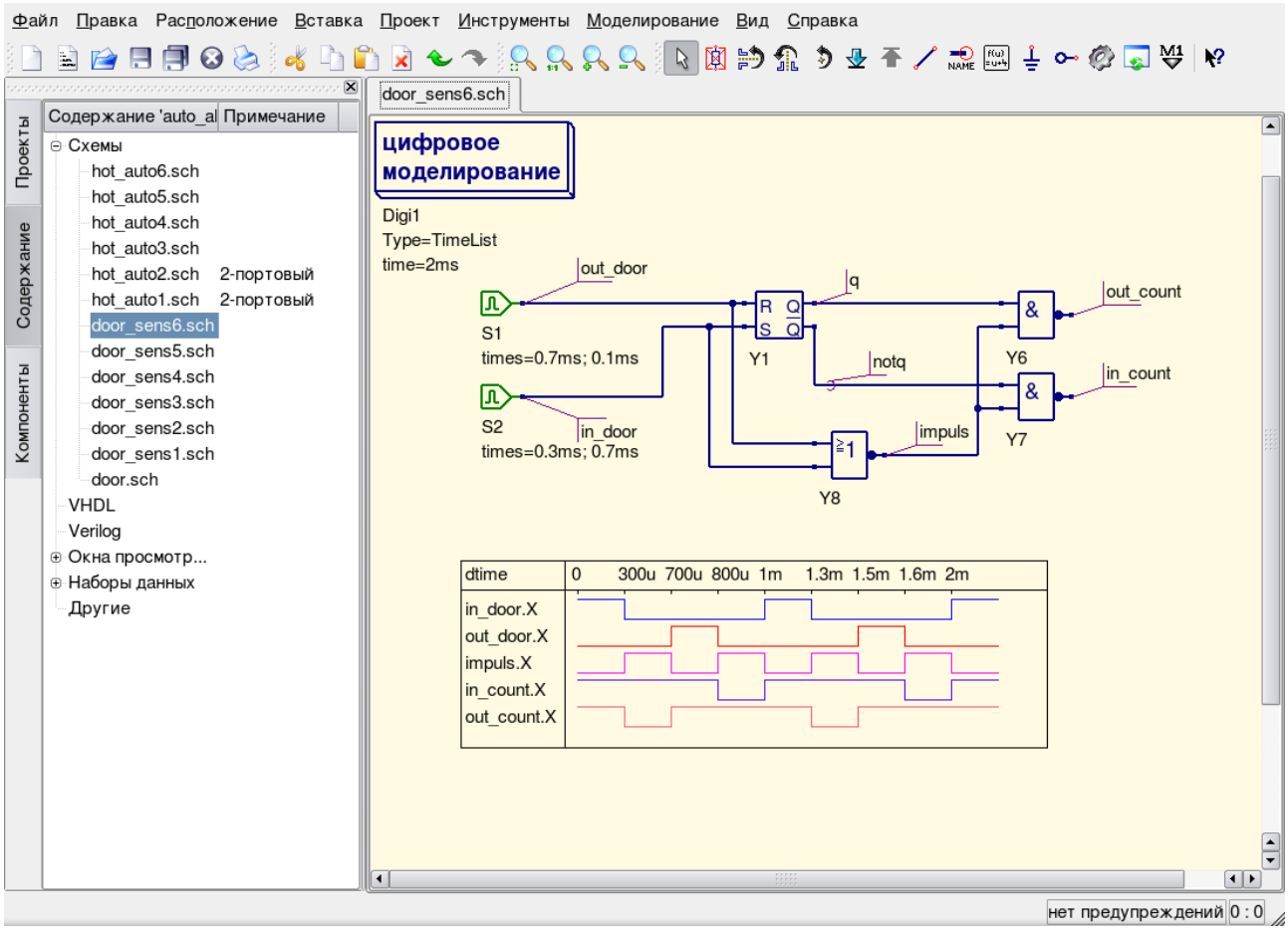


Рис. 12.14. Формирование импульса счета

Для формирования импульса счета используется момент, когда обе двери закрыты, а оба датчика замкнуты. Двух-входовая схема ИЛИ-НЕ меняет свое состояние, а сформировать короткий импульс, который распределитель, показанный на предыдущем рисунке, донесет до нужного входа счетчика, можно аналогично тому, как мы получали короткие импульсы для подачи на S и R входы триггера. На диаграмме нужные импульсы представлены двумя нижними сигналами *in_count.X* и *out_count.X,* которые получаются благодаря изменению длительности импульсов генераторов S1 и S2.

Как я и говорил, я не собираюсь строить всю схему в данный момент, и затеял я «автомат для рассеянных» не только для того, чтобы показать возможность построения устройства на отдельных элементах. Мне хотелось показать, что когда в программе Qucs не удается полностью нарисовать и смоделировать схему, можно разделить ее на отдельные функциональные узлы, которые можно проверить по-отдельности.

К автомату для рассеянных мы вернемся при рассказе о микроконтроллерах. Но прежде я хотел рассказать об одном интересном преобразователе. О нем мне поведал еще один Александр, он же познакомил меня с программой PSIM. О преобразователе, о программе и из-за чего весь сыр-бор пойдет разговор в следующей главе.

Глава 13. Один интересный преобразователь

В конечном счете никому из нас не нужно ни напряжение, ни ток сами по себе. Ну, зачем нормальному человеку напряжение? Ему нужно то, во что это напряжение или тот ток можно превратить. Таким образом, вся электроника – это сплошное преобразование напряжения. Мы пытаемся преобразовать напряжение в свет, тепло, движение. В последнее время все больше внимания уделяется преобразованию напряжения в информацию. Ведь, согласитесь, работа компьютера, не более чем сложные преобразования напряжений, получаемых от блока питания. И выходит, что родоначальником того текста, который я сейчас печатаю в текстовом процессоре по имени *OpenOffice.org Writer*, вставляя картинки, получаемые на экране монитора при работе программы Qucs, родоначальником всего этого является импульсный блок питания, стоящий в системном блоке. Если у вас возникли сомнения в этом, отключите блок питания, и сомнения рассеются. Тема преобразователей напряжения частично была затронута выше, и, возможно, импульсным блокам питания следовало уделить гораздо больше места, чем я планирую, но это очень обширная тема, заслуживающая отдельной книги. И тема интересная, как для опытных, так и для начинающих любителей. Я не готов сделать большего, как только чуть-чуть рассказать о преобразователях на примере конвертера Cuk'a. Я использую материал, присланный мне Александром, статью из CHIP NEWS, авторы которой Ю. Розанов, М. Рябчицкий и А. Кваснюк, справочник «Источники электропитания РЭА» издательства «Радио и связь», 1985, и все, что смогу придумать на эту тему. Попутно я хочу обратить внимание, особенно начинающих, на тот факт, что программы симуляции, о которых я много говорю, не всегда могут успешно справляться с любыми схемами, и, являясь прекрасным инструментом и для изучения предмета, и для работы, требуют обязательной проверки на макетной плате «всех личных достижений» в области электроники. Только профессионалы, годами работающие с одной программой разработки электрических схем или разводки печатных плат, готовы определить, нуждаются ли результаты работы в дополнительной проверке. Можно утверждать, что любителям это тоже доступно, но я не рекомендовал бы уповать на подобные утверждения, лучше, все-таки, проверить. Выбор за вами!

Вначале, чем интересны импульсные преобразователи в практическом плане? Напомню, что простейшим преобразователем переменного напряжения будет трансформатор, давно и активно используемый в электрике. Поэтому речь пойдет о преобразовании постоянного напряжения в переменное или постоянного напряжения в постоянное. Так что в них, в этих преобразователях?

Преобразователи постоянного напряжения в переменное

Давным-давно, отправляясь в путешествие на собственном автомобиле, радиолюбители использовали преобразователь напряжения аккумулятора для питания электрической бритвы. Нужно же бриться! Самый простой преобразователь получался из генератора с самовозбуждением. На выходе такого генератора не синусоидальное напряжение, но электрические бритвы не были столь привередливы. Сам я таким преобразователем не пользовался, но, как мне кажется, встречал схемы похожих преобразователей именно для использования с электробритвами в автомобиле.

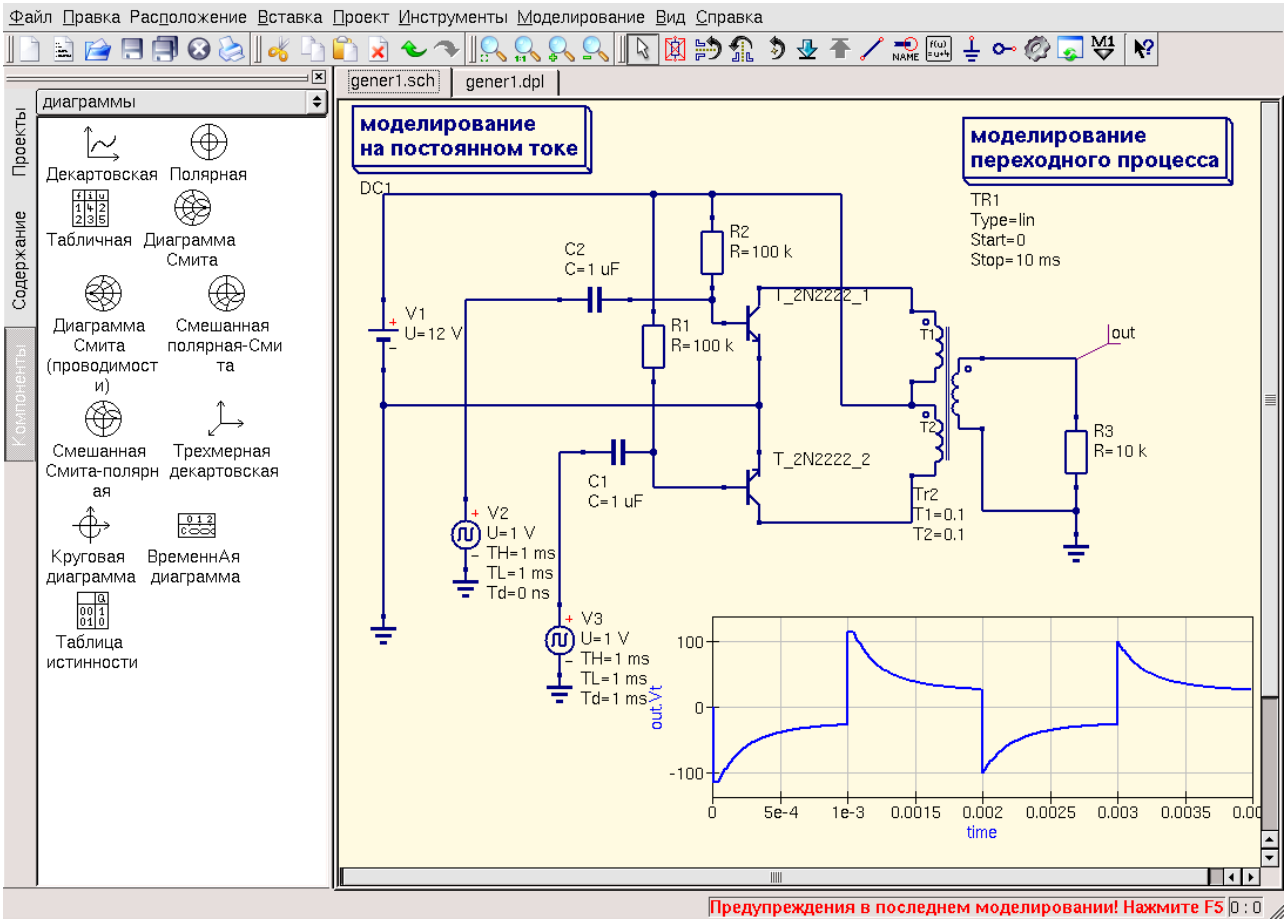


Рис. 13.1. Простейший преобразователь постоянного напряжения в переменное

В таких преобразователях, как правило, используются дополнительные обмотки трансформатора Tr2, включаемые в базовые цепи транзисторов, для получения самовозбуждения генератора. Мне пришлось использовать два генератора (второй работает в противофазе с первым, обратите внимание на его параметр Td). Но принцип работы реального преобразователя похож: при включении питающего напряжения V1 начинает возрастать ток коллектора транзисторов, протекая по обмоткам T1 и T2 трансформатора; наводимое в обмотках, включенных в цепи баз, напряжение (на рисунке их нет) для одного транзистора способствует нарастанию тока коллектора, для другого препятствует. В итоге один из транзисторов полностью открывается, а второй закрывается, но не сразу – индуктивное сопротивление «сопротивляется» этому. Однако, когда изменение тока прекращается, это приводит к уменьшению напряжения во всех вторичных обмотках трансформатора (он не «трансформирует» постоянный ток), и это, в свою очередь, приводит к закрыванию первого транзистора и открыванию второго. Затем процесс повторяется.

Не следует думать, что так, как изображено на рисунке, схему собрать нельзя. Можно. В этом случае генератор-преобразователь будет работать не в режиме самовозбуждения, а в режиме принудительного преобразования, и транзисторы не более, чем каскады усиления, что я и хотел подчеркнуть их обычным включением, и в этом случае есть определенная выгода – легче настроить частоту преобразования. Если в генераторе с самовозбуждением частота будет определяться во многом свойствами трансформатора, которые трудно менять, то с независимыми генераторами (или, как правило, генератором) частота может задаваться, например, RC-цепью, где можно сделать резистор переменным, позволяющим легко получить требуемую частоту.

Как реально происходит процесс запуска генератора в режиме самовозбуждения можно посмотреть на следующем рисунке.

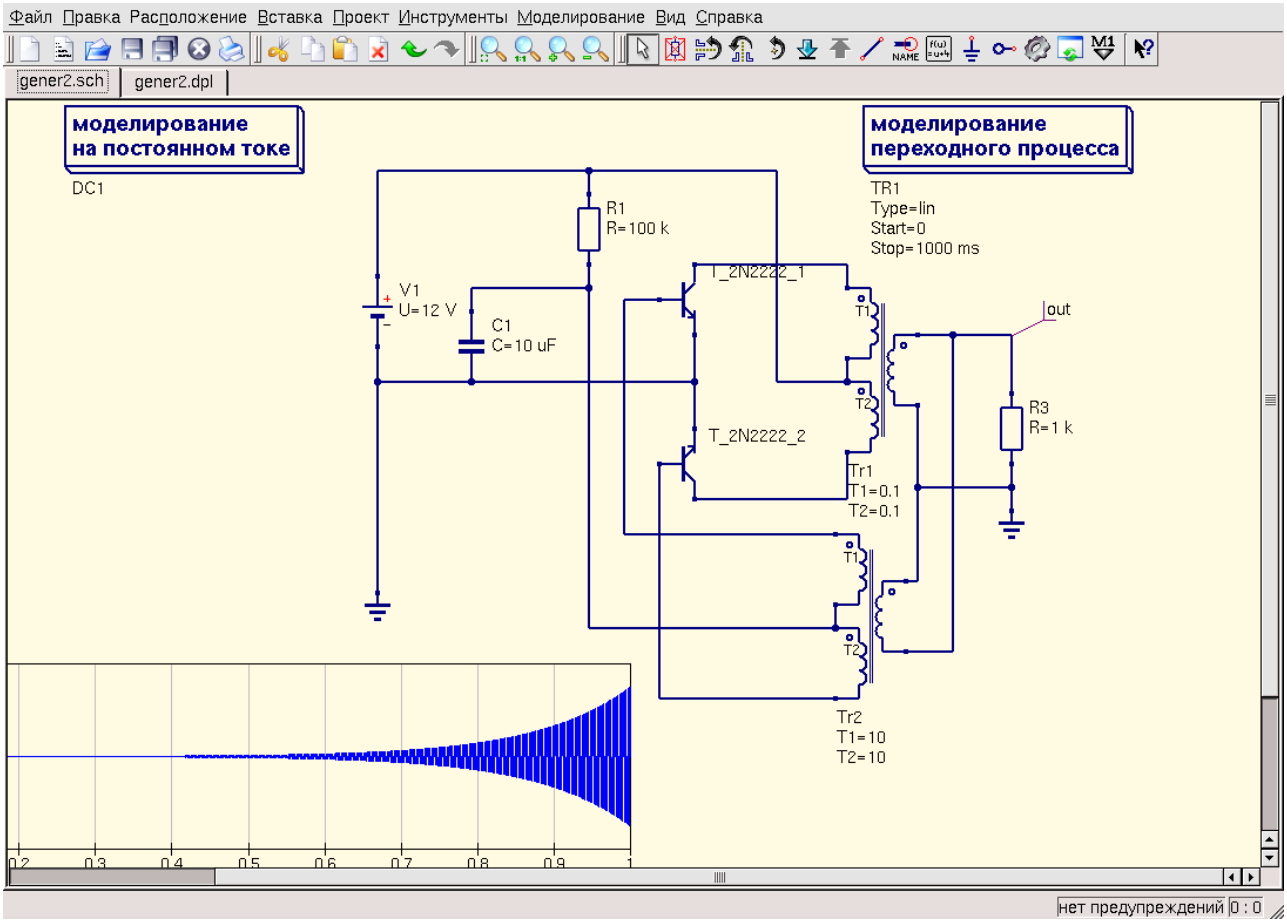


Рис. 13.2. Моделирование процесса самовозбуждения генератора

Использование похожих генераторов с самовозбуждением я встречал в приборах. Действительно, если не считать трансформатора, во всем остальном проще не придумаешь. Трансформатор, кстати, может иметь две выходные обмотки, а в этом случае легко получить двух-полярный блок питания от одного источника напряжения, например, батарейки, а два полученных источника напряжения использовать для питания операционных усилителей. Более простое, правда, решение, без трансформатора, я встречал в схеме тестера, где использовался операционный усилитель, а питания тестера было построено на одной батарейке. Добавленная цифровая микросхема использовалась для построения генератора и преобразователя.

Иногда для питания электронной схемы переносного устройства требуется высокое напряжение, которое трудно получить от батарейки, и в это случае используется преобразователь. Подобное решение я встречал в весьма современном пульте управления фирмы Philips для питания лампы, подсвечивающей сенсорный дисплей.

Разные типы современных преобразователей

Статью, которую я упоминал, «Вторичные источники питания: от сетевого трансформатора до корректора коэффициента мощности» авторы предваряют словами: «За короткий срок вторичные источники питания радикально изменились: на смену громоздким устройствам с сетевым трансформатором и линейным стабилизатором пришли миниатюрные импульсные

модули...».

Не буду пересказывать статью, любой пересказ предвносит точку зрения автора, что никак не улучшает исходный материал, а вот что хочу, так это «перепоказать» некоторые положения статьи. Авторы подразделяют преобразователи по виду схем на: понижающие, повышающие, повышающе-понижающие и, наконец... а вот об этом типе отдельно.

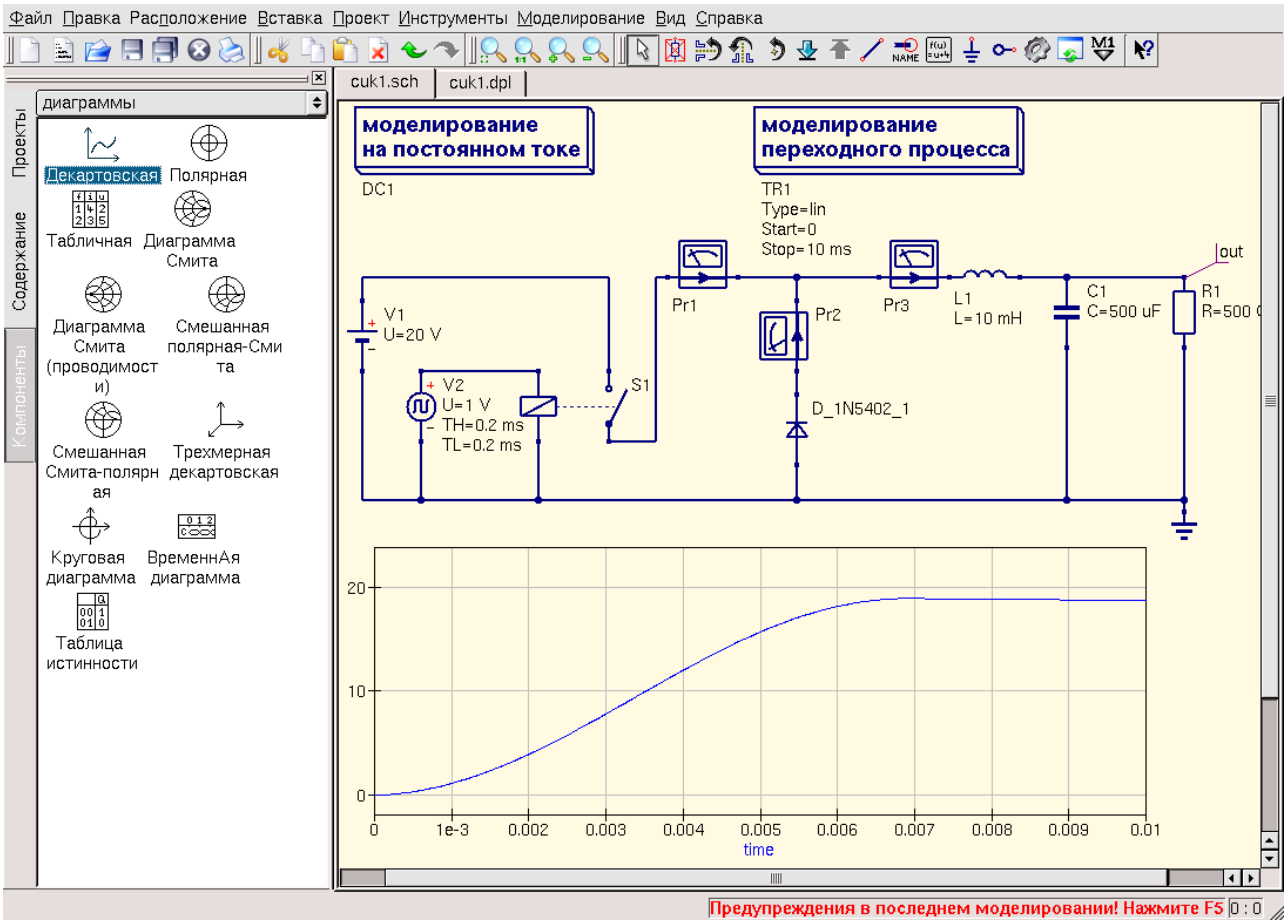


Рис. 13.3. Понижающий преобразователь

Для моделирования схемы в программе Qucs в качестве ключевого элемента использовано реле. Разновидность реле, специально предназначенная к этому, некогда использовалась в преобразователях. Но в данном случае реле используется только для моделирования в программе в сочетании с генератором прямоугольных импульсов V2. В реальных схемах используются транзисторы (на месте контактов S1), и именно появлению транзисторов с подходящими параметрами импульсные источники питания обязаны столь быстрой экспансией. Конечно, работа транзистора в режиме «ключа» имеет ряд особенностей, но в первом приближении можно рассматривать его работу следующим образом. Для включения транзистора необходимо задать такой базовый ток, при котором транзистор полностью открыт, то есть, его напряжение коллектор-эмиттер минимально, что определяется током коллектора и сопротивлением нагрузки. А в режиме выключения ток базы может отсутствовать, что приведет к уменьшению тока коллектора до «неуправляемого остатка», а все напряжение будет падать на транзисторе, приложенное к его эмиттеру и коллектору.

В статье приводится несколько диаграмм, иллюстрирующих работу преобразователя. Я постарался сделать диаграммы, соответствующие диаграммам статьи.

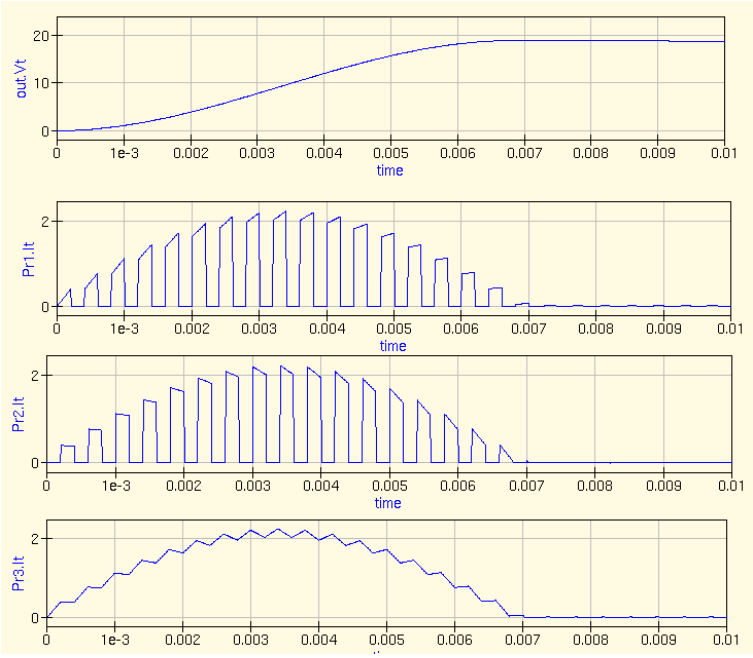


Рис. 13.4. Диаграммы симуляции схемы, включая токи амперметров

Попробуйте изменить параметр TL генератора V2, скажем, увеличив его до 0.4 мС. Вы должны получить диаграмму напряжения на выходе (метка *out*) следующего вида:

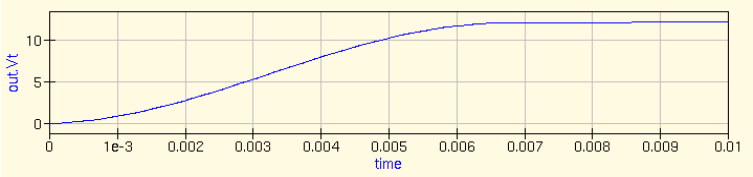


Рис. 13.5. Изменение выходного напряжения при увеличении TL

Именно так, изменением коэффициента заполнения, можно регулировать выходное напряжение. Думаю, вам доставит удовольствие придумать способ автоматической регулировки выходного напряжения.

Схема повышающего преобразователя основана на эффекте накопления энергии в индуктивности при замыкании ключа с последующей отдачей этой энергии, когда ключ размыкается, в нагрузку. Чем дольше ключ замкнут, тем больше выходное напряжение. Схема, приводимая в статье, соответствует эксперименту, схема которого представлена ниже. Как и в предыдущем случае, вы можете менять значения замкнутого состояния ключа (параметр TH генератора) и разомкнутого (TL), а так же оценить влияние значения индуктивности, конденсатора фильтра и сопротивления нагрузки. Я не устанавливал измерителей тока на схеме, но вы можете найти их в разделе *измерители* на вкладке *Компоненты* программы, чтобы получить набор диаграмм, соответствующий приведенным на рисунке 13.4.

Измерения в электронике, наряду с расчетами, играют основополагающую роль. Не пренебрегайте возможностью лишний раз измерить или увидеть на экране осциллографа все, что происходит в схеме. Впоследствии вам легче будет представить работу схемы, глядя только на схему и не пользуясь описанием работы схемы. Такая возможность полезна, когда нет описания схемы. И особенно полезна, когда нет самой схемы. Устройство есть, а схемы

нет.

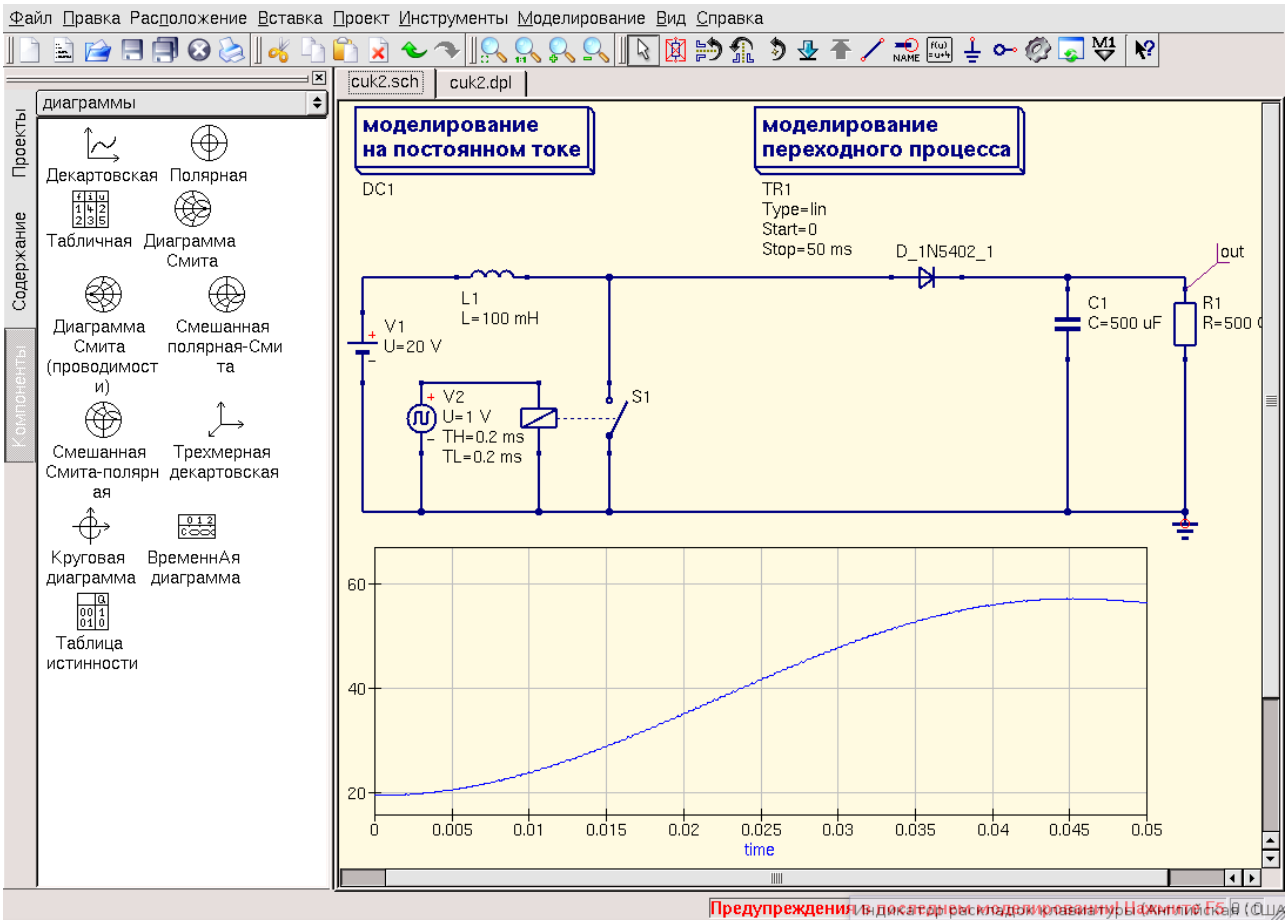


Рис. 13.6. Повышающий преобразователь

Получение более высокого напряжения, чем напряжение источника питания, весьма привлекательная особенность преобразователя. Особенно, когда вы ограничены применением батарейки в переносном устройстве. Особенно, когда вам очень хотелось бы использовать операционный усилитель в своей схеме, а он требует для своего питания двух источников, положим, + 15 В и – 15 В. И никак не соглашается на меньшее. А батарейка, которую вы можете применить не дает больше 9 В. И особенно привлекательно для многих то, что не требуется трансформатор.

Следующий тип преобразователя повышающе-понижающий. Он позволяет регулировать выходное напряжение от нуля до значения, ограниченного паразитными параметрами схемы. И обратите внимание на схему ниже, полярность выходного напряжения отлична от полярности источника питания! Тоже полезная особенность преобразователя.

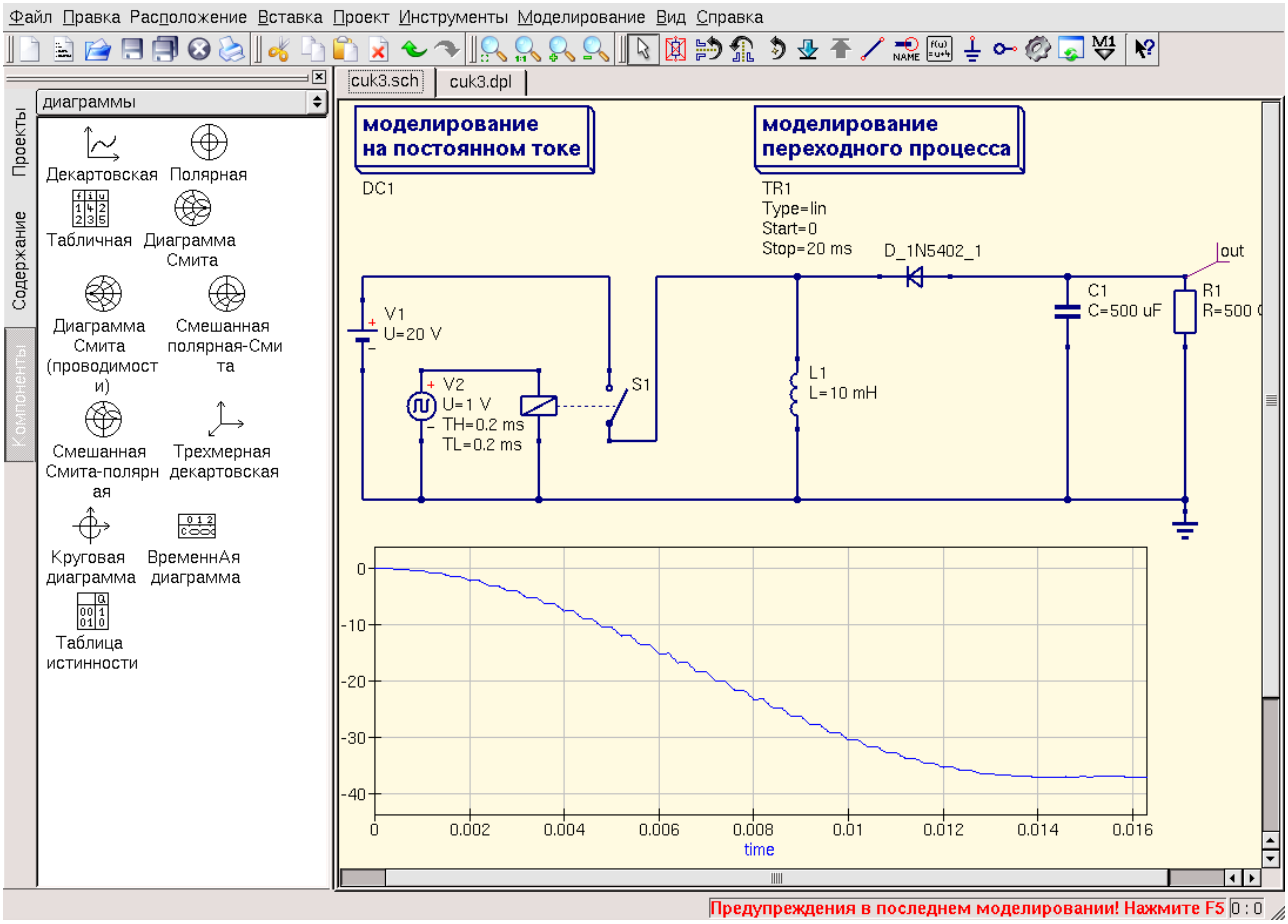


Рис. 13.7. Повышающе-понижающий преобразователь

Используя схемы преобразователей, подобные описанным в статье, следует не забывать, что генератор, управляющий работой ключа, должен обязательно работать. Если ключевой транзистор окажется включен, скажем, на схеме выше, а генератор работать не будет, то нагрузкой транзистора станет индуктивность L1, активное сопротивление которой может быть очень маленьким. Для транзистора подобное включение это короткое замыкание, и он может не выдержать подобного испытания. Конечно, вы добавите, надеюсь, в схему предохранитель, но имейте в виду, что предохранитель «срабатывает» гораздо медленнее транзистора. Удобный подход в этом случае таков: работа ключевого транзистора начинается с выключенного состояния. То есть, если вы включили схему, транзистор выключен, а если генератор не заработал, то транзистор оказывается выключен, и ничего страшного для него в этой ситуации быть не должно.

И, наконец, последний тип преобразователя, упоминаемый в статье, это преобразователь Чука (Čuk). О существовании этого типа преобразователя мне сообщил Александр Кушнеров, который и привлек мое внимание к преобразователям вообще, и к существованию проблем с симуляцией подобных преобразователей, да и импульсных схем, в программах EDA. Именно он предложил мне осуществить симуляцию этого конвертера в программах Qucs и PSIM. Именно по причине симуляции работы импульсных схем я заговорил в этой книге о программе PSIM. Программа специализированная, и когда я писал о программах EDA, я не стал ее рассматривать. А Александр, которому приходится уделять много времени импульсным преобразователям, использовал эту программу с наибольшим успехом. Впрочем, и программа LTSpice, о которой я рассказывал в книге «Наглядная электроника» тоже успешно справляется с симуляцией импульсных схем. Так что, имейте в виду, что не все

программы успешно справляются с симуляцией любых схем. Но вернемся к преобразователям.

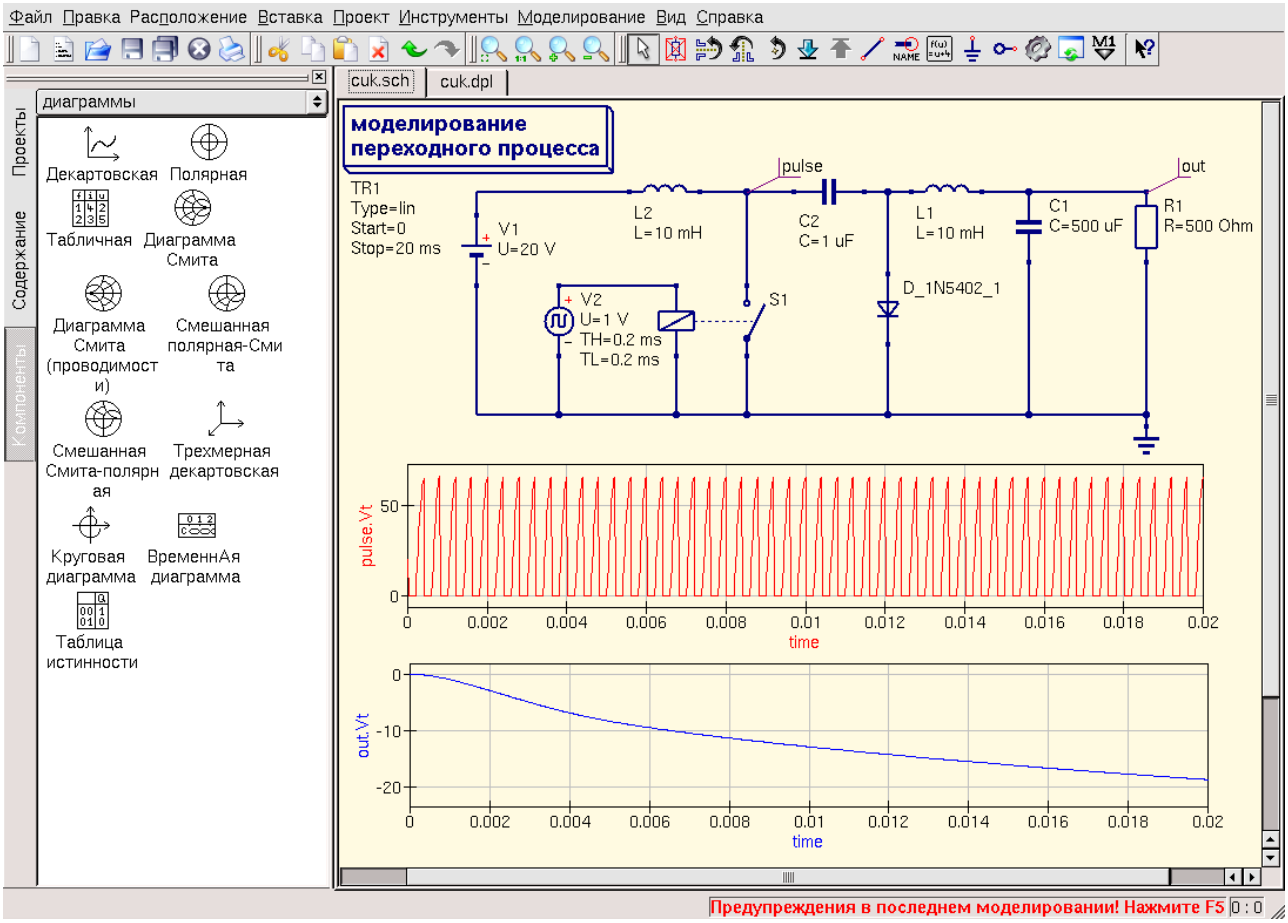


Рис. 13.8. Преобразователь Чука

Как видно из схемы, преобразователь Чука имеет черты сходства и с повышающим, и с понижающим преобразователем. Напряжение на выходе, в зависимости от длительности включенного состояния ключа, может быть и больше, и меньше напряжения источника питания. Как пишут авторы статьи, одним из важных свойств этого преобразователя является низкий уровень пульсаций входного и выходного токов, что, конечно, упрощает построение входных и выходных цепей преобразователя.

Бурному росту использования импульсных источников питания ранее мешало то, что они строились с использованием биполярных транзисторов в качестве ключевых элементов схемы. Особенности работы этих транзисторов в ключевом режиме ограничивали такие выходные параметры вторичных источников питания как удельные (по мощности) габариты, вес. Появление полевых транзисторов существенно улучшило ситуацию и ускорило переход от громоздких схем с понижающими трансформаторами к компактным и легким импульсным блокам питания. Конечно, как пишут авторы, ничто не дается даром. Широкое использование импульсных вторичных источников питания, благодаря их импульсному характеру работы, привело к появлению новых проблем, связанных с обратным негативным воздействием на силовые цепи, с незапланированным излучением радиоволн, как самими преобразователями, так и проводами электросетей. Сегодня эти проблемы рассматриваются и успешно решаются средствами современной электроники.

Импульсные преобразователи – это очень интересная область электроники, но я хотел бы

вернуться к тому, с чего начал эту главу. Если предыдущая схема достаточно уверенно работала в программе Qucs, то предложенный Александром вариант, в сущности мало отличающийся от изображенного на рисунке выше, при запуске симуляции работать не хотел.

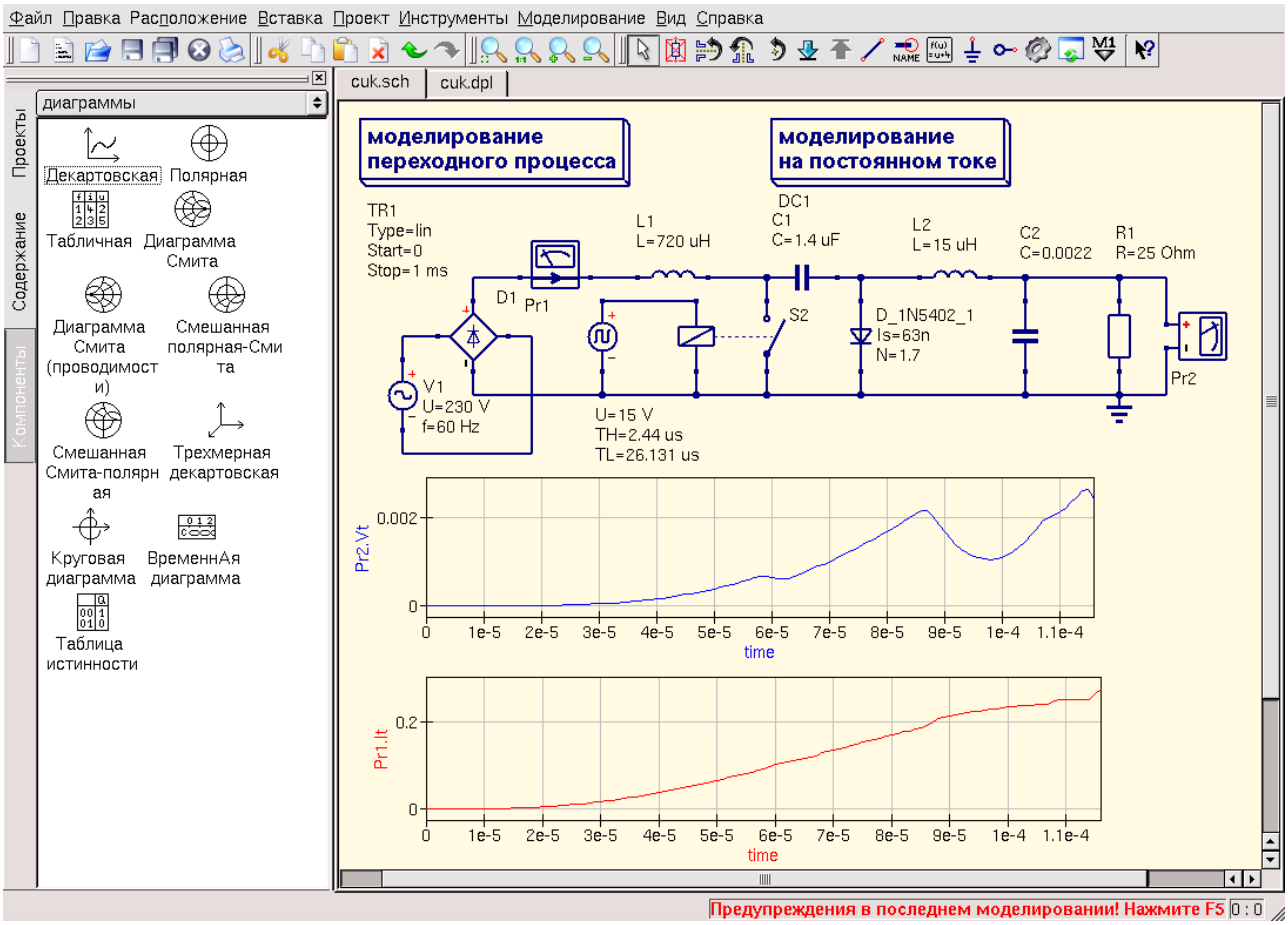


Рис. 13.9. Модификация предыдущей схемы

В отличие от Александра, занимающегося исследованиями в области импульсных схем, я «шапочно» знаком с ними. Проведя несколько попыток осуществить симуляцию, я быстро отказался от этого и попробовал осуществить эту симуляцию в программе PSIM по совету Александра, который опробовал несколько программ, и в частности Multisim, но более всего его удовлетворила работа в PSIM. Конечно, его интересовали исследования более глубокие, чем обычно осуществляются в любительских условиях. Но трудно сказать, и я не берусь это утверждать с уверенностью, что может заинтересовать любителя, если он истинный любитель, а уж терпения и знаний у него, подчас, не меньше, чем у профессионала. В этом смысле мне показалось поучительной эта история с преобразователем Чука. Мне кажется, что последовав моему совету и приняв в качестве одного из инструментов программу для работы со схемами, начинающий любитель электроники с большей вероятностью столкнется с подобной ситуацией, и возникающие трудности при симуляции схемы могут навсегда оттолкнуть его. Было бы очень жаль, если б такое случилось. Вот как выглядит работа схемы в программе PSIM. Схему я получил от Александра, но сама программа работает в демонстрационной версии, что ограничивает ее возможности. Однако процесс симуляции проходит быстро, да и ключевой элемент использован «наиновейший» – полевой транзистор.

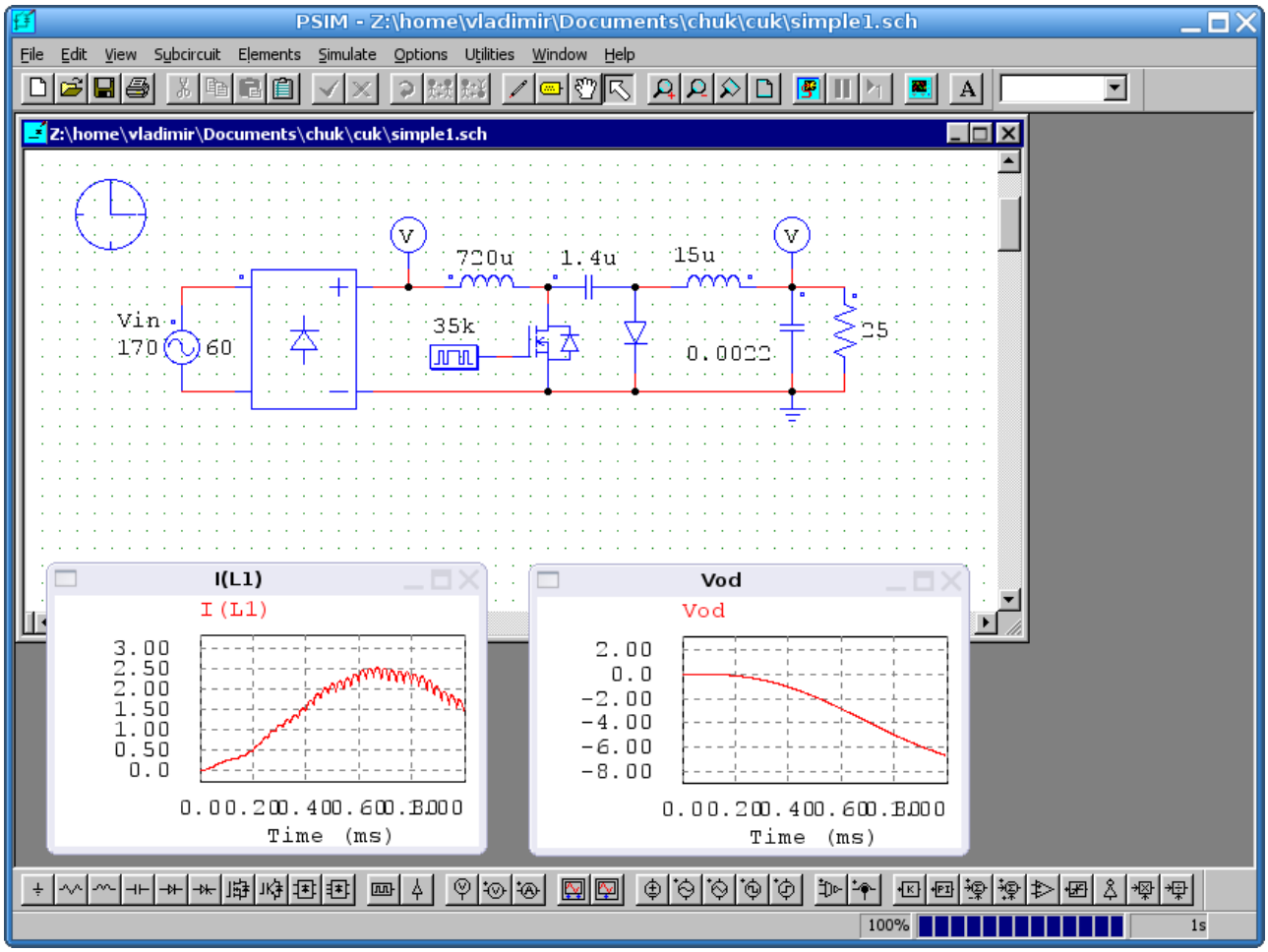


Рис. 13.10. Работа предыдущей схемы в программе PSIM

Захотите окунуться в мир преобразователей – выбирайте «бассейн» по имени PSIM или SwCAD III. Первая программа есть в демо-версии, вторая имеет «усеченную» версию под названием LTSpice. А я хочу перейти к тому, почему я заговорил об этом вообще.

Проблемы симуляции электрических схем

Люди давно используют специализацию в общественной жизни. Никому, или почти никому, не приходит в голову все делать самому, не обращаясь к помощи других. Одни из нас хорошо умеют выращивать пшеницу, другие варить сталь, третьи из стали делать комбайны, чтобы первые могли собрать выращенную пшеницу. Как сказал классик: «Мамы всякие нужны, мамы всякие важны».

Программы, как Qucs, дают возможность моделировать работу сотен различных схем, но, видимо, есть ограничения, когда некоторые задачи выполняются не лучшим образом. Для тех любителей электроники, чьи интересы ближе, скажем, авиа-моделированию, а там тоже есть где приложить свои силы радиолюбителям, при работе с исполняющими устройствами в виде электромоторов может больше пригодится некая специализированная программа. Значит ли это, что Qucs не для них. Отнюдь. Можно так преобразовать задачу, что полученные функциональные узлы вполне хорошо позволят обработать себя и в этой программе. Означает ли, что программа Qucs хуже, чем программа PSIM? Тоже нет. Далеко не факт, что универсальная во всех отношениях программа, позволяющая делать все, не окажется настолько сложной в работе, что не специалист не сможет реализовать в ней даже

простейшую схему. Но я ушел в сторону от того, о чем хотел рассказать.

Проблемы симуляции, скорее математические, чем электрические, и о них можно было бы не упоминать. Но задача эта интересна сама по себе, и, кроме того, кто-то, кто начинал как радиолюбитель может почувствовать тягу к программированию или математике. А без этих специалистов сегодня электроника явно не может обойтись. Я уже говорил, ссылаясь на статью, что сегодняшним успехам импульсные блоки питания во многом обязаны появлению новых типов полевых транзисторов. Но они-то появились не сами, их создавали физики, математики, технологи, машиностроители – люди сотен специальностей, это их труд вложен во все современные технологии. И электроника, как одна из областей человеческой деятельности, открывает двери в смежные области, что, как мне кажется, очень хорошо.

Специалисты в области электроники не обязаны быть хорошими математиками, однако они обязательно изучают математику. Если вам надоело «тупо» повторять чужие схемы, вы рано или поздно захотите научиться создавать свои, и обязательно столкнетесь с математикой. Так вот одна из проблем с симуляцией, как я говорил, скорее математическая, чем электрическая. При симуляции электрической схемы компьютер производит вычисления. Чтобы не быть голословным, я приведу уравнения из работы Александра по конвертеру Чука, относящиеся к варианту, не представленному на рисунке, но они помогут мне рассказать о процессах, происходящих в самом симуляторе работы электрической схемы.

$$\begin{cases} V_{L1} = L_1 \cdot \dfrac{di_1}{dt} + M \cdot \dfrac{di_2}{dt} \\ V_{L2} = L_2 \cdot \dfrac{di_2}{dt} + M \cdot \dfrac{di_1}{dt} \end{cases}$$

Рис. 13.11. Уравнения при взаимоиндукции в электрической схеме

Программа-симулятор работы электрической схемы должна обработать подобные уравнения. Для этого есть численные методы решения уравнений, но я не стану пересказывать соответствующие разделы учебника по математике, а обращу ваше внимание только на существование производных в этих уравнениях.

Мы знаем, что скорость обычного движения, пешком ли, на автомобиле, выражается отношением пути ко времени его прохождения. Отмеряем сто метров. Включим секундомер при начале движения бегуна по дорожке, остановим секундомер, когда бегун пересечет финишный створ, и с гордостью можем записать: V=S/t. Мы определили скорость, с которой он бежал. Н это, увы, не даст нам представление о характере его бега, о тех мучениях, которыми достался ему рекорд. А, ведь, при симуляции электрических схем нас весьма интересует именно характер движения, определяющий сигналы, с которыми мы имеем дело.

Мы можем разбить весь путь на равные промежутки, отмечая время прохождения этих промежутков. Для каждого из этих промежутков мы можем записать: V=ΔS/Δt. Чем меньше ΔS, тем больше участков разбиения, но тем точнее мы определяем скорость в малые промежутки времени, а, значит, и характер движения. Когда промежуток времени стремится к бесконечно малой величине, мы наиболее точно знаем характер изменения интересующей нас величины, но при этом объем вычислительной работы становится бесконечно большим. И, заметьте, что если при движении происходят резкие изменения: промежуток Δt очень мал, а величина (эк, он рванул) ΔS велика, скорость резко вырастает. Если мы не будем учитывать этих резких бросков, то можем получить неверную картину, а если будем учитывать, то их нужно отыскивать и оценивать, оценивать их влияние на конечный результат.

Примерно такую работу выполняет компьютер, когда работает программа-симулятор. Мы далеко не всегда можем до выполнения симуляции сказать, какой характер будет носить изменение той или иной величины интересующих нас токов или напряжений. Этого не знаем ни мы, ни программа. Чтобы эти вопросы как-то решить в программе используют разные алгоритмы вычислений. Большинство программ EDA работает с системой SPICE, имеющей определенные вычислительные алгоритмы. Они успешно справляются с различными электрическими цепями, но не с любыми, и не в любой ситуации. Например, если задать параметры преобразователя близкими к тем, что на рисунке 13.10, но заменить выпрямитель источником постоянного напряжения, и смоделировать поведение конвертера Чука в программе Qucs, то мы получим следующий результат.

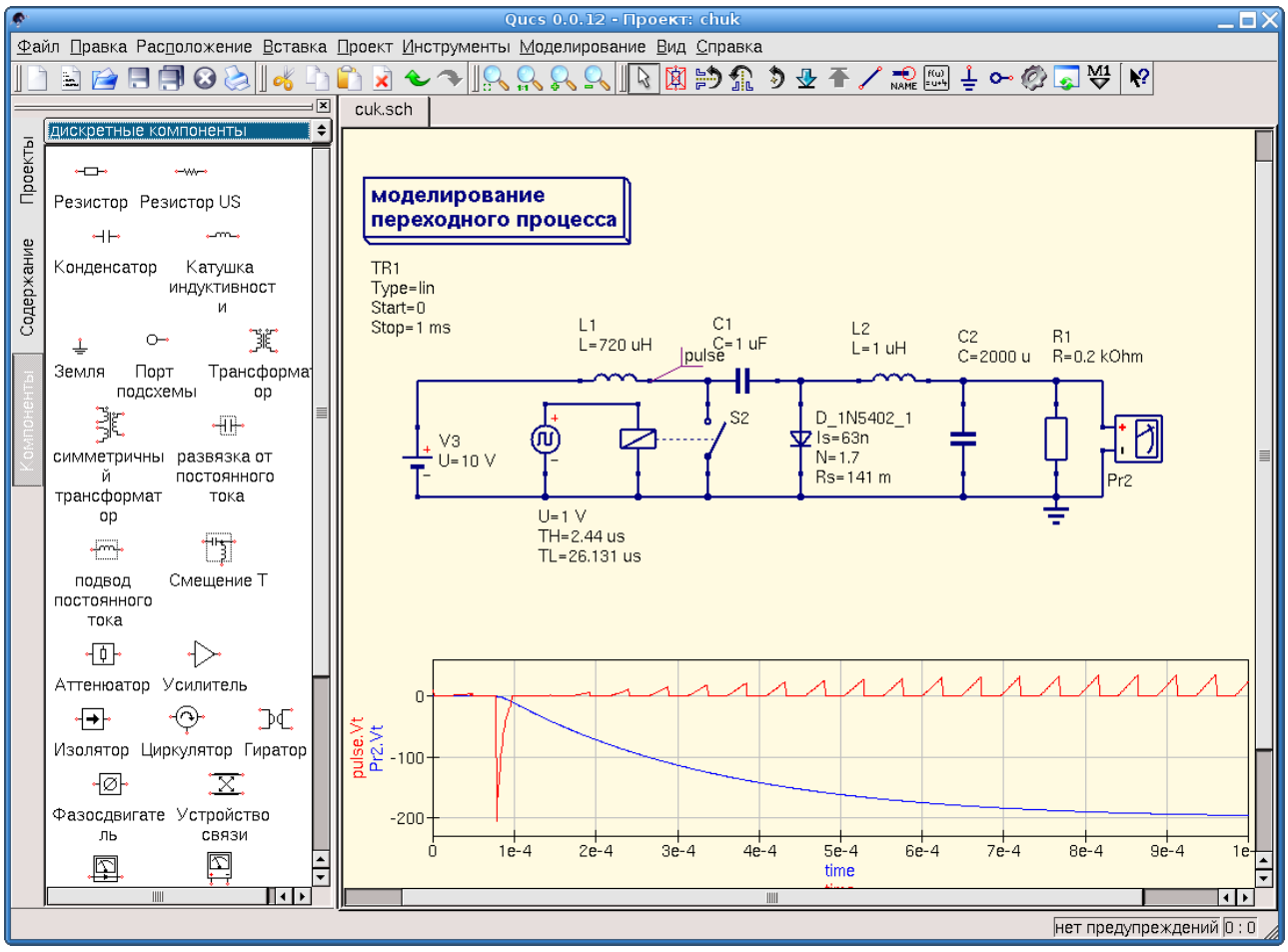


Рис. 13.12. Моделирование конвертера Чука в программе Qucs

Напряжение на выходе при питающем напряжении 10 В составляет – 200 В. Каким будет это напряжение при выпрямлении переменного напряжения ~ 230 В? Если это действительно так, то пытаясь повторить эксперимент «живьем», на макетной плате, мы наверняка столкнемся «с определенными трудностями». Не сталкивается ли с ними и программа Qucs? И, если мы изменим алгоритм вычислений, то не «выплеснем» ли мы с водой и ребенка? Нужно подумать.

А пока я хочу продемонстрировать, как легко можно ошибиться, на примере рисунка 13.9, выделив часть из собственноручно построенной схемы, которая у меня не захотела работать.

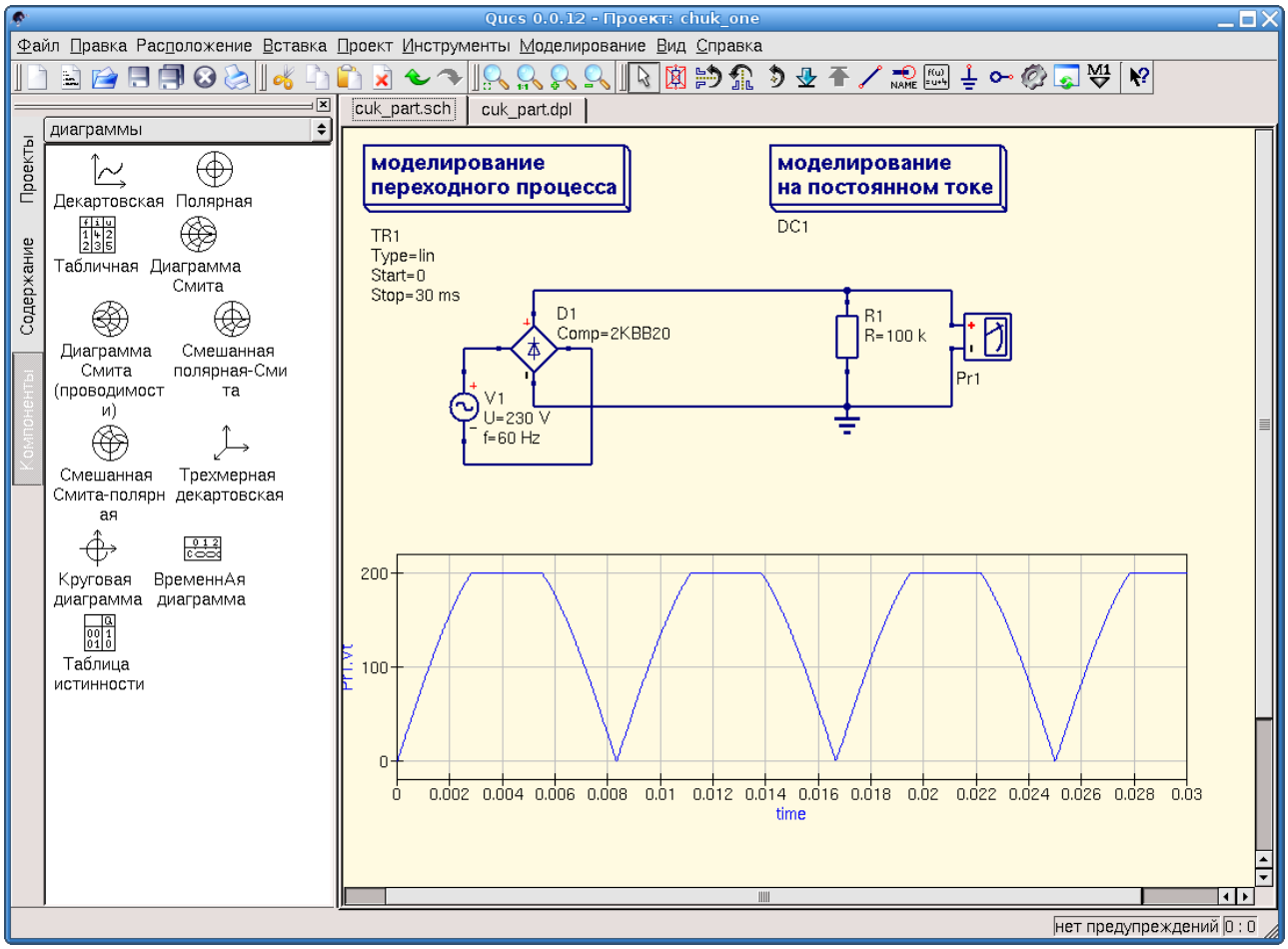


Рис. 13.13. Моделирование выпрямителя в программе Qucs

Меня очень удивила полученная диаграмма. Думаю, и вас она приводит в замешательство. Куда подевались вершины знакомой нам с детских лет синусоиды? Я специально выделил на этом рисунке тип, использованного мной моста: 2KBB20. Последние две цифры – сокращение от напряжения, видимо, предельно допустимого для этого моста, то есть, 200 В. Вот куда пропали верхушки синусоид. Поменять мост на 2KBB40, и они появятся. Но суть, все-таки не в этом, проблема симуляции есть, и от этого никуда не денешься, но...

Как вы видите, это тоже интереснейшая область исследований. И если, решив все проблемы, вы построите нужный вам конвертер, то сможете увидеть на экране осциллографа работу своего творения.

Рис. 13.14. Конвертер Чука на экране осциллографа

Это изображения я тоже взял из работы Александра Кушнерова, если он не возражает,

поскольку получить его на экране своего осциллографа не удосужился. И причина этого проста, я намерен сбежать из далекой мне области преобразователей к более мне знакомым микроконтроллерам, но перейти к рассказу окольным путем. Очень окольным путем.

Глава 14. Как считает домашний компьютер

Мало кто сегодня пользуется компьютером для выполнения расчетов. Хотя задумывался компьютер именно для этой цели. И обошлось не без участия военных. Им нужны были средства для решения дифференциальных и интегральных уравнений, средства расчета траекторий снарядов. Первыми компьютерами, использовавшимися для этого, были аналоговые компьютеры. Операционные усилители позволяли быстро собрать дифференцирующие или интегрирующие, суммирующие или масштабирующие усилители, сочетания которых превращали входные функции в нужные значения. Для решения каждой задачи приходилось собирать схему для решения именно этой задачи. Вот один из примеров.

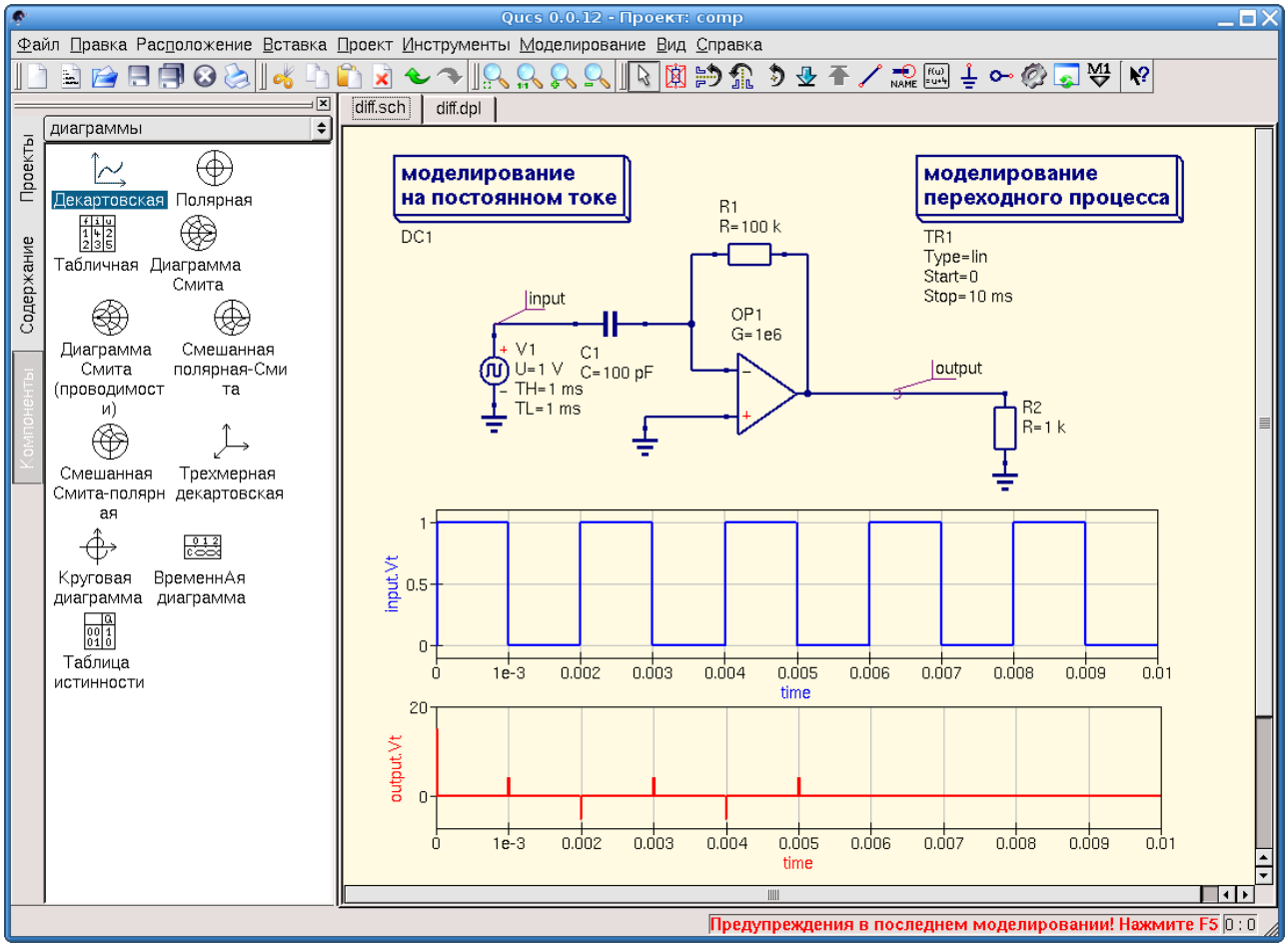


Рис. 14.1. Дифференцирующий усилитель

Кроме калькулятора, входящего в состав любой операционной системы, сегодняшний компьютер предлагает еще одно удобное средство расчетов – электронную таблицу. Если вам приходится часто выполнять одни и те же расчеты, то даже очень удобный калькулятор, как в Linux, не спасет вас от необходимости ввода всех операций и напряженного отслеживания процесса вычислений.

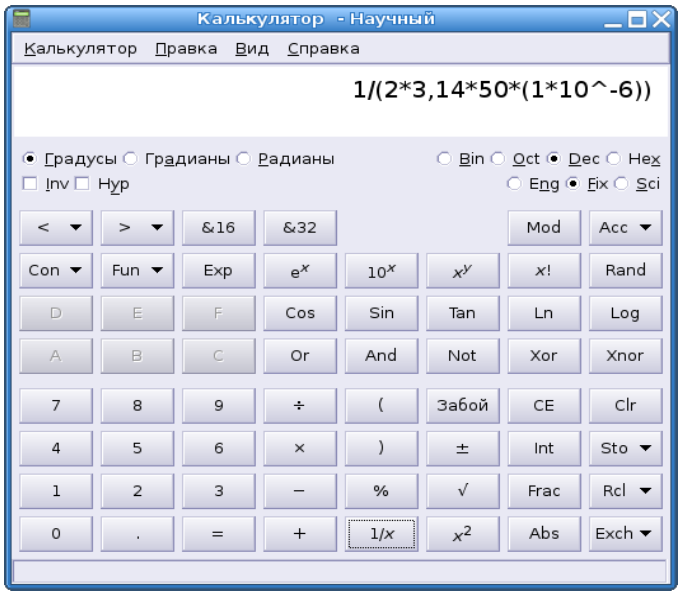


Рис. 14.2. Калькулятор в оконном менеджере Gnome

Возьмем, например, вычисление сопротивления конденсатора на частоте сети, 50 Гц, это показано на калькуляторе. Придумаем повод для этого – нам хотелось бы заменить гасящий резистор для светодиода конденсатором. Для этого нужно знать величину сопротивления конденсатора. Если применить конденсатор в 1 мкФ, то результат вычислений на калькуляторе будет около 3 кОм. Нам надо погасить напряжение 220 В при токе 10 мА. Величина гасящего резистора должна быть порядка 22 кОм. Значит, конденсатор нужно взять с емкостью раз в семь меньше. А захочется изменить ток? Опять пересчитывать.

Для таких часто повторяющихся вычислений есть более удобное средство – электронная таблица. Даже в те давние времена, когда наиболее распространенным редактором в нашей стране был К52, а документы создавались в роскошном текстовом редакторе Lexicon, даже в те давние времена существовали электронные таблицы. Они не были столь удобны, не позволяли сделать многое из того, что доступно сегодня, но даже тогда их охотно применяли все, кому приходилось часто повторять однотипные расчеты.

Думаю, что электронные таблицы задумывались для применения в бухгалтерских расчетах, хотя могу и ошибиться в своем предположении, но мне, хотя и недолго, довелось использовать электронную таблицу для составления смет на проведение работ. Как правило, состав и цены на проведение работ определяются достаточно однозначно, но заказчику иногда очень важно, чтобы окончательная сумма в смете не выходила за некоторые границы. Переделывать смету с целью снизить стоимость работ, уменьшая, где это возможно, объем работ, занятие очень утомительное. И как же я был приятно удивлен, когда создание сметы в электронной таблице сводилось именно к изменению отдельных цифр в объемах работы, все остальное таблица мгновенно пересчитывала сама.

Но я опять отклонился от рассказа.

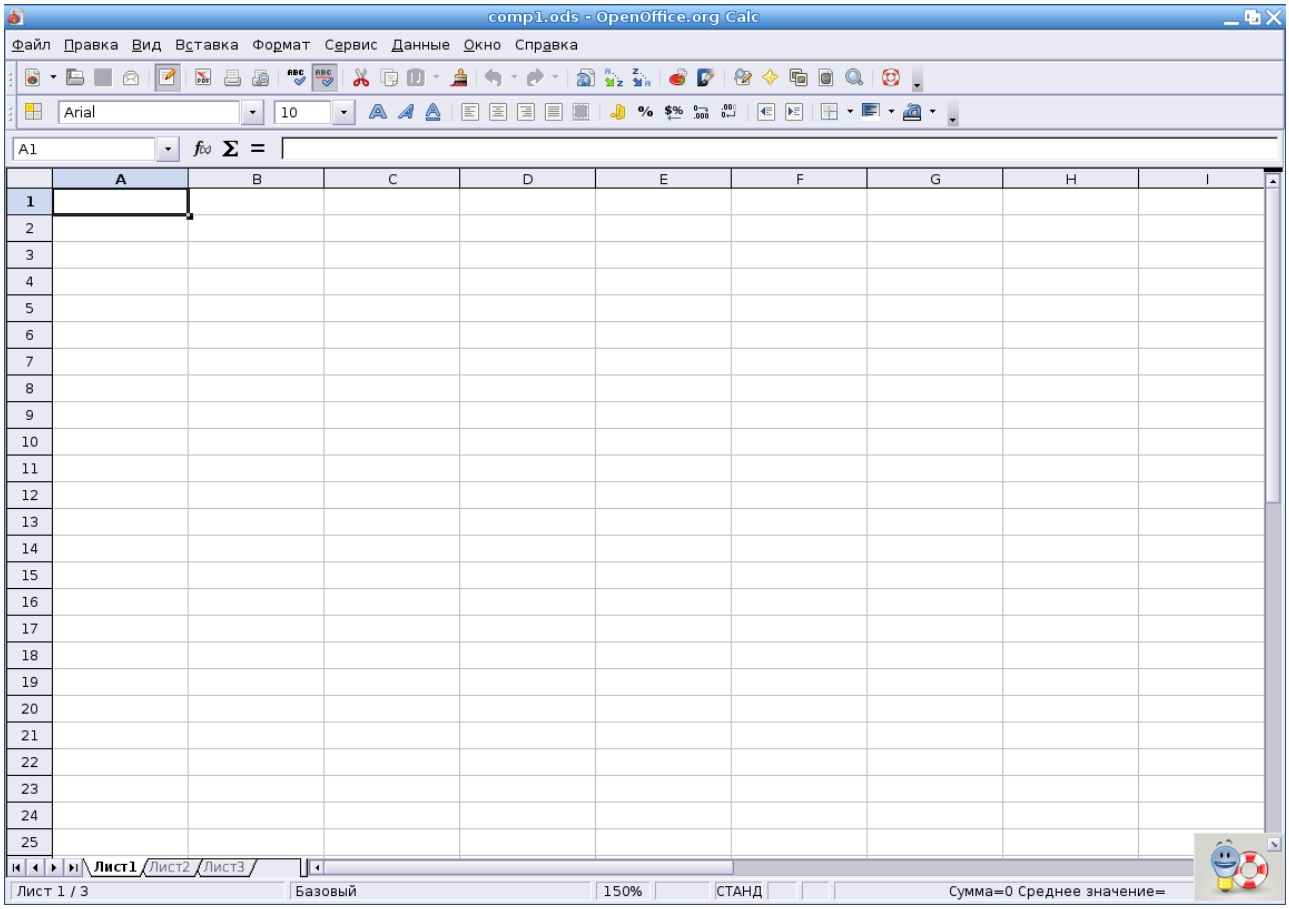


Рис. 14.3. Одна из электронных таблиц в Linux

В таблицу можно вписать несколько вариантов расчета данных. Положим, часто приходится определять результирующее сопротивление двух параллельно включенных резисторов. Создадим подходящую запись расчета. Для этого в первую ячейку впишем: *Параллельно включенные резисторы*. Надпись, если ничего не делать, продолжится поверх следующих ячеек.

Затем в ячейку A2 впишем R1 – наш первый резистор, в следующие ячейки последовательно впишем R2 и Rрез. Это позволит нам впоследствии понять, где у нас что. Под обозначениями резисторов зададим их величину (в Омах), а в ячейку под Rрез, впишем формулу. Для этого достаточно переместить маркер к ячейке и нажать знак «=» на клавиатуре. Расчетная формула: 1/(1/A3 +1/B3). Теперь можно задавать нужные величины резисторов и, нажав клавишу *Enter*, получать ответ.

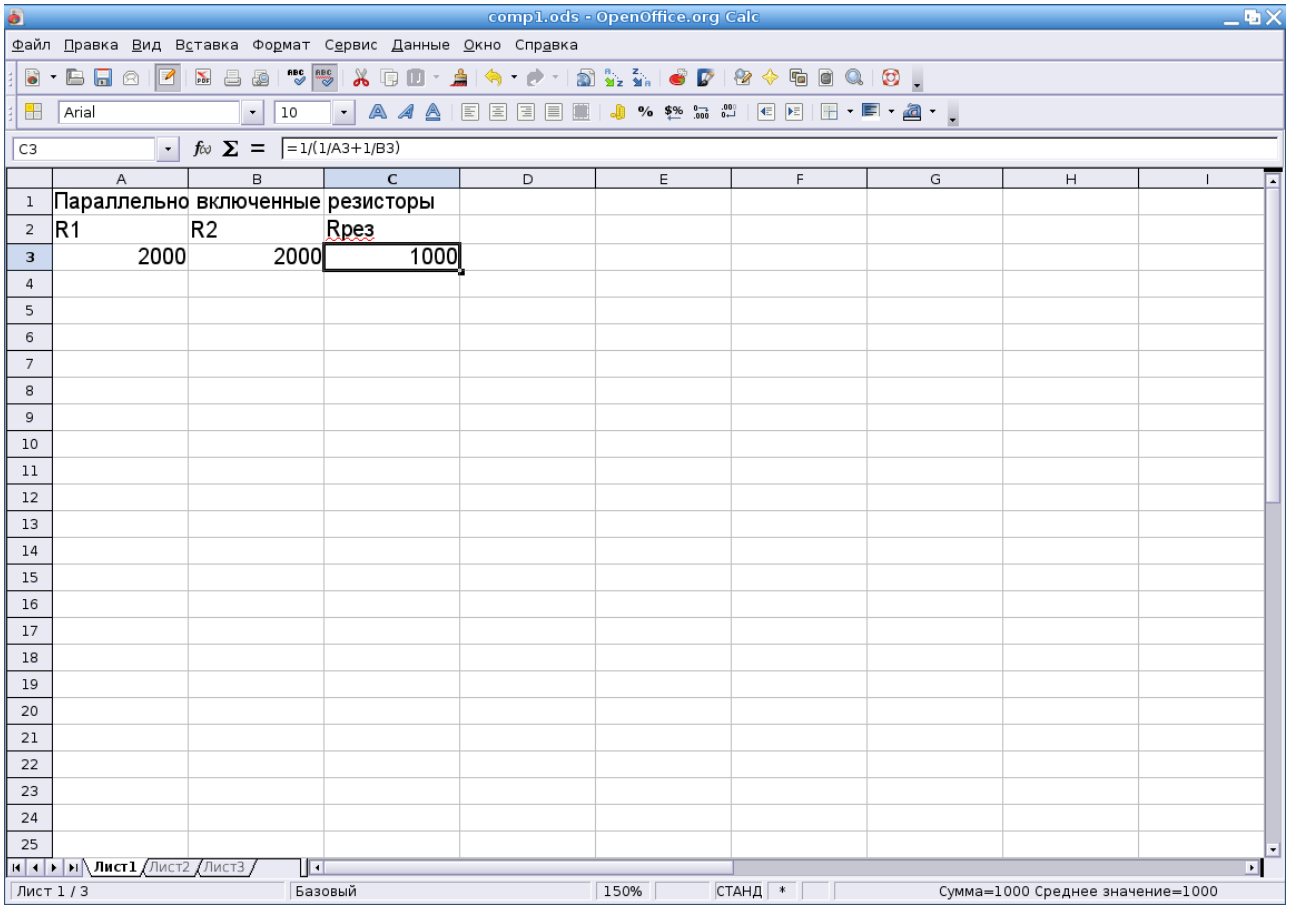


Рис. 14.4. Расчет параллельного включения резисторов

Подобный расчет достаточно прост, обычно не хочется затевать дополнительных операций, наподобие использования электронных таблиц, но, если вам часто приходится подбирать резисторы, попробуйте подсчитать, сколько лишнего времени вы потратили на работу с калькулятором.

На той же странице таблицы можно сделать расчет емкостного сопротивления на заданной частоте при вводе емкости конденсатора. Не самый сложный расчет. Если не приходится его многократно повторять.

Современные электронные таблицы хорошо приспособлены к анализу данных. Они позволяют создавать и поддерживать базы данных, имеют в своем составе большой набор математических функций. Многие типовые математические операции, связанные с расчетами элементов электрических схем, можно с помощью формул представить в электронной таблице. Создание «расчетного листа» займет у вас не больше времени, чем требуется для записи формул, но сэкономит много времени, если вы будете пользоваться этими записями. Единственная трудность, например, возникла у меня при добавлении расчета емкостного сопротивления и была связана с определением количества знаков после запятой в результате вычислений. Вернее, результат вычислений при делении, скажем, единицы на триста оказывался равным нулю, пока я не увеличил количество знаков после запятой до трех. И еще одна мелочь может помешать быстрой записи формулы для расчетов – запятая или точка используется в десятичных числах. Проще всего это можно выяснить методом проб (и ошибок), хотя можно еще посмотреть формат числа.

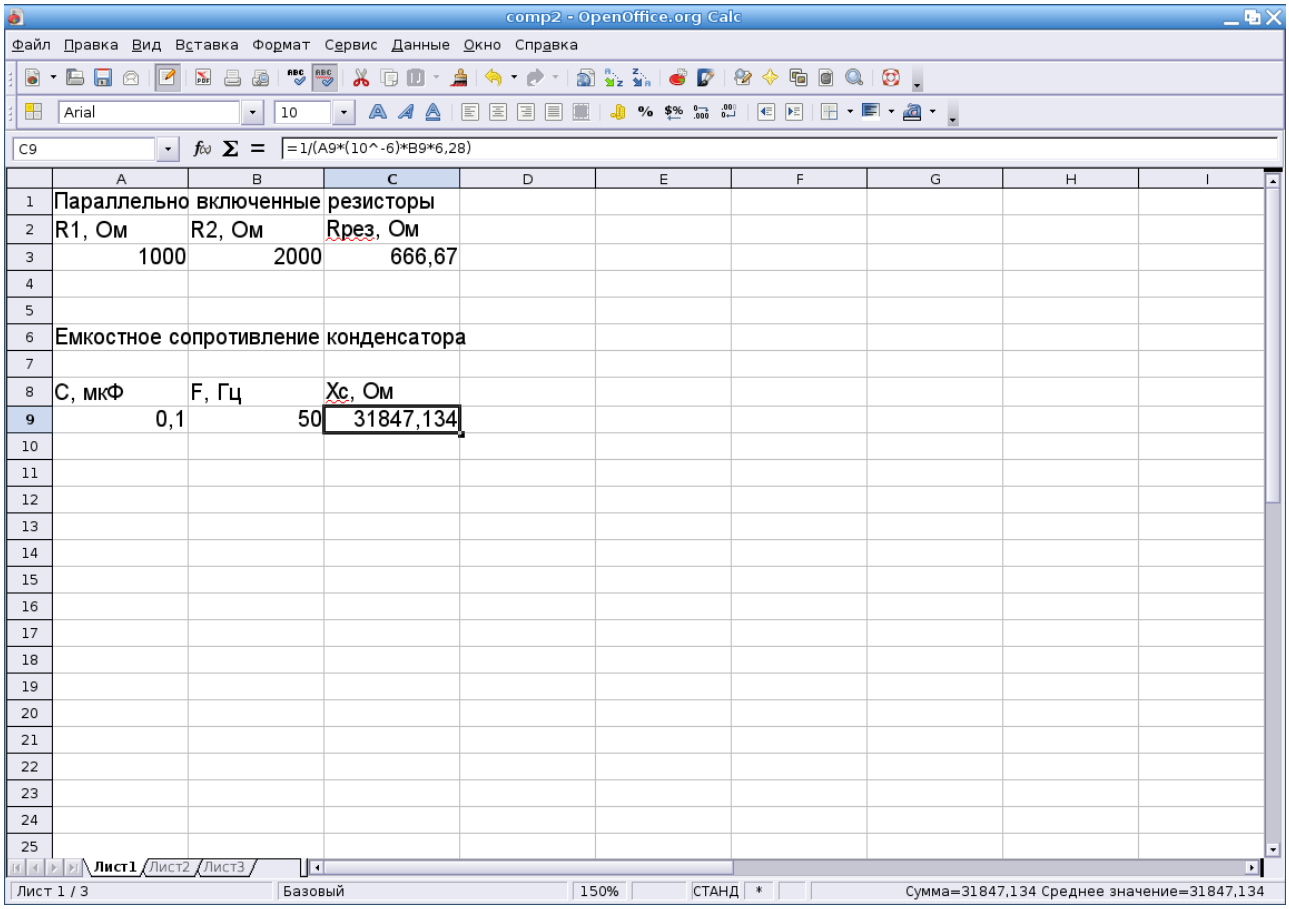


Рис. 14.5. Добавление второго расчета в электронной таблице

Для более сложных математических операций есть специальные программы, такие как matlab в Windows и scilab (или octave) в Linux. Они поддерживают свой язык расчетов и требую больше времени на освоение этого математического инструмента, и начинающему любителю могут быть совершенно не интересны. Но это только до тех пор, пока вам не захочется самостоятельно произвести полный расчет, например, усилителя мощности.

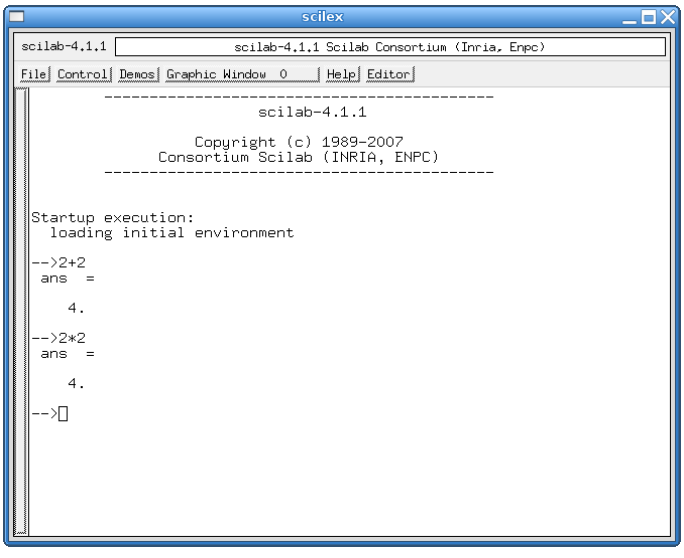


Рис. 14.6. Программа scilab

Как и в любом другом случае вы можете записать расчетные формулы на бумаге, а вычисления сделать с помощью калькулятора. Но не думаю, что вам удастся с первого раза достичь желаемого результата, и почти уверен, что после второй или третьей попытки вы забросите это надолго. А напрасно. Попробуем провести расчеты, касающиеся тех тем, которые были рассмотрены ранее. Я старался не загружать рассказ формулами и расчетами с тем, чтобы посвятить им отдельную главу. Ее можно пропустить при первом чтении. Она не дает ничего нового. Но если вам захочется рассчитать, можно воспользоваться тем, что есть в этой главе.

Пример расчета максимальной выходной мощности

(двухтактного УМЗЧ по материалам книги П. Шкритека «Справочное руководство по звуковой схемотехнике»).

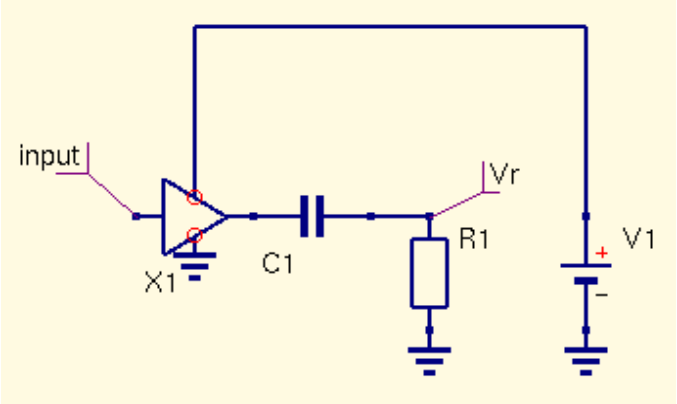


Рис. 14.7. Схема для расчета мощности

Максимальная мощность выходного каскада при усилении синусоидального сигнала достигается при полном размахе (сумма амплитуд положительной и отрицательной полуволн) выходного напряжения равном напряжению питания. Но этому препятствует эффект «насыщения» транзисторов, когда напряжение эмиттер-коллектор полностью открытого транзистора не равно нулю.

Расчетные формулы (при условии, что напряжение насыщения для биполярных выходных транзисторов равно 1.5 В):

$Vr = V1/2 - 1.5 = (V1 - 3)/2$ – амплитудное значение выходного напряжения.

$Vr.eff = (V1 - 3)/2*1.41$ – действующее значение для синусоидального сигнала.

$Ir = Vr/R1$ – амплитудное значение выходного тока.

$Ir.eff = Vr/R1*1.41$ – действующее значение для синусоидального сигнала.

$Pr = Vr^2/R1$ – пиковое значение мощности в нагрузке.

$Pr.eff = Vr^2/R1*2$ – действующее значение.

Для режима А:

$Pm = Vr.eff^2/R1 = Vr^2/2R1 = Vr*I_0/2$ – мощность, рассеиваемая на транзисторе в отсутствии сигнала, где I_0 = Vr/R1 определяет рабочую точку.

Для режима В:

$Pm.max \sim 0.1Vr^2/R1$ – максимальная рассеиваемая на транзисторе мощность.

Формулы достаточно просты, чтобы разместить их в электронной таблице. Подобный шаг, казалось бы, не более, чем прихоть ярого приверженца виртуального компьютерного мира. Нормальный человек может это подсчитать в уме. И это так. И не так.

Сегодня вы помните эти формулы, завтра забыли – первая причина для создания таблицы.

Вы задумали усилитель мощности. Рассчитали напряжение питания и получили нужную мощность на выходе усилителя. Но напряжение питания оказалось несколько отличным от расчетного. Что произошло с вашей расчетной мощностью? - вторая причина.

Вы задумали и рассчитали усилитель мощности, но решили сделать систему трех-канальной. Разделить полностью низкие, средние и высокие частоты. По меньшей мере для высокочастотного усилителя вам потребуется пересчитать усилитель с учетом другого сопротивления нагрузки – третья причина.

Но, вместе с тем, я согласен, как только все начинаешь перекладывать на компьютер, так сам все начинаешь забывать, и я не уверен, что смогу извлечь квадратный корень из числа без помощи компьютера – минус одна причина.

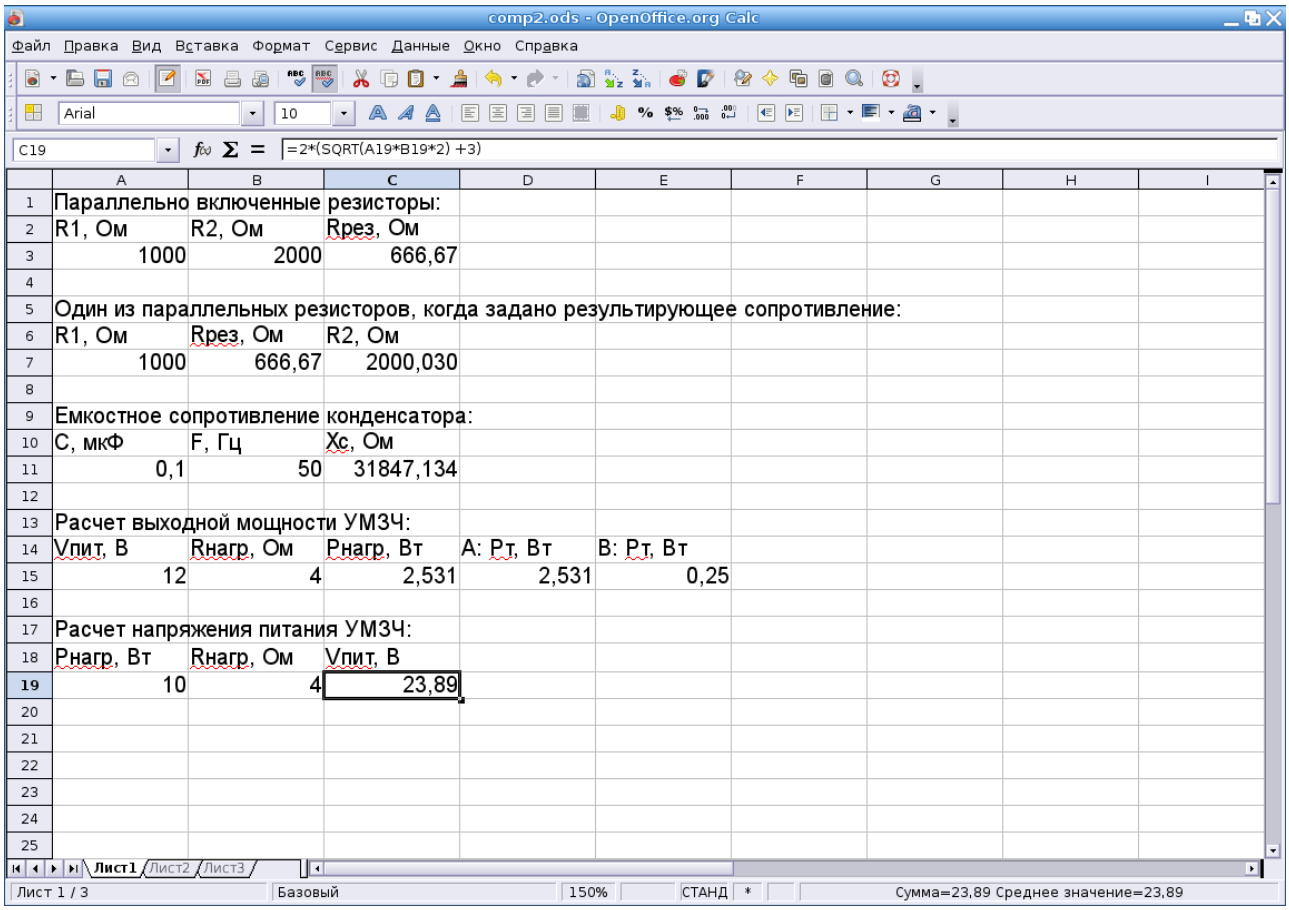


Рис. 14.8. Пример расчета УМЗЧ в электронной таблице

Пример расчета схемы стабилизатора

(по материалам книги Э.Н. Воронкова и Ю.А. Овечкина «Основы проектирования усилительных и импульсных схем на транзисторах»).

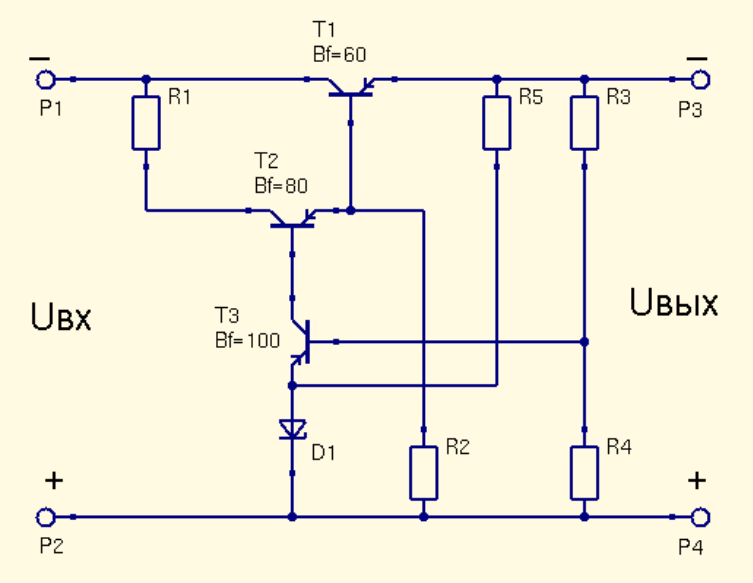


Рис. 14.9. Схема для расчета стабилизатора

Исходные данные:

Кс – коэффициент стабилизации равный *(ΔUвх/ΔUвых)*100, %.*

Uвых – выходное напряжение.

Iн.макс - максимальный ток нагрузки.

Rвых – выходное сопротивление равное *ΔUвых/ΔIн*.

Uвх.макс и Uвх.мин – максимальное и минимальное входное напряжение.

Порядок выбора параметров транзисторов :

1. Выбор транзистора T1 по току: Iк.макс > Iн.макс, где Iк.макс – предельно допустимый ток коллектора транзистора.
2. Выбор транзистора T1 по напряжению: Uк-э.макс > Uвх.макс – Uвых, где Uк-э.макс – предельно допустимое напряжение коллектор-эмиттер транзистора.
3. Предельно допустимая мощность рассеивания на коллекторе транзистора T1 (может потребоваться расчет радиатора): *Рк.макс1 > (Uвх.макс – Uвых)*Iн.макс.*
4. Предельно допустимый ток коллектора транзистора в усилителе T2: Iк.макс > *(Iн.макс/βмин) + Iрез2*, где βмин – минимальный коэффициент усиления по току транзистора T1, Iрез2 – ток, протекающий через резистор R2 и равный Uвых/R2 (при условии, что он больше I к.обр – максимальный обратный ток коллекторного перехода при максимальной рабочей температуре).
5. Предельно допустимое напряжение транзистора усилительного каскада T2: Uкэ.макс2 ~ Uкэ.макс1.
6. Предельно допустимая мощность рассеивания T2: *Рк.макс2 > Uкэ.макс2*Iк.макс2*.
7. Предельно допустимый ток коллектора транзистора сравнивающего каскада T3: *Iкэ.макс3 > (Iк.макс2/ βмин2) + Iрез1*, где βмин2 – минимальный коэффициент усиления по току T2, Iрез1 – ток, протекающий через сопротивление R1 и прближенно равный *(Uвх – (Uвых – 1))/R1*.
8. Предельно допустимое напряжение на коллекторе T3: *Uк-э.макс3 > Uвых – Uстаб,*

где Uстаб – рабочее напряжение стабилитрона D1.

9. Предельно допустимая мощность рассеивания транзистора Т3: *Рк.макс3 > Iк.макс3*(Uвых – Uстаб).*

Примечание: с увеличением коэффициента усиления по току регулирующего и усилительного транзисторов коэффициент стабилизации возрастает.

Порядок расчета схемы:

- Определение значения R1: *R1 = Uрез1/Iрез1 ~ (Uкэ.макс1* βмин1* βмин2)/Iн.макс*, где выбор резистора определяется, с одной стороны, стремлением обеспечить максимальное усиление каскада, а с другой, желанием получить максимальный ток регулирующего каскада. Отысканию оптимального значения помогают следующие соотношения: *Iрез1 ~ Iб.макс2 ~ Iн.макс/(βмин1*βмин2).*
- Определение значения R2: *R2 = Uрез2/Iрез2 ~ Uвых/(2÷3)Iк.обр,* где для определения величины резистора используются соотношения *Uрез2 = Uвых – Uэб1, Iрез2 = (2÷3)Iк.обр* (Iк.обр см. п.4 выше).
- Определение значения резисторов делителя R3 и R4 начинается с учетом того, что ток через делитель должен более чем на порядок превосходить ток транзистора сравнивающего каскада, и составлять порядка 10% от максимального тока нагрузки. Задаваясь, с учетом вышесказанного, током через делитель Iдел, рассчитывается общее сопротивление делителя *R3 + R4 = Uвых/Iдел*. Напряжение на резисторе R4 определяется как Uрез4 ~ Uстаб, а резистор *R4 = Uстаб/Iдел*. Соответственно *R3 = (Uвых – Uстаб)/Iдел.*
- Определим R5: *R5 = (Uвых – Uстаб)/0.005*, где 0.005 – оптимальный ток через стабилитрон, равный 5 мА.
- Окончательно определяем коэффициент стабилизации и выходное сопротивление: *Кс ~ (β_3 * $r_{к1}$ *$r_{к2}$)/$r_{экв}$($r_{к1}$ + $r_{к2}$)*; *Rвых ~ $r_{экв}$/(β_1* β_2* β_3)*, где $r_{экв} = r_д + r_{б3} + \beta_3 r_{э3}$.

Используемые в последних формулах:

$r_д$ – сопротивление диода, как я полагаю, дифференциальное сопротивление в рабочей точке, которое можно взять из справочника. Например, для стабилитрона КС447А при рабочем токе 43 мА и нормальной температуре $r_д$ = 18 Ом.

$r_э$, $r_б$, $r_к$ – дифференциальные сопротивления эмиттера, базы и коллектора, насколько я понимаю, эквивалентной Т-образной схемы транзистора. Эти данных в справочнике найти не всегда можно, но ориентироваться можно на следующее:

$r_э$ – дифференциальное сопротивление эмиттерного перехода, включенного в прямом направлении, обычно составляет десятки Ом.

$r_б$ – сопротивление базы, обычно составляет несколько сотен Ом.

$r_к$ – дифференциальное сопротивление коллекторного перехода, смещенного в обратном направлении, обычно составляет сотни тысяч Ом.

Дифференциальное сопротивление стабилитрона можно получить по его статическим рабочим характеристикам зависимости тока от напряжения, взяв малые приращения напряжения и тока в рабочей точке: $r_д = \Delta U/\Delta I$. Или можно измерить эти значения. Аналогично можно поступить и с дифференциальными сопротивлениями эмиттера и коллектора.

Не буду приводить вид таблицы, поскольку формулы мало отличаются от предыдущих по

характеру, но хочу предложить вам, записав формулы, провести симуляцию в программе Qucs и, собрав схему на макетной плате с удобными для исследования параметрами, провести проверку и сравнение всех трех испытаний: расчетного, программного и реального. Уверен, будет очень интересно и познавательно.

Примеры соотношений в транзисторных схемах

Некоторые соотношения для расчетов транзисторных схем, могут быть получены с использованием статических характеристик конкретных типов транзисторов, но для качественной оценки данные можно получить, используя программы симуляции с учетом того, что программы, как правило, не имеют моделей отечественных транзисторов.

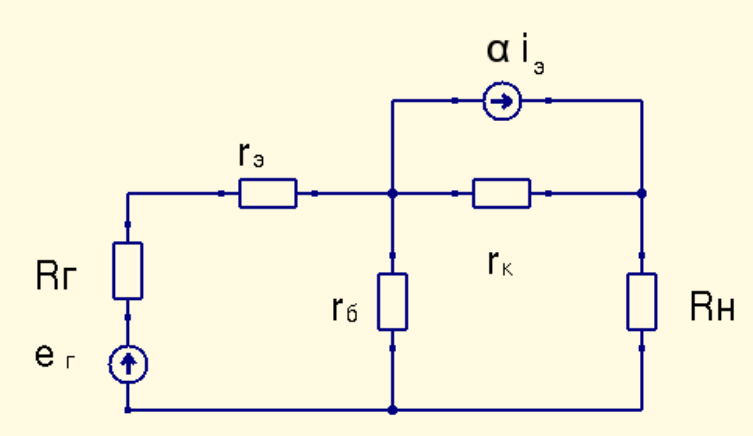


Рис. 14.10. Эквивалентная схема включения транзистора с общей базой

Для схемы с общей базой:

Коэффициент усиления по току $Kiб = i_к/i_э = \alpha < 1$.

Входное сопротивление $Rвх.б = u_{э\text{-}б}/i_э$.

Коэффициент усиления по напряжению $Kн.б = u_н/u_{вх} = i_к*Rн/(i_э*Rвх.б) = \alpha*(Rн/Rвх.б)$.

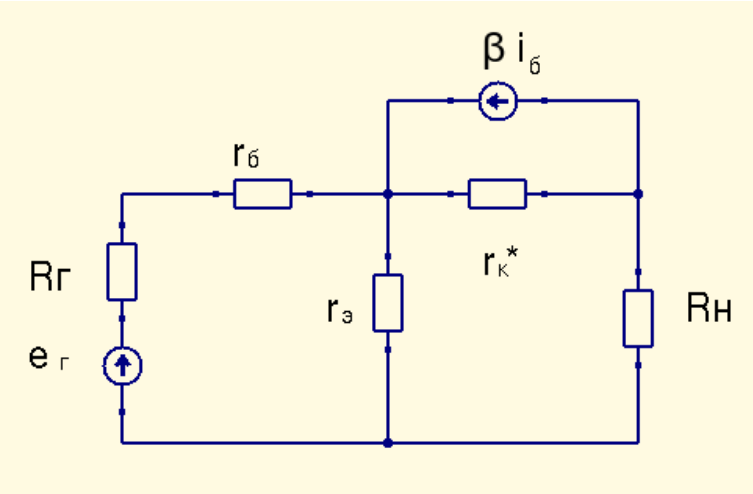


Рис. 14.11. Эквивалентная схема включения транзистора с общим эмиттером

Для схемы с общим эмиттером:

Коэффициент усиления по току $Kiэ = i_к/i_б$, но так как $i_к = \alpha * i_э$, $i_б = i_э - i_к = (1 - \alpha)\, i_э$, то

$Kiэ = (\alpha * i_э)/(1 - \alpha)\, i_э = \alpha/(1 - \alpha) = \beta$.

Входное сопротивление $Rвх.э = u_{б\text{-}э}/i_б = u_{б\text{-}э}/(1 - \alpha)\, i_э$

или $Rвх.э = Rвх.б/(1 - \alpha) = Rвх.б(\alpha + 1 - \alpha)/(1 - \alpha) = Rвх.б(\beta + 1)$, поскольку по величине $u_{б\text{-}э} = u_{э\text{-}б}$.

Коэффициент усиления по напряжению $Kиэ = u_н/u_{вх} = i_н * Rн/(i_б * Rвх.э) = \beta(Rн/Rвх.э)$. Заменяя Rвх.э и β, можно получить $Kиэ = (\alpha Rн(1 - \alpha))/(Rвх.б(1 - \alpha)) = \alpha(Rн/Rвх.б) = Kиб$.

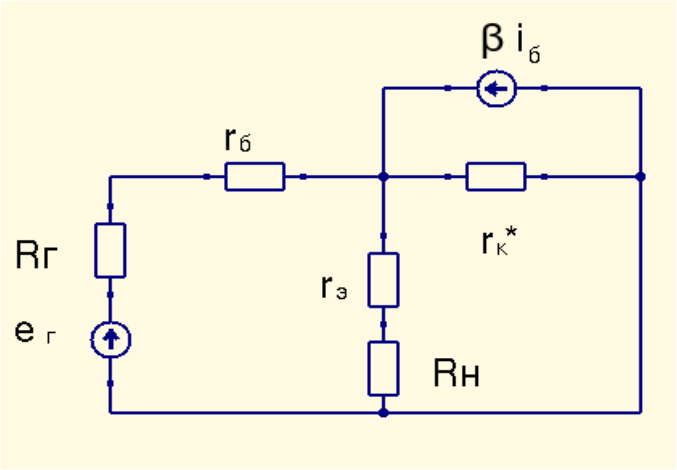


Рис. 14.12. Эквивалентная схема включения транзистора с общим коллектором

Для схемы с общим коллектором (эмиттерный повторитель):

Коэффициент усиления по току $Kiк = i_э/i_б = \beta + 1$.

Входное сопротивление $Rвх.к = (u_{б\text{-}э} - u_н)/i_б = (Rвх.б + Rн)(\beta + 1) \sim Rн(\beta + 1)$.

Коэффициент усиления по напряжению $Kuк = u_н/u_{вх} = u_н/i_б Rвх.к \sim 1$.

Эти формулы тоже можно добавить на страницу расчетов, связанных с усилителями.

Программы для более сложных расчетов

Как я выше говорил, для более сложных расчетов в Linux можно использовать программы scilab и octave. Это, конечно, для самых любознательных, тех, кто не боится математики и имеет некоторый опыт работы. Я долго сомневался, следует ли упоминать об этом, но, если с программами для Windows многие знакомы хорошо, то о существовании нужных программ в Linux они иногда не подозревают.

Я давно отвык от работы в терминале, по причине чего использую программу Koctave, имеющую графическую оболочку. К сожалению в моей основной в данный момент операционной системе Fedora 7 нет готового для установки пакета. Но это не беда, есть исходные файлы, которые можно скачать на сайте Koctave, и есть три волшебные команды: *./configure*, *make* и *make install* (последнюю следует использовать с правами *root*). Занимает это вместе с загрузкой исходного текста программы минут пять. В итоге я нахожу исполняемый файл по адресу */usr/local/kde/bin/koctave3*, добавляю на рабочий стол кнопку запуска, и запускаю программу.

Это, как видно на рисунке, собственно программа *octave*, но в графическом наряде. Для нее в Fedora 7 есть ряд приложений, дополнительных библиотек.

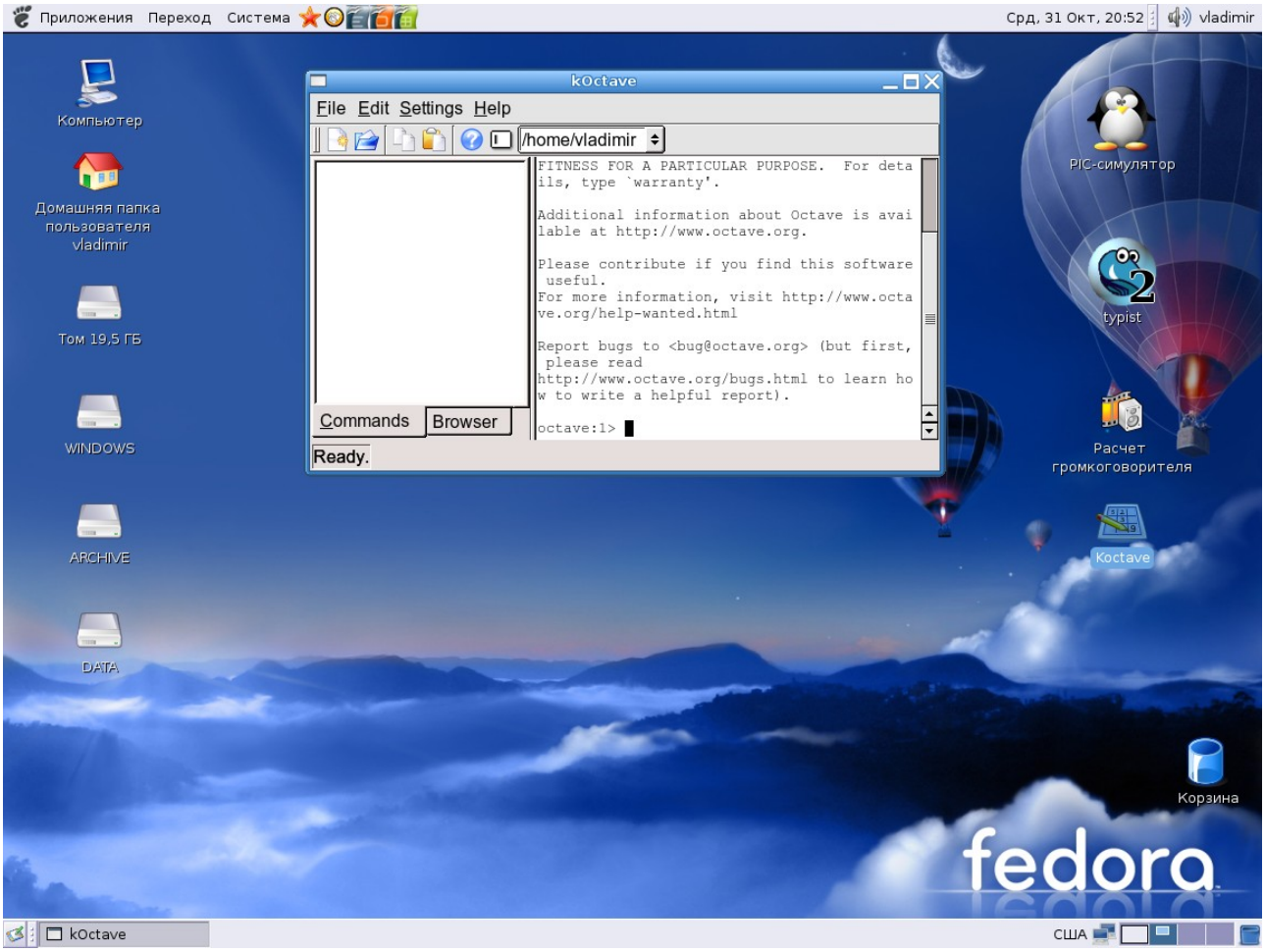


Рис. 14.13. Запуск программы Koctave

Какое отношение математические программы имеют к электронике я хочу пояснить на примере, скажем, метода контурных токов. Этот метод применяют для расчетов, чтобы уменьшить количество уравнений, получаемых для схемы по второму закону Кирхгофа. Использование метода контурных токов связано с матричным представлением, а работать с матрицами лучше в *octave*. Это же относится и к методу узловых потенциалов, или, например, к работе с четырех-полюсниками.

Программа позволяет выполнять вычисления непосредственно – ввел операцию и данные, нажал на клавишу **Enter**, получил ответ.

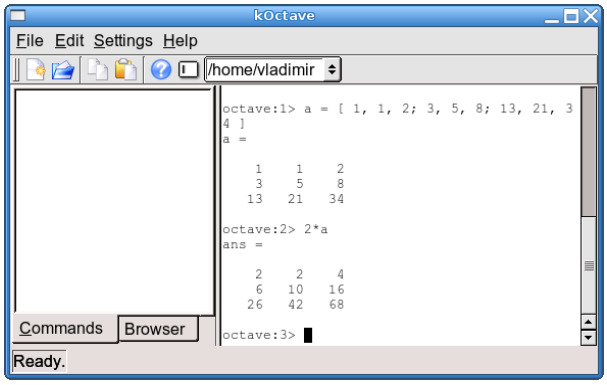


Рис. 14.14. Работа с матрицами в Koctave

Записав матрицу в виде: *a = [1, 1, 2; 3, 5, 8; 13, 21, 34]*, – ее можно «увидеть» в привычной форме, умножить на другую матрицу или на число, просто записав *2*a*. Эти примеры я беру из руководства к *octave*, как и следующий пример решения дифференциального уравнения вида: $dx/dt = f(x, t)$, $x(t=t_0) = x_0$. Этот пример лучше записать в файл, например, используя встроенный редактор, хотя это текстовый файл:

```
function xdot = f (x, t)
r = 0.25;
k = 1.4;
a = 1.5;
b = 0.16;
c = 0.9;
d = 0.8;
xdot(1) = r*x(1)*(1 - x(1)/k) - a*x(1)*x(2)/(1 + b*x(1));
xdot(2) = c*a*x(1)*x(2)/(1 + b*x(1)) - d*x(2);
endfunction
x0 = [1; 2];
x = lsode ("f", x0, t);
t = linspace (0, 50, 200);
plot (t, x)
```

Последняя строка относится к выводу графики, для которой есть свое окно. А в редакторе есть кнопка выполнения, нажав на которую можно получить результат:

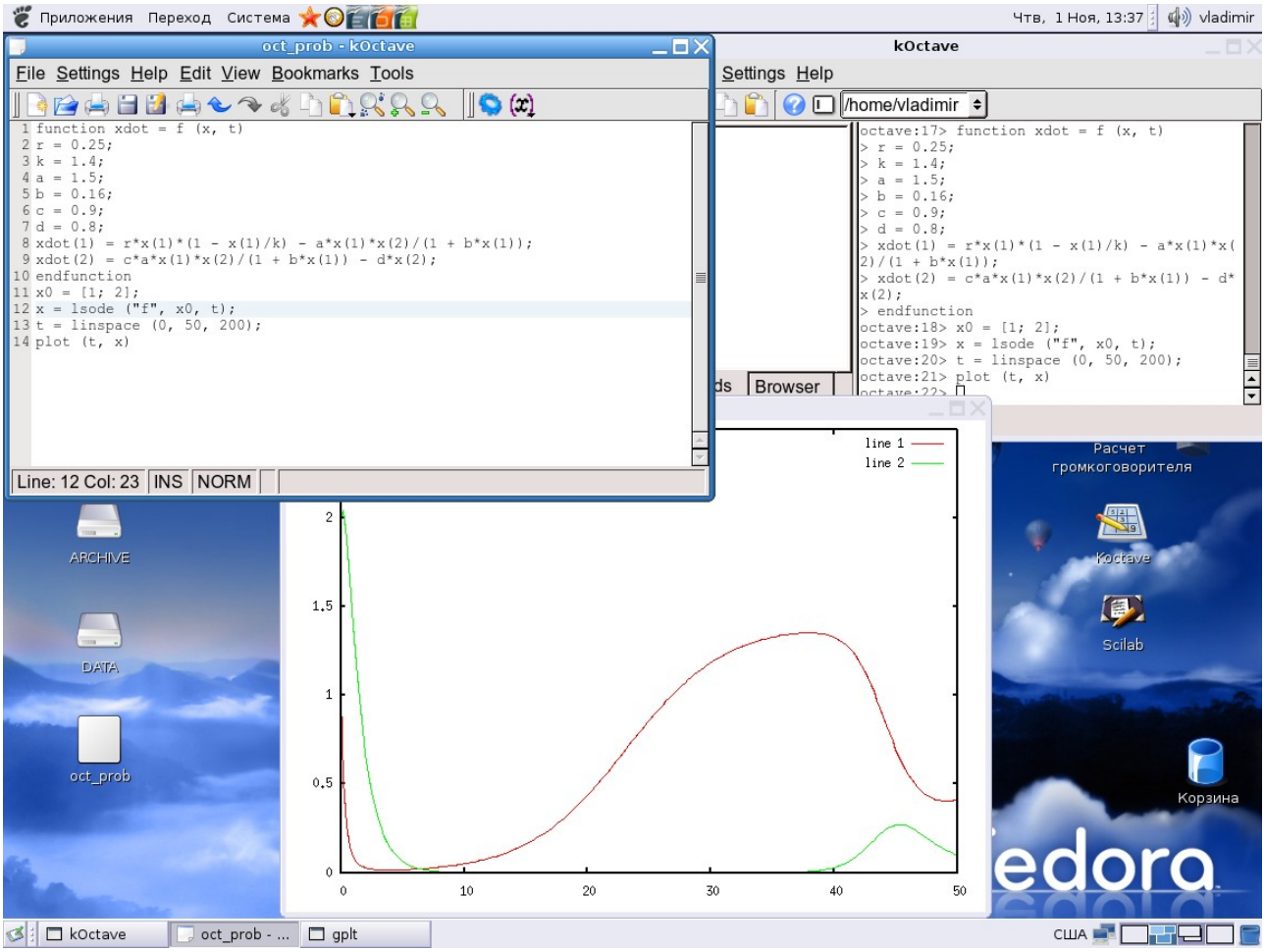


Рис. 14.15. Результат решения уравнения в Koctave

Надеюсь, я не очень напугал начинающих. Вам сегодня это не нужно. Но... завтрашний

день всегда интереснее сегодняшнего. Да и с помощью этого примера я хотел показать, откуда берутся все «картинки» сигналов в программе, например, Qucs. И с помощью этого примера, я хотел показать, что электроника достаточно тесно связана с математикой, и что теперь можно перейти к следующей теме.

Глава 15. Микропроцессор и программирование

Для рассказа о микропроцессорах в программе Qucs нет подходящих простых средств. Есть другие программы, позволяющие работать с процессорами, но тема эта неисчерпаема, а главу я включил только для того, чтобы перейти к рассказу о микроконтроллерах. Именно они, хотя никто не возбраняет вам использовать микропроцессор, но именно микроконтроллеры находят все большее применение в практической работе, становятся сердцем электронного устройства. Микроконтроллер я бы даже назвал микрокомпьютером, если бы не та существенная разница между контроллером и компьютером, что последний предназначен к выполнению множества программ, а первый для выполнения одной программы, заложенной в него. Есть и другие отличия, но о них лучше узнать из книг по микроконтроллерам, микропроцессорам и компьютерам. А в основе всех этих устройств лежит работа микропроцессора. Некогда процессор для компьютера или контроллера выполняли на множестве микросхем (я не беру времена еще более ранние). С появлением технологии «упаковки» всего снаряжения процессора в одну микросхему началась эра «компьютеризации всея Руси», то есть, всего, до чего можно дотянуться, если не очень лениться. Как же выглядит внутренний мир микропроцессора, над которым мы пролетим со скоростью звука.

Галопом по европам

Микропроцессорная техника тесно связана с двумя дисциплинами – электроникой и математикой. Электроника вошла в этот симбиоз через цифровую технику, а математика через программирование. Невозможно отделить одно от другого без того, чтобы не погубить и то, и другое. Никому не нужны программы сами по себе. И если цифровые устройства еще работали в «свободном плавании», то микропроцессорные без программы беспомощны как новорожденные. Какими они вырастут зависит от программы.

Сердце микропроцессора – арифметико-логическое устройство (АЛУ). Как можно сложить два числа с помощью цифровых устройств мы уже знаем. Похожим образом происходит вычитание, умножение и деление, составляя арифметическую область процессора. А тот факт, что все операции производятся над двоичными числами, позволяет рассматривать и логические операции. Даже названия цифровых микросхем во многом происходят уже из логики: И-НЕ, ИЛИ-НЕ. То есть, даже на ранней стадии цифровая техника ориентировалась на применение в качестве основы логических устройств.

Каждый микропроцессор имеет определенный, в настоящее время весьма обширный, набор команд. Собственно все, что он и может, и для чего предназначен, это выполнять эти команды. Команды, это те же двоичные числа, состоящие из последовательности нулей и единиц. Только их понимает процессор, и только они способны вывести его из беспробудной задумчивости. Последовательность таких команд – это суть программы.

Команды процессору могут быть очень простыми и лаконичными, например, «стоп». Получив такую команду процессор прекратит работу. Команды могут быть достаточно сложными, например, переписать содержимое аккумулятора во внешнюю память по адресу такому-то. Чтобы выполнить ее, нужно, по меньшей мере, знать, что такое аккумулятор, что такое внешняя память, и что такое адрес, да еще и что такое переписать. Но это забота внутреннего управляющего устройства в процессоре.

Для его диспетчерской работы ему нужны «бухгалтер» – счетчик команд, «кладовщик» – учетчик адресов и окно выдачи – шина данных...

Это присказка, а сказка впереди.

В двоичном числе команды процессора уже есть информация о свойствах команды, таких как ее сложность. За сложными командами могут располагаться числа данных. Отличить число, выражающее данные, например, слагаемое при сложении, от адреса, например, для запоминания результата сложения, и от самой команды невозможно. Все числа одинаковы. Один и тот же набор нулей и единиц может быть адресом, операндом и командой. Только устройство управления в процессоре может определить, кто есть кто, адрес отправить на адресную шину, данные на шину данных, а команду принять к исполнению, подключив к ее реализации АЛУ.

Я уже несколько раз упомянул слово шина. Это, естественно, не автомобильный термин, а процессорный, так обозначают совокупность выводов процессора, соединенных с другими устройствами вне процессора. У компьютера внешних устройств много: и монитор, и клавиатура, и дисковод, и принтер, и... Много. Чтобы как-то их обозначить им присваивают число – адрес. И все выводы процессора, которые используются для адресации, вместе с проводниками, соединяющими процессор с внешними устройствами, называют адресной шиной. Но чаще всего по этой шине процессор адресуется к памяти. Оперативная память компьютера – место размещения всех программ, хотя для процессора это тоже внешнее устройство. Одновременно оперативная память может быть и местом размещения всех данных: операндов, результатов вычислений, промежуточных результатов и т.д. Для данных есть еще одна шина – шина данных. Количество выводов шины адреса или данных часто кратно байту – восьми битам или единицам информации. Каждый бит может быть нулем или единицей. В сущности, это двоичные числа. Байтовый характер имеют многие регистры – хранители временной информации, временных чисел, еще одно место размещения операндов и место размещения выводимых данных.

Это все еще присказка...

Многие микропроцессоры после включения питающего напряжения выставляют на адресную шину нулевой адрес. Процессор ждет, что по этому адресу будет расположена первая команда. Положим, что мы имеем дело с устройством, где есть процессор, есть оперативная память, клавиатура и дисплей. Процессор начал работу и ждет первой команды, и ждать процессор может долго. Оперативная память пуста. Прежде, чем он сможет что-то начать делать, в эту оперативную память нужно записать команды. И не с клавиатуры. Клавиатура тоже достаточно сложное внешнее устройство, его работу тоже нужно каким-то образом организовать. В стародавние времена в оперативную память записывали числа (команды, данные) по битам. Позже стали применять ПЗУ (постоянное запоминающее устройство) – энергонезависимую память, то есть, такую, которая сохраняет записанную в нее информацию (двоичные числа) и после выключения питания. В нее можно записать хотя бы программу, которая будет обслуживать клавиатуру и выводить на монитор то, что будет набрано с клавиатуры. Кстати, до пары клавиатура-монитор использовали приспособленную к этому пишущую машинку. Печатаешь на машинке, результат отправляется к процессору и ответ печатается на бумажной ленте, заправленной в пишущую машинку. И в те времена обслуживанием пишущей машинки занимались контроллеры, выполненные на цифровых микросхемах.

Имея пополненный ПЗУ арсенал, можно заставить процессор что-то сделать. Например, после включения питания процессор обращается к ПЗУ, где первая команда начинает программу, которая будет обслуживать последующий ввод команд. Теперь можно с клавиатуры (пишущей машинки) отправить первую команду, затем вторую и т.д. К слову, каждое нажатие клавиши на клавиатуре отправляет число. Тогда уже, как мне кажется, придумали использовать не двоичную, а восьмеричную и шестнадцатеричную систему счисления для чисел (команд и данных). Гораздо проще ввести, скажем, команду «С3», чем заполнить это нулями и единицами. Команды и данные, собранные и упорядоченные,

образуют программу, существующую в виде чисел.

Н даже устроившись с пишущей машинкой или клавиатурой, вводить программу каждый раз, когда нужно что-то сделать, оказалось слишком утомительно, расточительно, а, главное, подвержено часто ошибкам. Тогда все, что нужно ввести в оперативную память, стали записывать. Вначале на специальные бумажные карточки, затем на специальную бумажную ленту, еще позже на магнитофонную ленту. Включаете магнитофон, и микропроцессор слушает свою музыку «нулей и единиц». Зтем на смену магнитной ленте пришли магнитные «грампластинки» – дискеты и жесткие диски. Теперь программу стало легко читать в оперативную память, легко ткнуть процессор в нужное место, чтобы он начинал работать. И дисководы, и оперативная память, и контроллер клавиатуры, и много еще чего – все они подключаются к адресной шине и шине данных процессора, а управление, я бы сказал, идет по шине управления, где собраны все управляющие сигналы – всеми устройствами нужно управлять. Процессор стал управлять и дисководами, и принтером, и монитором, а им, ведь, всем нужно все растолковать, для этого существуют управляющие сигналы.

И это тоже присказка, но пора переходить к сказке.

Как организована работа процессора

В общих чертах я рассказал, как организована работа процессора. Рассмотрим некоторые частности. Например, я упомянул, что в оперативную память записываются двоичные числа. А как? Мы знаем, что есть триггер, имеющий два состояния, отображаемые на его выходе. Сам триггер мы можем установить либо в одно состояние, когда на его выходе «0», либо в другое, когда на выходе «1». Значит, один бит мы можем записать в триггер или запомнить в триггере. А восемь D-триггеров, если мы подключим их входы данных к шине данных, а выводы управления записью соединим вместе и подключим к управляющему сигналу записи, то они смогут запоминать байт. Похожим образом организовано то, что я называл регистром.

Конечно существующие микросхемы оперативной памяти устроены несколько иначе, но и в их основе лежит управляемый сигналом триггер, имеющий два состояния. Это может быть, например, полевой транзистор, на затвор которого подается заряд, после чего затвор отключается от цепи, а транзистор сохраняет свое состояние. Если заряд с затвора снять, то транзистор перейдет в другое устойчивое состояние. Вот и получается «камера хранения» для чисел. Есть еще одна деталь. Записать мы записали. А прочитать, я имею ввиду устройство из D-триггеров. У них все, что записывается, сразу появляется на выходе. Но шина данных у нас одна, на нее мы выводим данные для записи, на нее же должны вывести данные при чтении. D-триггер сразу, как только мы его подключим к шине данных, займет шину, установив на ней записанное в него число, и не важно, будет ли это нуль или другое число. Шина окажется занята. Чтобы этого не происходило, устройства оперативной памяти, да и другие устройства, работающие с шиной, устроены так, что они имеют третье состояние. Когда на их входе разрешения вывода устанавливается активное состояние, они выводят на шину записанное в них число, когда сигнал снимается, их выход переходит в состояние с высоким сопротивлением.

Сейчас я попробую изобразить, то, о чем пытался сейчас рассказать, с помощью двух транзисторов и программы Qucs. Выход ячейки памяти, присоединенной к шине данных, обратите внимание, будет очень похож на выходной каскад усилителя мощности.

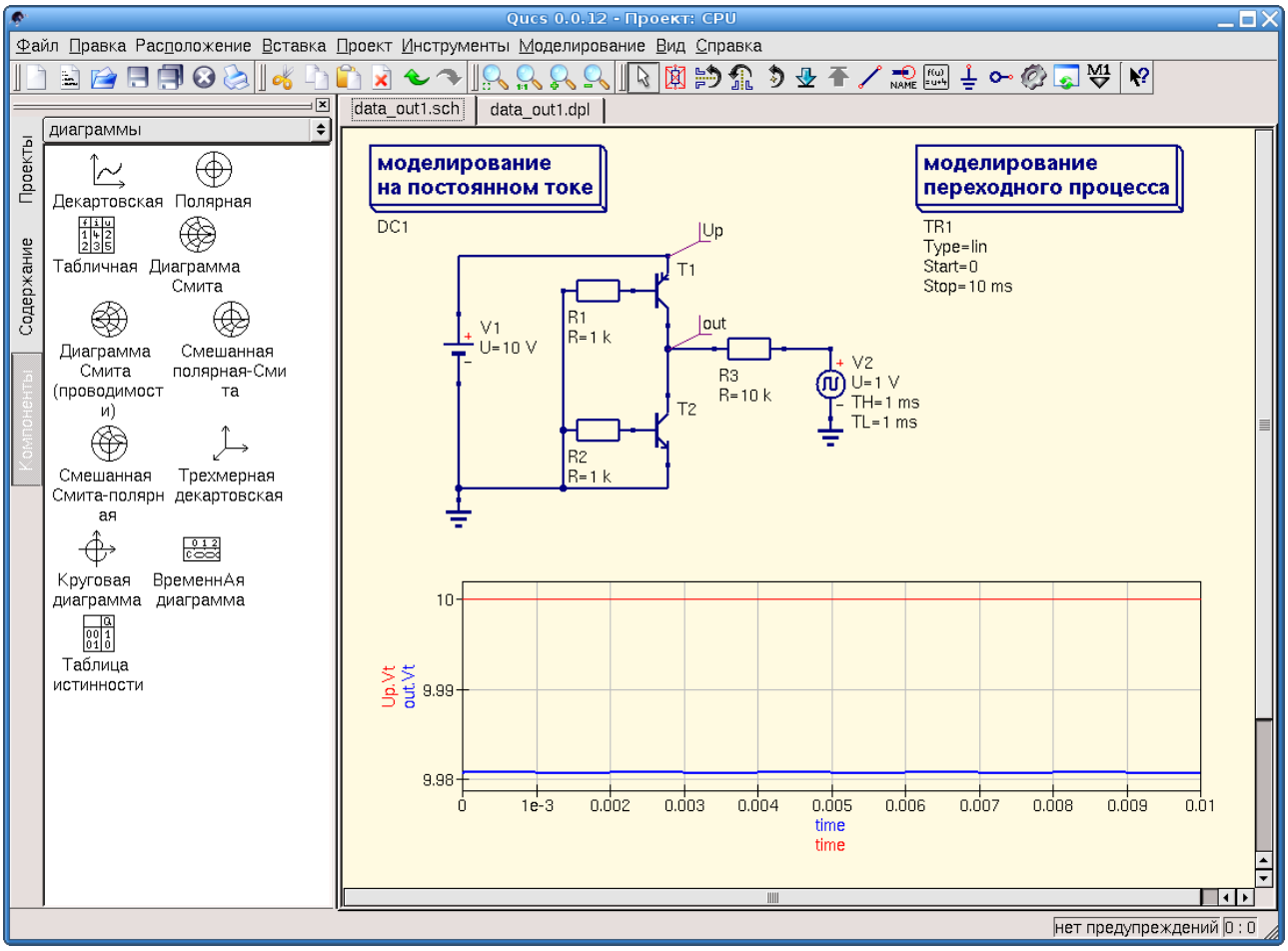


Рис. 15.1. Выход в состоянии логической «1»

Заметьте, что сигнал (нижняя кривая) находится в состоянии близком к напряжению питания (верхняя кривая). Это состояние логической единицы может быть прочитано при обращении к этому биту.

Если теперь мы выключим транзистор T1 и включим транзистор T2, то на выходе получится логический ноль. Для этого точку соединения резисторов «оторвем» от земли и подключим к плюсу питающего напряжения. Диаграмма приобретет вид:

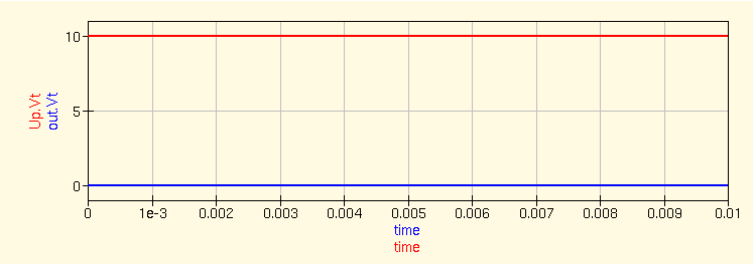


Рис. 15.2. Диаграмма состояния логического нуля

Но если мы выключим оба транзистора, как показано на рисунке ниже, сигналы от генератора V2 (назовем его генератором чисел на шине данных), появятся на шине данных.

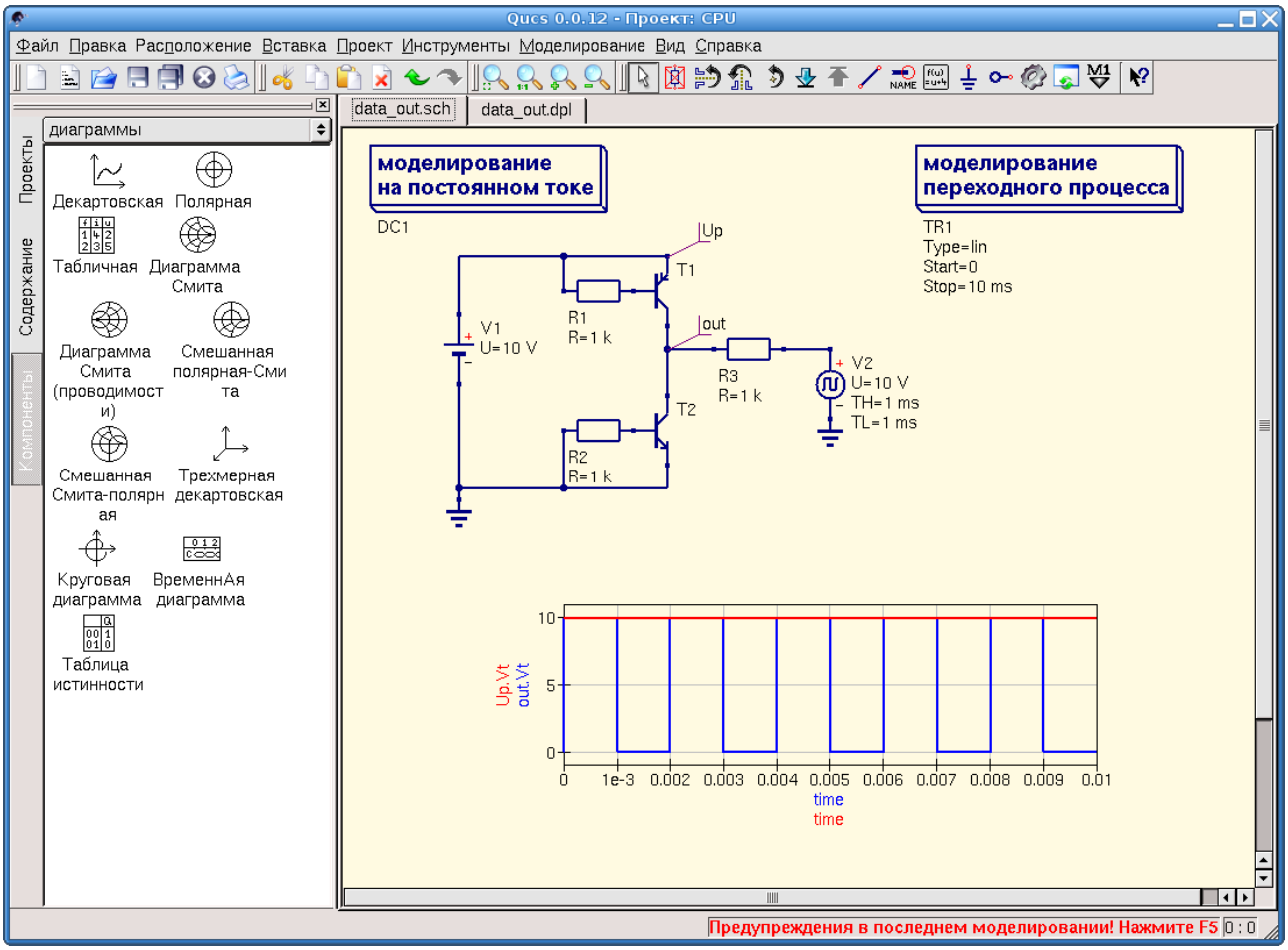


Рис. 15.3. Выход в третьем состоянии

На диаграмме видно, что шина свободна, на ней можно устанавливать данные.

Если и не совсем так, но похожим образом, реализуется выход с третьим состоянием у всех устройств, подключенных к одной шине данных.

С этой деталью мы, будем считать, разобрались.

Посмотрим теперь, как может быть организована реакция устройства на обращение к нему по адресу. Для определенности пусть устройством будет ячейка памяти. Конечно реальные устройства памяти имеют не одну ячейку даже организованную в восемь бит, а множество таких ячеек. Но каждая ячейка будет иметь уникальный адрес. Опять-таки, для определенности положим, что наша адресная шина четырех-битовая, нет нужды загромождать рисунок. Нас интересует, чтобы только одна комбинация логических единиц и нулей на входе, из всех возможных комбинаций, давала реакцию на выходе.

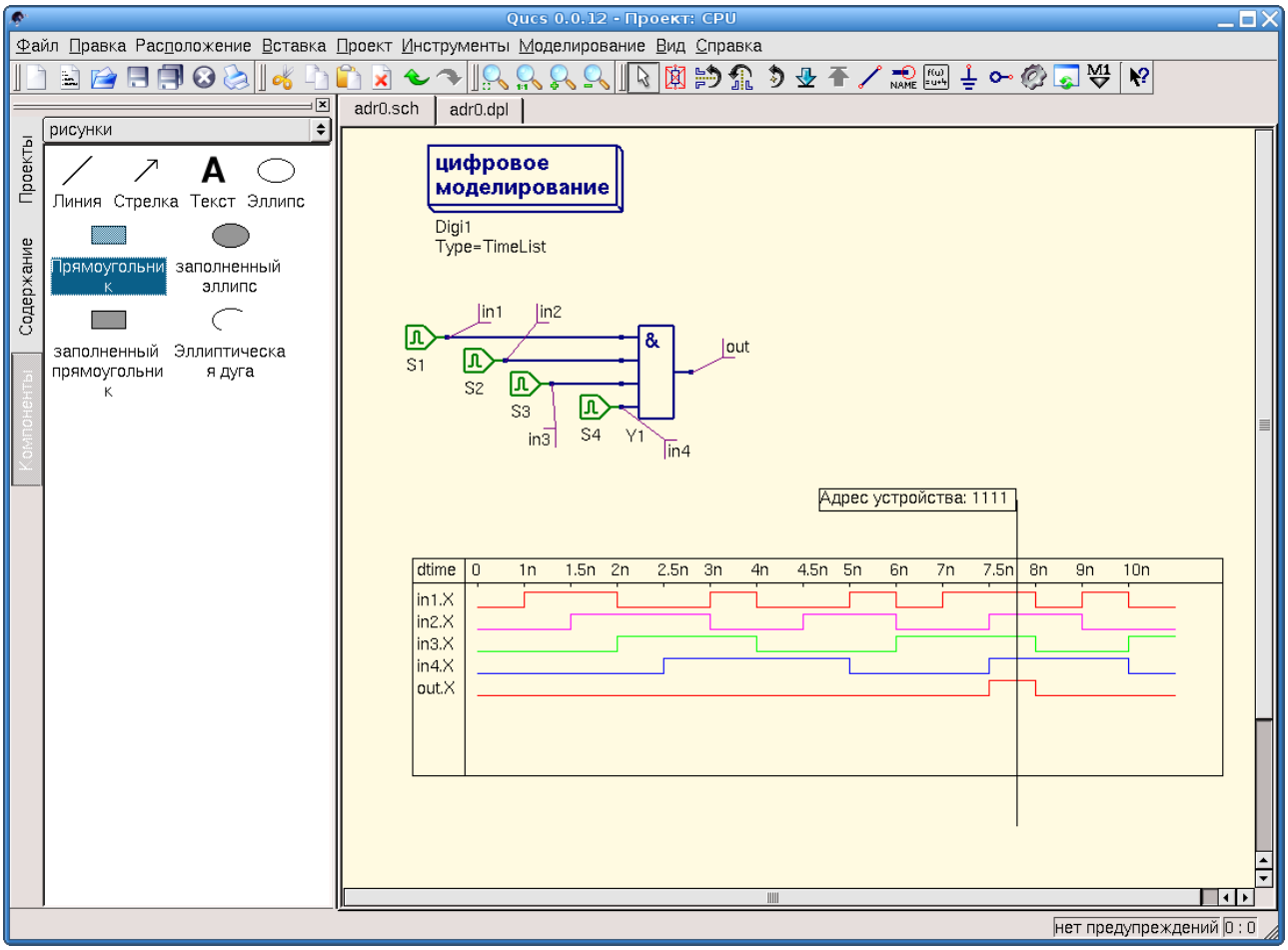


Рис. 15.4. Адресация с помощью многовходовой схемы И

Но это только один адрес. А как можно получить, например, второй? Поставим инвертор между входом схемы «И» и первым источником сигнала S1. Диаграмма этого процесса изменится.

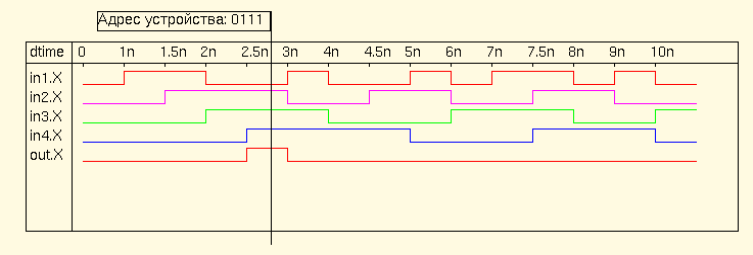


Рис. 15.5. Диаграмма изменения адреса с помощью инвертора

Так с помощью многовходовой микросхемы «И» и инверторов мы можем получить то, что называется адресный селектор. Конечно, есть специальные микросхемы адресных селекторов, но существа дела это не меняет.

Если к нашему адресному селектору мы добавим ячейку памяти из нескольких D-триггеров, то получим еще и запоминающее устройство.

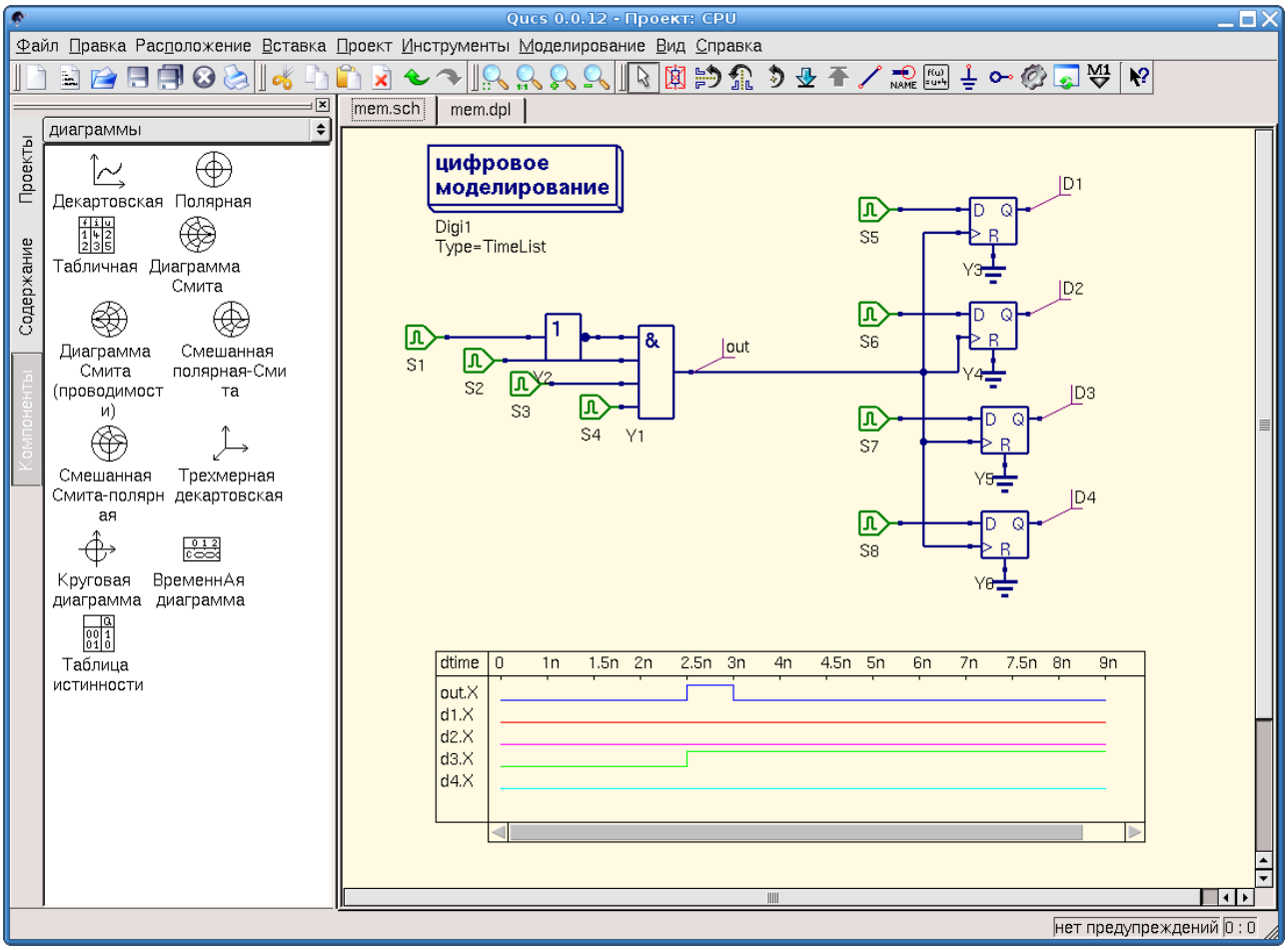


Рис. 15.6. Адресуемая ячейка памяти

Источники S1-S4 на рисунке представляют нашу адресную шину, а S5-S8 – шину данных, а соединение, помеченной меткой *out*, образует шину управления. После появления на адресной шине адреса нашей ячейки памяти, когда сигнал *out* переходит в состояние единицы (верхняя кривая), состояние шины данных записывается в нашу импровизированную память, которая в начальный момент времени имела записанное число 0000, а после прихода сигнала выбора приняла число 0010, десятичное 2.

«Трудовые будни» микропроцессора, едва на него подается питание, начинаются с чтения команды из первой ячейки памяти. Как я уже говорил, команды могут занимать только одну ячейку памяти, но могут быть дополнены одной или двумя следующими ячейками, в которых располагаются данные. В соответствии с характером команды внутренний счетчик, а что такое цифровой счетчик мы уже знаем, изменяет свое состояние, отсчитывая правильное количество ячеек памяти, чтобы после выполнения команды адрес, задаваемый процессором, приходился на следующую команду.

Для выполнения операций в микропроцессорах используют еще и внутренние ячейки памяти, которые называются внутренними регистрами. Регистр, повторюсь, можно себе представить таким же, как наша ячейка памяти. Регистры могут быть полностью равноправными, но могут быть выделенные регистры, таков, например, аккумулятор – регистр, в который записываются результаты операций. С точки зрения процессора его рутинная работа по сложению чисел выглядит, примерно, так...

- Устанавливается на выводах адреса адресной шины нулевой адрес.
- Прочитывается число, записанное по этому адресу, в устройство управления.

- Устройство управления устанавливает счетчик команд в соответствии с командой.
- Из ячейки, следующей за командой, устройство управления считывает число и записывает его в первый регистр процессора.
- Из ячейки, следующей за этой, устройство управления считывает число и записывает его во второй регистр процессора.
- Устройство управления дает команду АЛУ процессора – сложить.
- Результат запоминается в аккумуляторе процессора.
- Счетчик команд изменился на 3.
- Адрес, задаваемый на шине адреса, изменился на 3.
- Прочитывается следующая команда.

Ориентируясь на такой порядок работы, можно представить структуру микропроцессора в следующем виде:

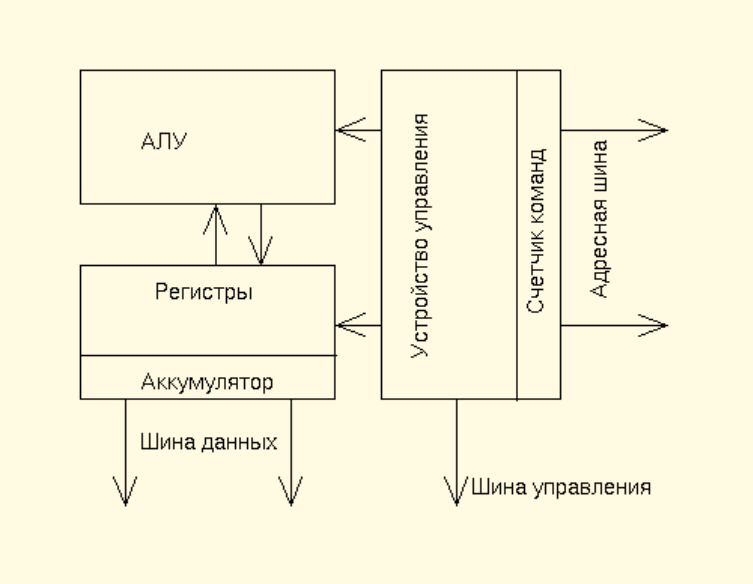


Рис. 15.7. Простейшая структура микропроцессора

Все устроено сложнее, чем нарисовано, но не думаю, что вы намерены заняться разработкой микропроцессоров, а мне не хотелось использовать больше понятий, чем те, с которыми мы познакомились. И, несмотря на существенное упрощение, структура близка к реальной.

Теперь, поковырявшись во внутренностях микропроцессора, попробуем разобраться, как устроен его «духовный мир», то есть, получить некоторое представление о том, что такое программа.

Что такое программа?

А мы уже говорили об этом. Последовательность команд, которые должен выполнить процессор для реализации программы, записанная в ячейках оперативной памяти, и есть программа. Самая простая структура программы – это линейная. Именно, все команды и данные записаны в ячейках памяти последовательно, одна за другой, и последовательно же прочитываются и выполняются.

С точки зрения процессора вся программа это последовательность двоичных чисел. Нам удобнее видеть эту последовательность, например, в шестнадцатеричном виде – так легче ее

воспринимать. Но еще легче нам воспринимать программу, если каждому числу, обозначающему команду, мы сопоставим некоторое буквенное обозначение. Желательно, чтобы это обозначение было сокращением привычных для нас слов. Вот, если говорить о предыдущем фрагменте работы процессора, как могла бы выглядеть команда записи числа из ячейки памяти в регистр:

Переместить число по адресу A3 в регистр 0.

Мы можем ей сопоставить, укоротив ее, такой вариант ПРМ A3, R0. В основном используется англоязычная аббревиатура, по причине чего команда выглядит так: MOV a3, R0. Здесь MOV – это сокращение от английского *move*. Согласитесь, такую команду легче запомнить, чем безликую 48 или, того хуже, 1001000.

Так программисты некогда перешли от программирования в машинных кодах к программированию на языке ассемблера. Чтобы программа, понятная программисту, была понятна и процессору, ее переводят на машинный язык. Для этого создается транслятор с языка программирования в машинные коды. А занимается переводом сам процессор. В настоящее время существует очень большое количество языков программирования. У каждого из них есть свое назначение, своя область применения, где язык наиболее эффективен. Кроме ассемблеров были созданы языки более высокого уровня, легче понимаемые программистом, а, значит, он быстрее может написать программу, программа будет содержать меньше ошибок, и можно писать программы, которые будут упрощать и общение с микропроцессором, и написание новых программ.

Применительно к радиолюбительским интересам ни микропроцессоры, ни программирование не являются обязательной составляющей, но посмотрите на современные устройства и скажите себе: «Я не хочу иметь с ними дело». А что в остатке?

Вернемся к программированию. Линейные программы получаются далеко не всегда даже в простых случаях. Практически в каждой программе возникает необходимость проверить некоторое условие, а по результатам проверки выполнить те или иные действия. Возникает необходимость в ветвлении программы. Обычно это во многих языках программирования выполняется операторами *If... then... else (Если... то... иначе)*. Подобные команды высокого уровня разбиваются на ряд машинных команд, но суть остается – есть необходимость и возможность выполнить разные действия в разных ситуациях.

Принято программы записывать в виде алгоритмов – структурных схем программы – перед написанием кода программы. Алгоритм примера приведенного выше выглядел бы похожим на такой:

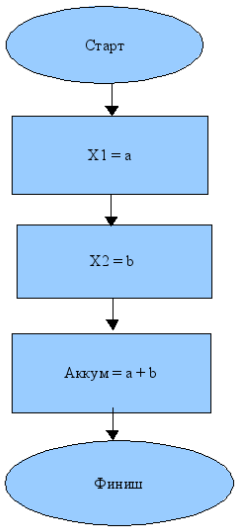


Рис. 15.8. Алгоритм программы сложения

Схема программы с проверкой условий может выглядеть похожей на следующую.

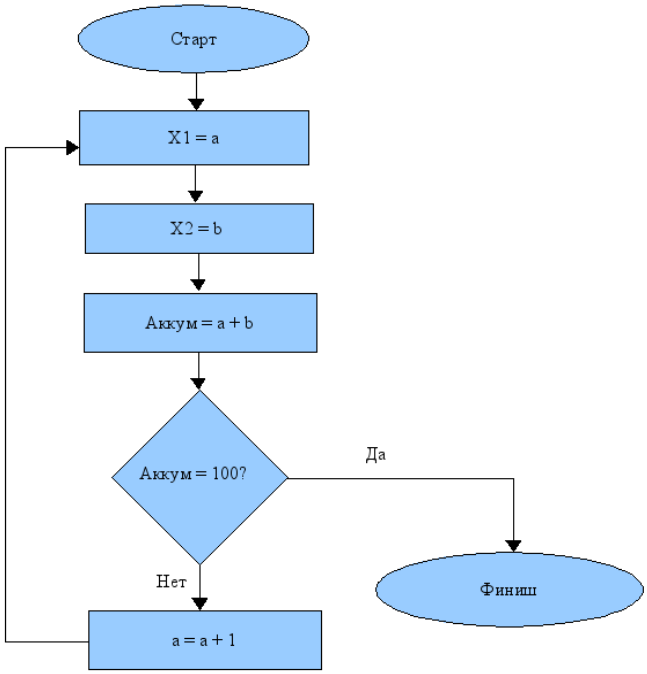


Рис. 15.9. Запись алгоритма при ветвлении программы

Какую бы предварительную форму записи программы вы не выбрали, вам следует описать программу, как ряд действий с достаточной детальностью. После этого можно приступать к написанию кода программы. Поскольку все, что я хотел рассказать о процессорах и программах, я сделал «в прицеле» к микроконтроллерам, я не буду детально описывать ни один из языков программирования, а только упомяну о том, что есть во всех, практически, языках.

Очень часто возникает необходимость многократно повторять один и тот же набор операций. Если вы точно знаете, когда нужно остановиться, вы можете использовать *цикл*. Это монотонно повторяющееся действие, которое заканчивается либо тогда, когда

выполняется условие, например, переменная принимает определенное вами значение, либо выполняется определенное количество раз, если вы знаете, сколько раз следует повторить операции.

Еще более общий смысл вкладывается в повторяющийся набор операций, когда вы объявляете его подпрограммой – частью программы, которая используется многократно. В частности, в любом языке высокого уровня есть такие подпрограммы в виде процедур или функций. В эти подпрограммы можно из основной программы передать некоторые значения, которые процедура или функция возвращает основной программе после выполнения всех предписанных действий.

Организация программы с применением функций дает возможность не только многократно использовать эти функции в программе, но создавать библиотеки функций, а в программе просто ссылаться на эти библиотеки, используя уже готовые функции.

Чаще других при работе с микроконтроллерами используют ассемблер и язык программирования «С». Поскольку каждый микроконтроллер имеет свой набор команд, каждый из них будет иметь свой ассемблер, а переводчик (транслятор, компилятор) с языка «С» должен иметь возможность переводить на язык именно этого ассемблера.

Получается, сколько контроллеров, столько ассемблеров. Собственно, ассемблер один, но разные записи команд ассемблера и разные трансляторы в машинные коды. Если вы решите перейти от работы с одним контроллером к другому, вам понадобится знакомство с его набором команд, но его ассемблер вы освоите быстрее, чем тот, первый.

Есть еще один нюанс в работе с микропроцессорами, касающийся их «физического» воплощения. В рассказе об осциллографе я, кажется, обещал об этом вспомнить, и время пришло. Имея в своем распоряжении осциллограф, бывает обидно, если не удается посмотреть, вернее увидеть, сигналы на шинах микропроцессорных схем. Проблема в первую очередь в том, что сигналы эти, как правило, не периодические. Возьмем схему на рисунке 15.6 в качестве «опорной». Очень хотелось бы увидеть сигналы на выходе после их записи. В изображенной схеме все просто, но если это реальная микросхема памяти, то данные в нее записываются часто, сигналы мелькают с такой скоростью, что на экране осциллографа рябит от событий.

В таких случаях осциллограф переключается в ждущий режим, в качестве сигнала запуска используется (на рисунке) *out*, и просматриваются выходы. При условии, что можно в машинных кодах написать, зациклив ее, программу записи в ячейку памяти. В реальной микросхеме это может быть сигнал CS (выбор микросхемы) или сигнал записи. Если микропроцессор позволяет это сделать, можно снизить тактовую частоту до приемлемого значения.

Вспомнил я это не только потому, что обещал, но и по другой причине: в микроконтроллере с наблюдениями еще хуже. В этом виновата, однако, их сущность. Так какова она?

Глава 16. Микроконтроллеры

Микроконтроллеры во многом похожи на микропроцессоры. Основная разница в том, что микропроцессор предназначен для работы с множеством программ, а микроконтроллер придуман для работы с единственной. Эту программу можно, конечно, заменить, перепрограммировав контроллер, но это внешний процесс, тогда как микропроцессор сам прочитает новую программу (по вашей команде), положим, с дискеты, и может начать с ней работать.

Еще одно отличие контроллера от процессора в том, что в контроллере есть процессор, но кроме него есть и некоторое количество оперативной памяти для размещения программы, есть ряд полезных устройств, таких как аналого-цифровые преобразователи или последовательный порт ввода-вывода, и, пожалуй, еще в том, что процессор в контроллере огорожен (и укрыт от внешнего мира) портами ввода-вывода. В этом смысле микроконтроллер ближе к микрокомпьютеру.

Развитие микроконтроллеров шло параллельно с созданием новых микропроцессоров, с появлением новых технологий изготовления микросхем и развитием вычислительной техники, которая требовала все новых внешних устройств для работы с компьютером. Когда-то мышка, без которой сегодня трудно обходиться, была экзотическим устройством. Дисководы для общения с процессором требовали громоздкого контроллера, внушительные платы с многочисленными цифровыми микросхемами, звуковым устройствам нужен был свой контроллер, монитору свой. Мне кажется, что это постоянное расширение числа внешних устройств, с одной стороны, и удешевление процессоров, с другой, заставили разработчиков микросхем унифицировать подход к понятию контроллера и разработать микросхему, которая называется микроконтроллер.

Что нужно для работы с микроконтроллером?

В первую очередь, и это главное, желание. Освоить азы программирования вы сегодня можете великолепным образом за компьютером. Есть доступные, в частности бесплатные, среды программирования. Для Linux есть удобная среда программирования под названием *Gambas* – это и язык программирования на основе Basic, и возможности быстрого создания графической части программы, и возможность работать со множеством компьютерных устройств, не задумываясь о деталях этой работы. Аналог этой программы для Windows – Visual Basic. Хотя язык Gambas, в отличие от Visual Basic, создавался сразу как современный объектно-ориентированный язык.

Освоить азы программирования легче именно в такой среде программирования за компьютером. Это, правда, не означает, что, создав несколько программ, вы сразу готовы программировать микроконтроллер. Для работы с контроллерами есть свои программы. А выбор этой программы сильно зависит от выбора микроконтроллера. Есть программы поддерживающие широкий спектр контроллеров, но, думаю, нет абсолютно универсальной программы.

Для работы с микроконтроллерами серии PIC мне очень понравилась программа *MPLAB* для Windows. В Linux есть свой вариант MPLAB – это, например, *Piklab*. Последняя требует симулятора *gpsim* для отладки программы.

Кроме программных средств написания и отладки программы для микроконтроллера нужен программатор. Его можно купить, есть доступные по цене варианты, но можно сделать самому. Я в предыдущих книгах описывал несколько вариантов программаторов и достаточно подробно, с моей точки зрения, описывал создание своих программ и в MPLAB,

и в Piklab. Не буду останавливаться на этом, но схему программатора для Piklab приведу, чтобы не было нужды искать ее.

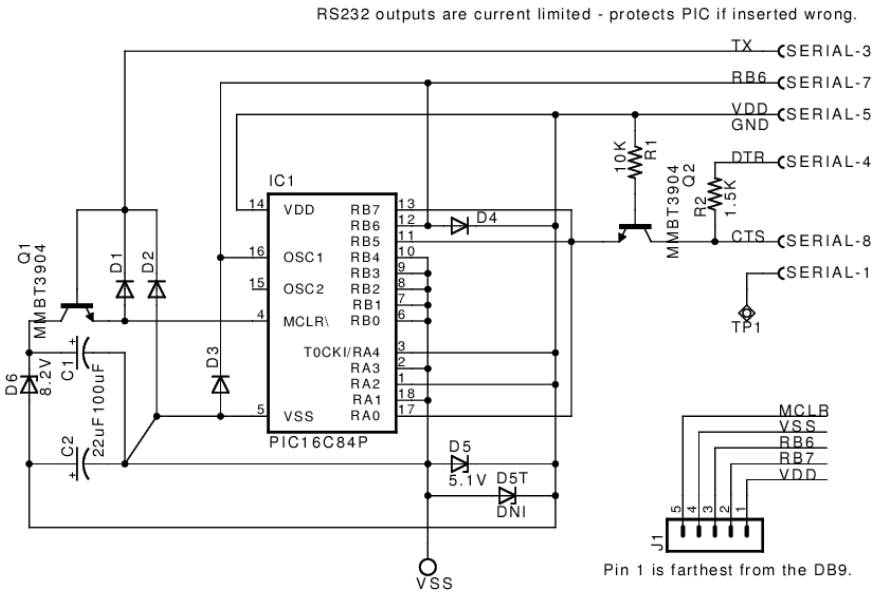


Рис. 16.1. Схема программатора для PIC16F628A

Схема достаточно проста, и это хорошая практика для начинающих, особенно если вы еще плохо умеете паять, собирать такие схемы. Трудность, правда, в проверке и налаживании этой схемы – она предназначена для подключения к COM-порту и работает с компьютером. Но, если все собрано правильно, схема работает сразу. Вообще, собирать свои приборы для радиолюбителей обычная практика, приборов всегда не хватает, а опыт, приобретаемый при сборке и наладке приборов, обязательно пригодится в последствии. Измерениям будет посвящена следующая глава, а сейчас вернемся к программам для написания, отлаживания и загрузки программы в микроконтроллер.

Программа MPLAB имеет более развитую часть, связанную с симуляцией работы осциллографа, чем симулятор gpsim. Но и последний имеет такую интересную особенность, как симуляция работы макетной платы. Кроме того, в программе Piklab хорошо работает свободный компилятор языка «С», полнообъемный и доступный. В несложных случаях для контроллера PIC16F628A особых ошибок у него в последней версии я не заметил.

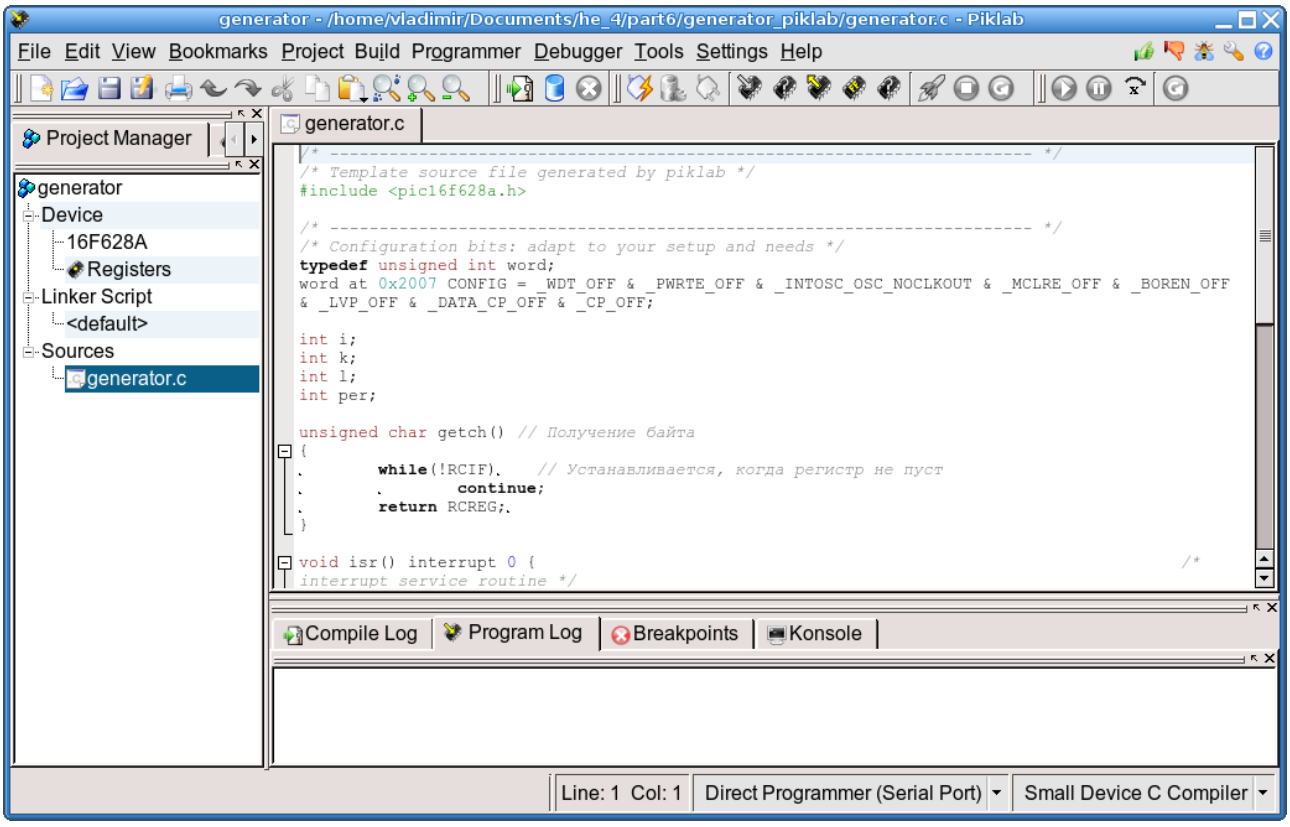


Рис. 16.2. Программа Piklab в Linux

Для работы с ассемблером конкретного микроконтроллера дополнительные средства не потребуются, достаточно выбрать тип контроллера, но если есть желание, а это гораздо удобнее, использовать язык «С», понадобится компилятор PICC Compiler (SDCC). На рисунке во встроенном тестовом редакторе программа на языке «С».

В настройках программы можно (и нужно) установить конкретный программатор (выше приведена схема JDM) и используемый транслятор. Работаете ли вы на языке высокого уровня, работаете ли на ассемблере, вам понадобится документация с описанием конкретной модели микроконтроллера. Для контроллеров серии PIC производства MicroChip на сайте российского представительства можно найти описания на русском языке.

В описании не только рассказано об устройстве контроллера, но приведены все сведения, включая полный набор команд, необходимые для программирования контроллера.

Вот, пожалуй, все для начинающего, что ему нужно для начала работы с микроконтроллерами. А теперь вернемся к незавершенным проектам.

Завершение проекта «Светофор»

Я много раз сетовал на то, что «я бы... если бы...» и вспоминал микроконтроллер. Больше не на что пенять, говорю я себе, загружай Piklab или MPLAB, покажи, как надо делать.

Иногда я жульничаю. Никто этого не замечает, а меня избавляет от неприятных объяснений. На этот раз я тоже сжульничаю, но объясню, почему я решил это сделать.

Для того, кто только начал осваивать электронику, микроконтроллер не цель, ради которой можно и программирование освоить, это одно из средств решения своих задач или работы в сфере своих интересов. На первом этапе я бы посоветовал, хотя позже очень полезно будет

вернуться к полным схемам работы — с ассемблером, языком «С» и чтением документации — я бы посоветовал для начала использовать программу KTechlab. Программа работает в среде Linux, и я не знаю есть ли ее аналог для Windows. Тем не менее, написание небольшой программы для микроконтроллера доставит вам удовольствие своей простотой.

Иногда в подобных случаях опытные электронщики ругают и подобные советы, я их понимаю, и подобные программы, здесь я не согласен, но это их личное дело, их собственное мнение. Если вы хотите присоединиться к их мнению, удалите эту часть книги, да и дело с концом. Вот такое у меня МНЕНИЕ.

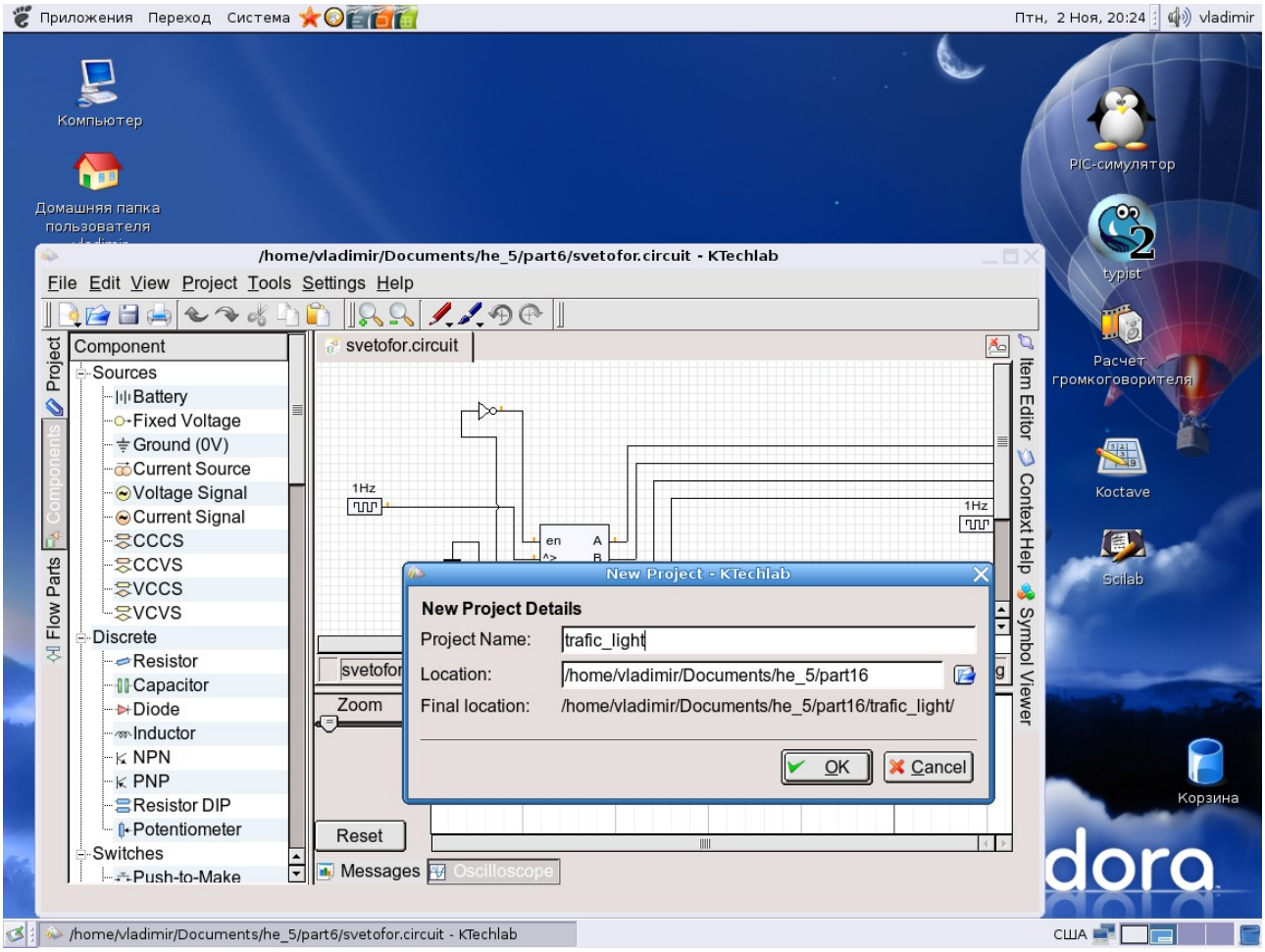


Рис. 16.3. Программа KTechlab

Как и многие среды работы над чем-то, программа позволяет работать с проектом, и после ее запуска есть смысл в пункте меню *Project* использовать раздел *New project*, открывающий окно диалога создания проекта, как это сделано на рисунке.

Выбрав имя проекта и место его расположения, с помощью основного меню (пункт *File*) или с помощью кнопки инструментального меню создайте новый файл. Вы сразу можете убедиться, что файлы в этой программе бывают разные.

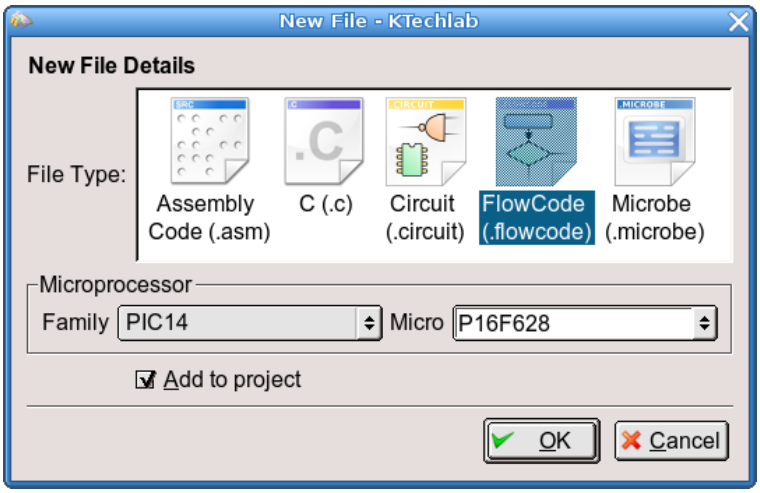


Рис. 16.4. Диалоговое окно создания нового файла

Для обычной работы со схемой есть возможность выбрать *Circuit*, для написания программ на ассемблере (я так думаю) *Assembly* и т.д. Здесь же выбор микроконтроллера, и не забудьте проверить флажок *Add to project*, чтобы созданный вами файл вошел в состав проекта. Как видите, я выбрал *FlowCode* — пошел по пути наименьшего сопротивления, не видать мне славы «крутого спеца», зато вы можете и без помощи закона Ома определить величину этого сопротивления.

Напомню, как должен работать наш светофор:

- Две секунды горит зеленый светодиод.
- Зеленый светодиод гаснет, зажигается желтый.
- Желтый светодиод горит одну секунду, затем он гаснет.
- Зажигается красный светодиод и горит две секунды.
- Гаснет красный и на секунду зажигается желтый, после которого зеленый и т.д.

Микроконтроллер имеет встроенные «подтягивающие» резисторы, на некоторых выводах портов их можно использовать, но я пользуюсь старым макетом, где у меня есть резисторы и светодиоды (все красные, но это разве важно?), резисторы включены одним концом к плюсу питающего напряжения, другим к выводу порта. В эту точку приходит анод светодиода. Что предполагает, когда на выводе единица, то светодиод через резистор (его величину вы уже легко подсчитаете, зная напряжение питания микроконтроллера 5 В, и выбрав ток через светодиод 5 мА) подключен к питающему напряжению и светится. А когда на выводе появляется логический ноль, светодиод оказывается «закорочен» этим состоянием вывода и гаснет. Все просто и ясно.

Первым делом (как на картинке ниже) я нажимаю клавишу **Advanced** нарисованного контроллера, поскольку в его верхней части есть примечание (или название) *PIC Settings*, заставляющее меня думать, что это ключик к установкам контроллера.

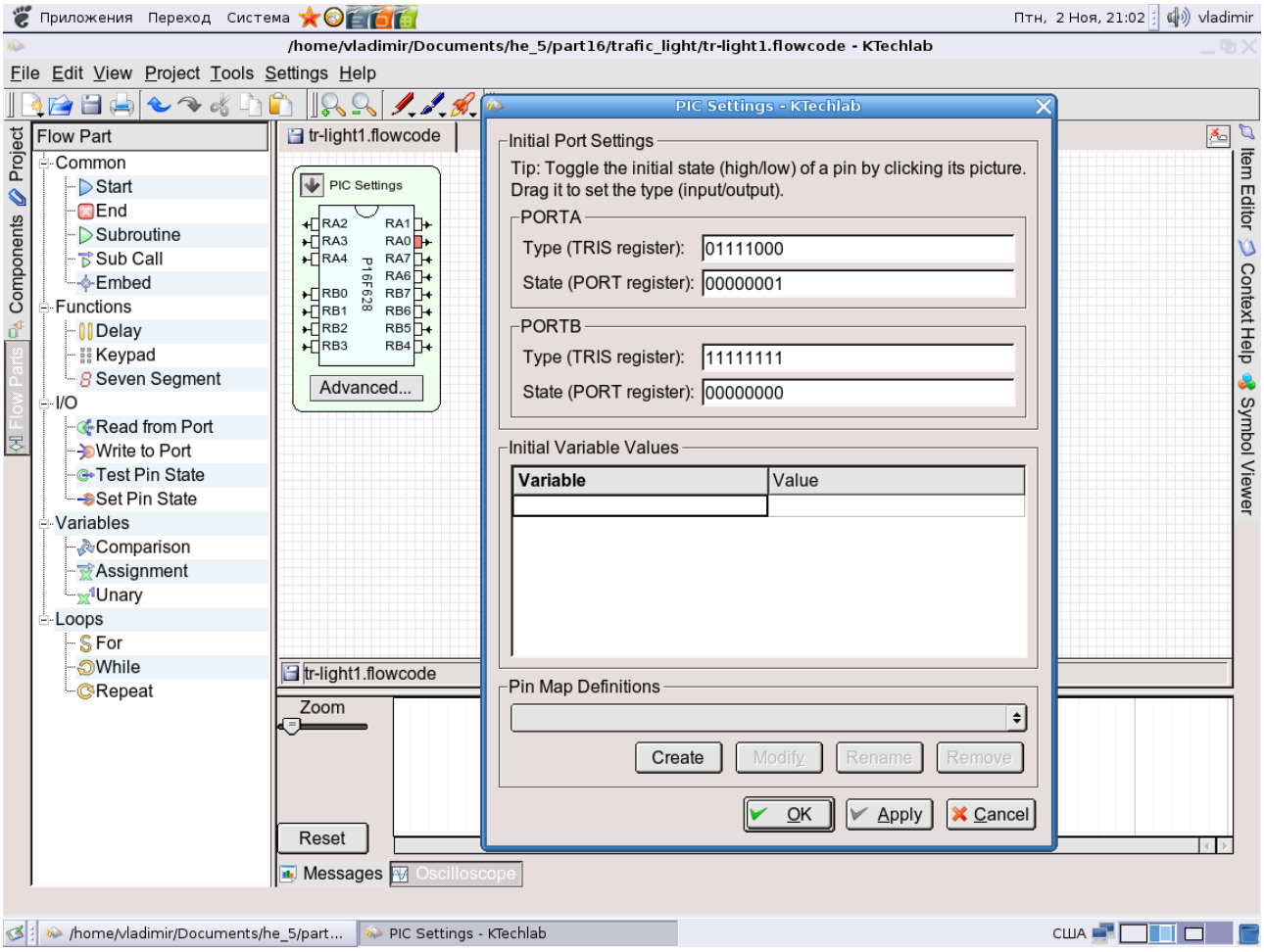


Рис. 16.5. Задание начальных установок микроконтроллера

Сознаюсь, что знаю назначение регистра TRIS (верхнее окошко), где вместо последних трех единиц записываю три нуля. Мало того, в следующем окошке, где были записаны одни нули, я меняю последний нуль на единицу. Затем, как и положено, последовательно нажимаю клавиши **Apply** (принять) и **OK**. Если вы теперь приглядитесь к рисунку микросхемы в верхнем левом углу рабочего поля, то заметите, что стрелки у выводов RA0, RA1 и RA2 изменили свое направление — результат работы первого окошка, эти выводы микроконтроллера будут работать на выход. А вывод RA0 еще и покраснел — результат работы второго окошка, вывод перешел в состояние логической единицы, на нем напряжение питания.

Простая конструкция: три вывода, три светодиода на них. Проще не бывает.

В левом окне интерфейса программы компоненты. Первый из них, который я перенесу на рабочее поле, это *Start* из набора *Common*. Я просто цепляю компонент мышкой и переношу на рабочее поле. Ниже из набора *Functions* я располагаю *Delay*. По умолчанию длительность задержки 1 сек. Мне нужно две. Щелкаю по рисунку этой задержки на рабочем поле левой клавишей мышки, выделяя его, и получаю в верхнем поле, где расположено основное меню, доступ к свойствам этой задержки, из которых я знаю только одно свойство — время.

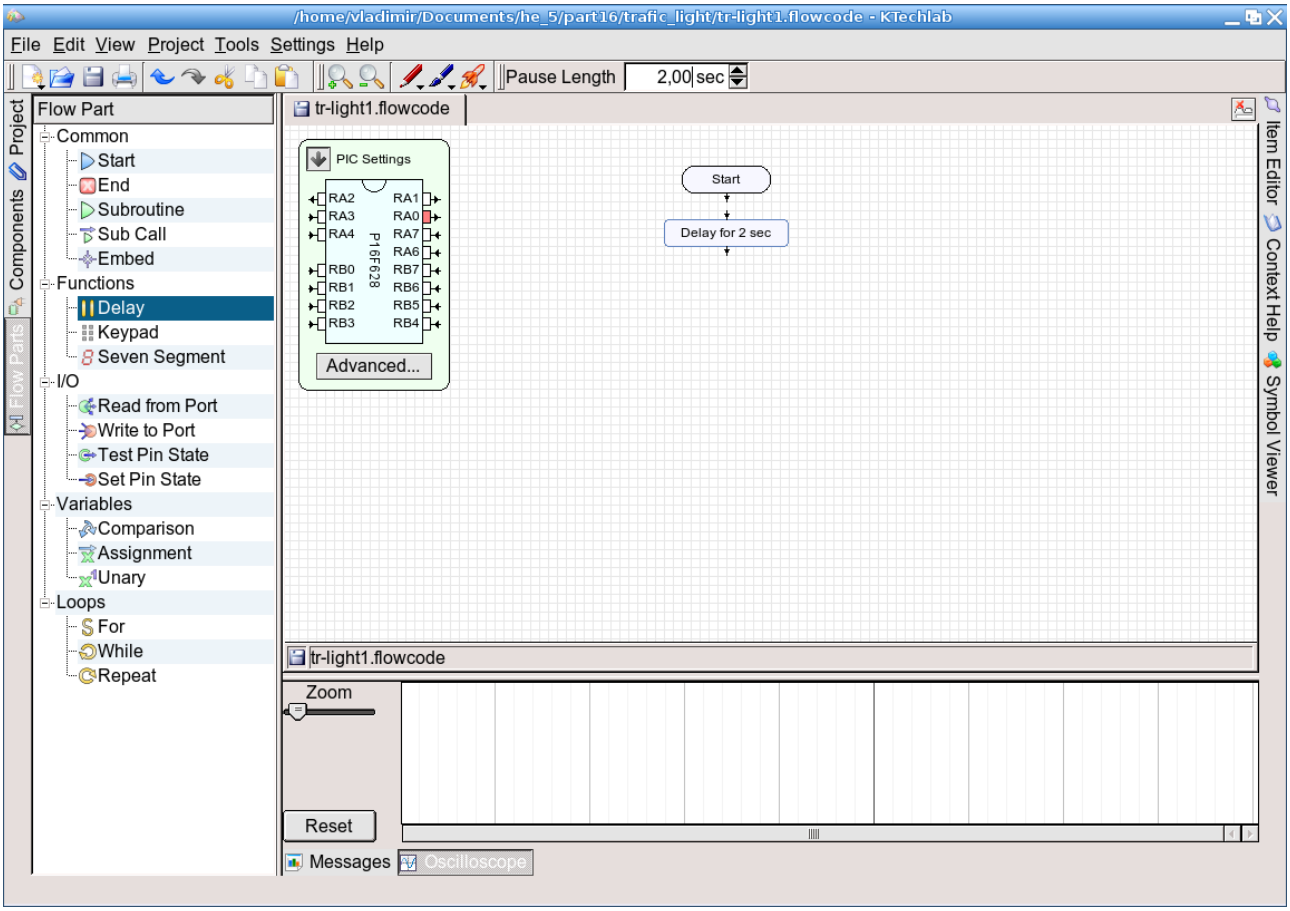


Рис. 16.6. Изменение времени задержки

Теперь обязательно нужно соединить верхнюю часть *Start* с *Delay*: подводим курсор к стрелочке в нижней части овала *Start*, курсор превращается в перекрестие, нажимаем левую клавишу мышки и ведем линию к стрелочке в верхней части *Delay*, и отпускаем клавишу мышки.

Следом за этим переносим на рабочее поле *Set Pin State* из меню компонент. Щелкаем по нему чтобы в его свойствах изменить значение вывода RA0 с *high* на *low*, выключая светодиод. Переносим еще один такой же компонент, у которого меняем RA0 на RA1. Его состояние оставляем как *high*. Конечно, соединяем все вновь установленные компоненты подобно первым двум.

Переносим еще один компонент *Delay*. И еще несколько раз повторяем размещение компонентов *Set Pin State* и *Delay,* думаю, вы уже догадались для чего, повторяя соединение компонентов, за этим следует следить особо. Если компоненты расположены очень близко друг к другу, то их стрелки как бы соединяются, создавая иллюзию соединения. Но компоненты, которые должны быть соединены, должно соединить очевидным образом.

Проведя эти несложные манипуляции, мы получим следующее.

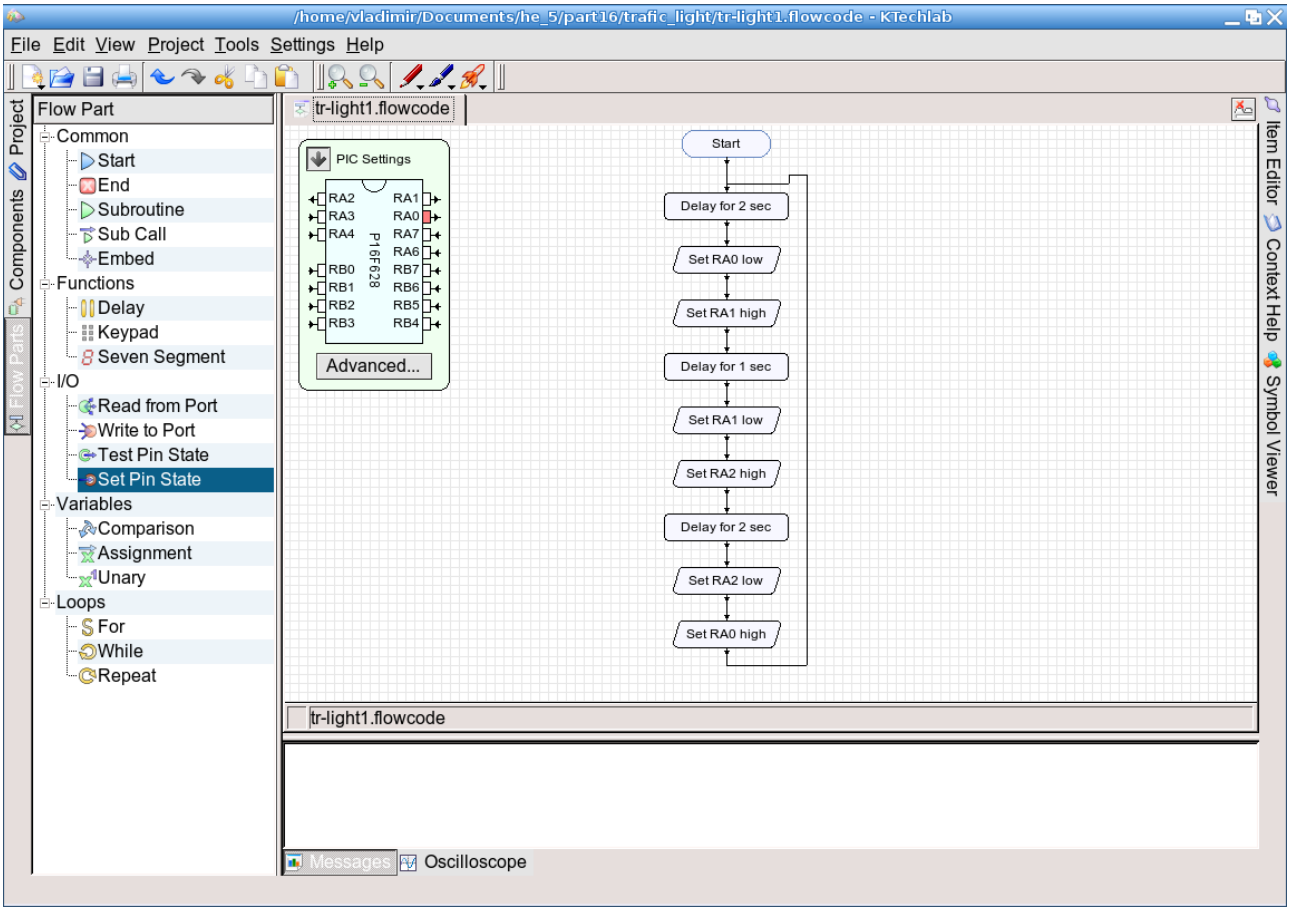


Рис. 16.7. Создание программы по проекту «Светофор»

Ничего вам это не напоминает? Мне напоминает запись алгоритма.

Многие профессиональные программисты считают, что создание программы — это создание алгоритма. Написание кода программы, скорее напоминает пайку схемы на макетной плате, и это дело чисто техническое. Так что, считайте, что программу мы создали.

Если бы от программы требовался только один проход, то закончить ее можно было бы элементом *End* (расположенным под элементом *Start*), но работа программы должна быть постоянной — светофор горит и день, и ночь — поэтому программа замкнута в бесконечный цикл.

А почему я говорил о жульничестве (немного раньше), а потому, что код программы мне писать лень. И не буду. Нажимаю кнопочку с изображением ракеты на инструментальной панели, выбираю в выпадающем меню *Convert to* раздел *Hex,* а в диалоговом окне оставляю флажок на *Display directly* и нажимаю свою любимую клавишу **ОК**.

То, что вы сейчас видите (ниже) это готовая к загрузке в микроконтроллер программа. Поверьте на слово. А поскольку программа KTechlab работает с JDM программатором, схему которого я приводил выше, я мог бы выбрать загрузку в контроллер *PIC (upload)* в выпадающем меню ракеты.

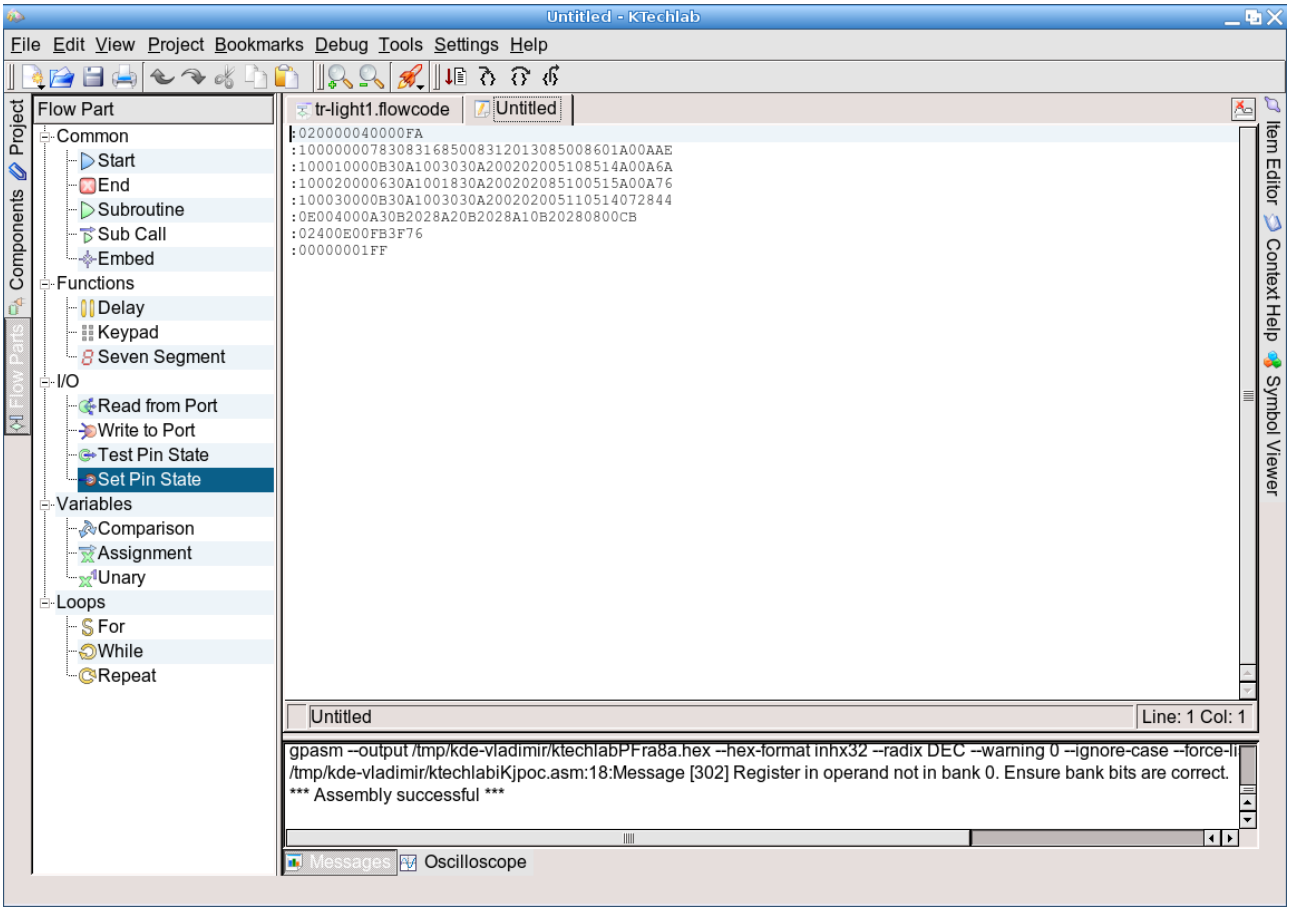


Рис. 16.8. Коды готовой к загрузке программы

Поскольку я сознался, что жульничаю, поскольку порой лукавлю, я, в наказание себе, сейчас включу программатор, загружу программу в микроконтроллер, и перенесу его на макетную плату. Если «светофор» не заработает, то честно об этом напишу.

Некоторые особенности работы с программатором

Ай-яй-яй, давненько не брал я картишки в руки... Все я подзабыл, увы.

В первую очередь это касается работы с программатором в KTechlab. Для его работы требуется дополнительно установить программную часть (утилита это или драйвер устройства я сейчас выяснять не буду). Называется это picprog. Как это называется, я прочитал в своей книге «Наглядная электроника». Хорошо, что сохранил ее.

В дистрибутиве Fedora 7, в списке готовых пакетов, нужной мне picprog не нашлось. Пришлось поискать исходные тексты, обнаруженные под именем picprog_1.8.3.orig.tar.gz. Для поиска программ в Интернете в Linux есть ряд средств, равно как и для установки программ с CD-диска. Распаковав найденный мною архив с исходными текстами на рабочий стол, посмотрев файл с названием README, я обнаружил, что набор команд для установки программы несколько отличен от стандартного (если считать его стандартным), первая команда ниже переход в папку с исходным текстом:

```
cd /home/vladimir/Desktop/picprog-1.8.3/
make dep
make
make install
```

Произведя эти несложные манипуляции, я получил возможность работать с программатором, который в KTechlab есть под именем *PICProg*. Выбор программатора осуществляется в пункте основного меню *Settings*, где выбирается раздел *Configure KTechlab*.

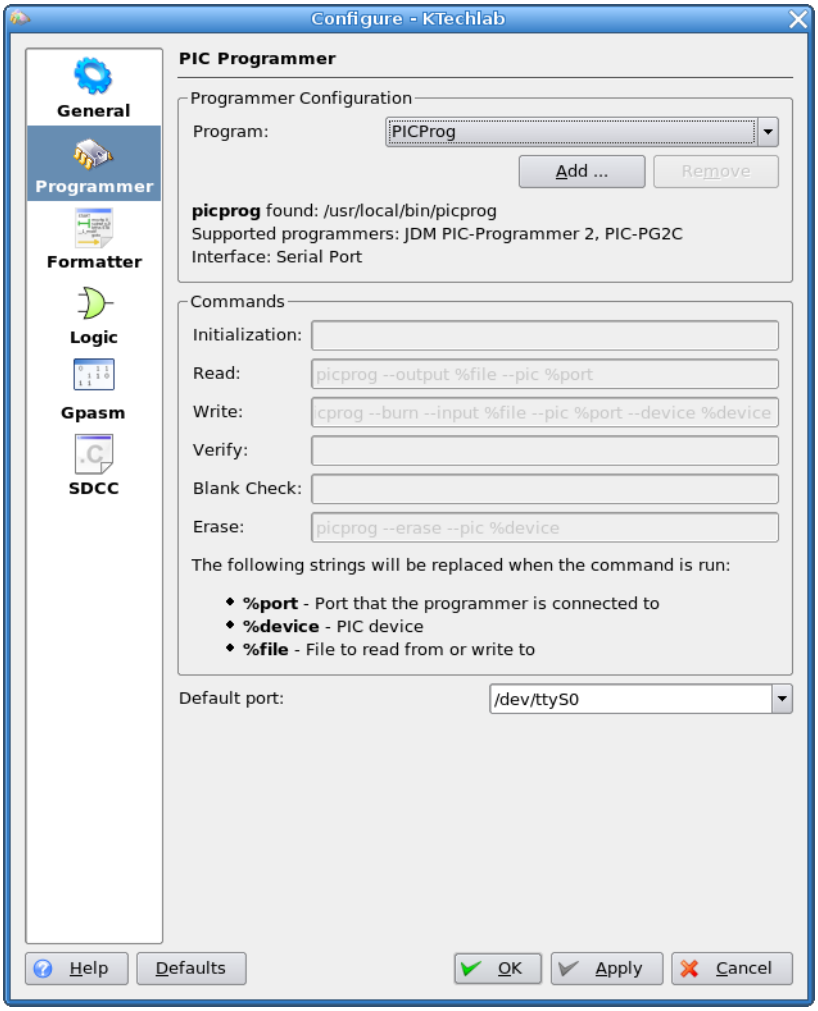


Рис. 16.9. Диалоговое окно настройки программатора

Но это еще не все приключения, о которых я прочитал прежде, чем получилась первая запись кода в программатор. При настройках по умолчанию для выбранного программатора запись не проходит. В опциях, как оказалось, следует указать для режима Write: -- erase. Кроме того, получается, что лучше очевидно указать тип микроконтроллера. В итоге, нажав на клавишу **Add...**, что позволяет создать новую версию настроек программатора, я добавляю picprog2 с предустановками:

Read: picprog --output %file --pic %port --device=pic16f628

Write: picprog --erase --burn --input %file --pic %port --device=pic16f628

Erase: picprog %port --pic %device

Клавиши **Apply** и **OK** позволяют сохранить новые настройки. Однако и этого, как оказалось, недостаточно, для защиты от проделок хакеров все операционные системы сейчас так защищены, что у пользователя компьютера, практически, нет прав ни на что. Доступ к COM-порту (/dev/ttyS0 в системе Linux) мне закрыт. Когда-то было достаточно добавить себя в группу *uucp*, но теперь и этого мало. Чтобы решить эту проблему приходится запускать KTechlab из терминала командной строкой: *su -c ktechlab*.

По дороге к успеху, в тот момент, когда я не обнаружил в дистрибутиве Fedora пакета picprog, я решил, что эту часть работы сделаю в Ubuntu, моей второй операционной системе Linux, дистрибутив которой основан на пакетах Debian.

В Ubuntu, в отличие от Fedora, все нашлось в готовых пакетах. Достаточно было их скачать и установить. Но при первой попытке трансляции программы я столкнулся с тем, что необходимо установить еще пакет gputil. Это справедливо, поскольку утилиты используются для трансляции с ассемблера в машинные коды, но я предполагал их наличие в пакете KTechlab. Тоже не проблема при наличии Интернета, скачал и в сторону. Но во всем остальном, что касалось настроек программатора и прочего, особенной разницы не обнаружилось, пришлось вернуться в Fedora 7.

Последнее, о чем следовало бы упомянуть, и с чем я не разобрался в прошлый раз, работая в основном в Piklab, это слово конфигурации микроконтроллера, без которого он не будет работать. Эта незадача решилась просто. Прежде, чем записывать программу в микроконтроллер, ее следует транслировать в ассемблер (пункт *Conver to Assembly* в выпадающем меню под значком ракеты). В ассемблерном коде в самом начале есть задание слова конфигурации.

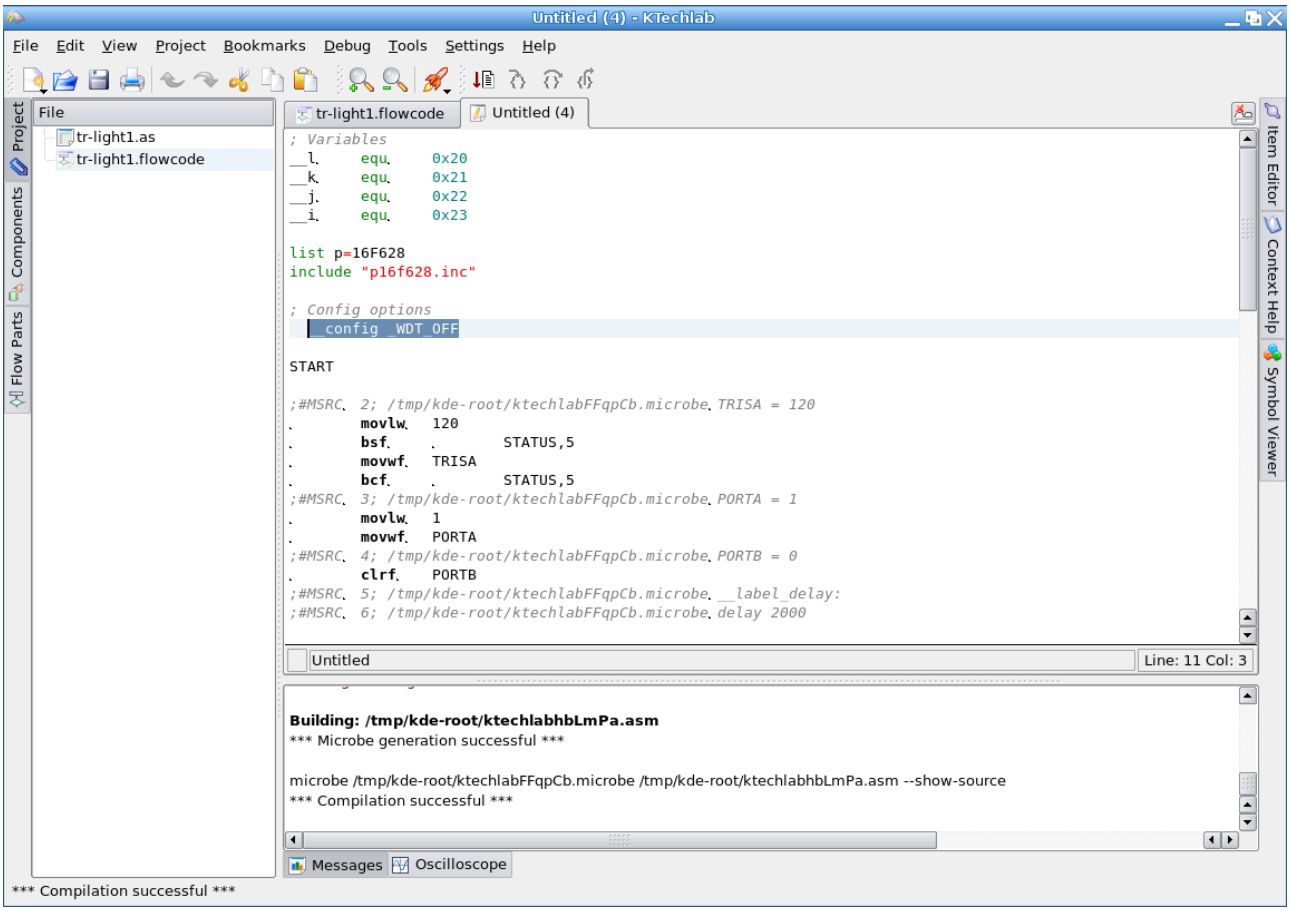


Рис. 16.10. Задание слова конфигурации контроллера в KTechlab

Это место выделено на рисунке. Достаточно, при тех настройках, которые я использую, исправить это на `_config 0x3f18`.

И, наконец, при моих настройках KTechlab запись происходит в чистую микросхему, то есть, мне нужно либо с помощью команд из терминала, обращенных к picprog очистить предыдущую запись, либо использовать, как я делал это раньше, программу Piklab для этих

целей.

Ничто, включая неприятности, не длится вечно. После нескольких нелепых ошибок, допущенных мною по невнимательности (у меня светодиоды не на тех выводах, что в программе), и исправления программы, где я забыл добавить обратный порядок включения светодиодов, программа оказывается в микроконтроллере, микроконтроллер водружается на панельку макетной платы (многострадальной, а потому увешанной дополнительными элементами), и после включения светодиоды отображают ход моих мыслей в процессе создания программы. Чтобы в этом не было сомнений, я приведу окончательный вид программы и добавлю фотографию макетной платы.

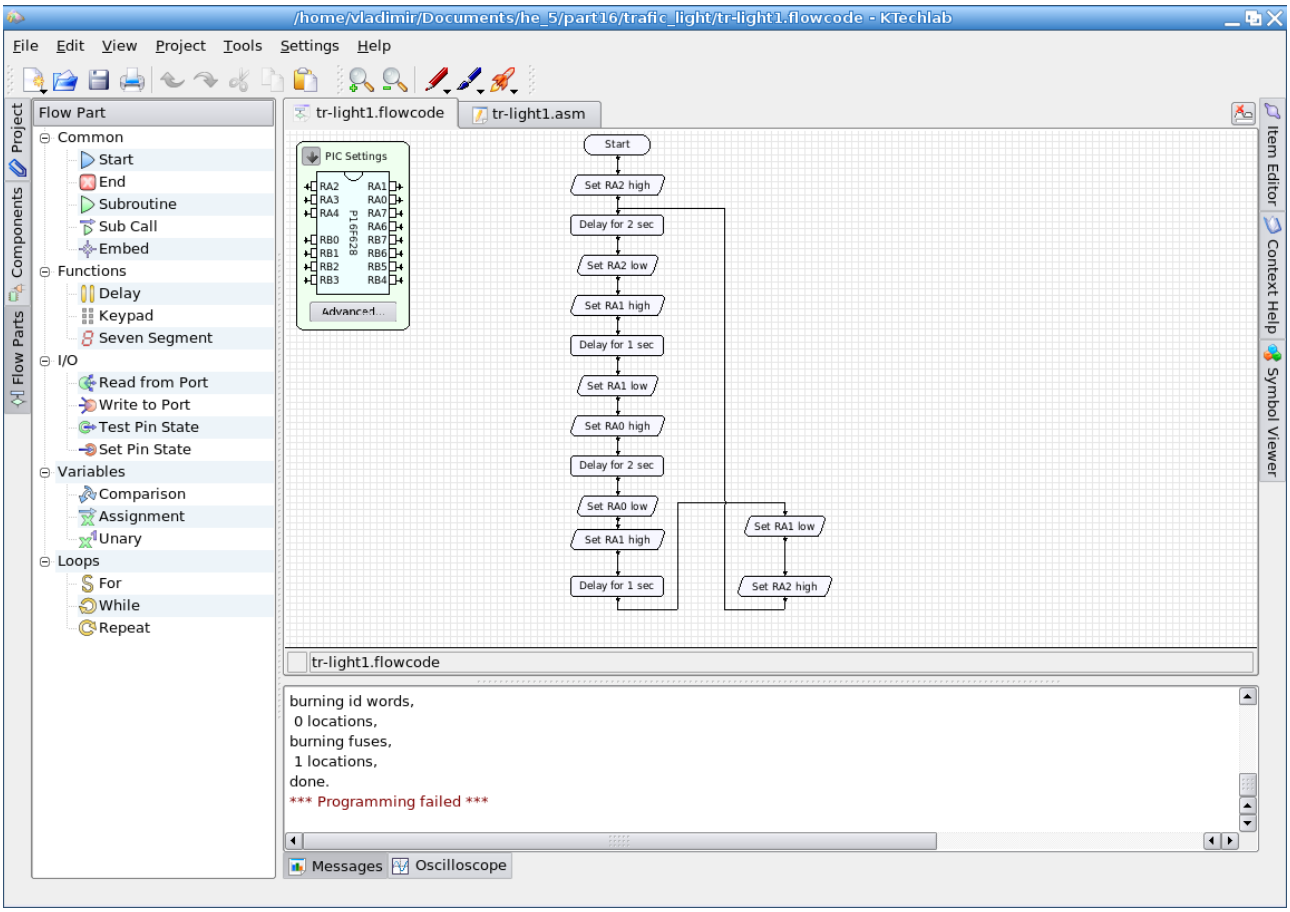


Рис. 16.11. Окончательный вид программы для проверки

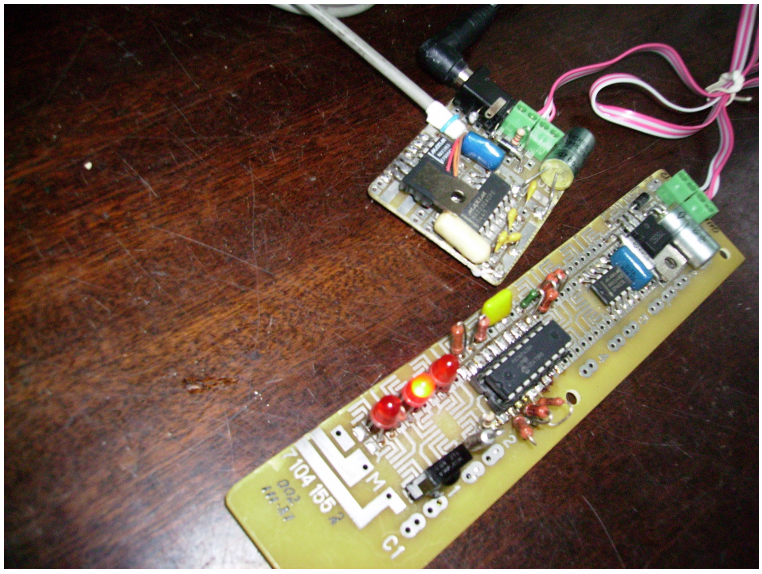


Рис. 16.12. Макетная плата с работающим микроконтроллером PIC16F628A

Чтобы избежать ошибок, даже по рассеянности, можно проверить работу программы до записи ее в микроконтроллер. Для этих целей предназначен отладчик gpsim. При загрузке KTechlab в Ubuntu, по зависимостям, отладчик загрузился без моих напоминаний.

Отладчик должен работать после того, как вы транслировали программу в ассемблерный код. Переходя на закладку этого кода вы можете увидеть, что главное меню изменилось, и в нем появился пункт *Debug*, в котором есть раздел *Run*, нажав на который вы запускаете режим отладки. Справа есть несколько закладок, одна из которых называется *Simbol Viewer*, она-то нам и нужна. Открыв ее можно увидеть содержимое регистров процессора и состояния выводов портов. На основной инструментальной панели теперь есть клавиши пошагового прохождения программы, пропуска и выхода из циклов. Пройдя по шагам, убегая из циклов, уж очень их долго нужно будет выполнять, можно посмотреть, как выводы порта А меняют свое состояние.

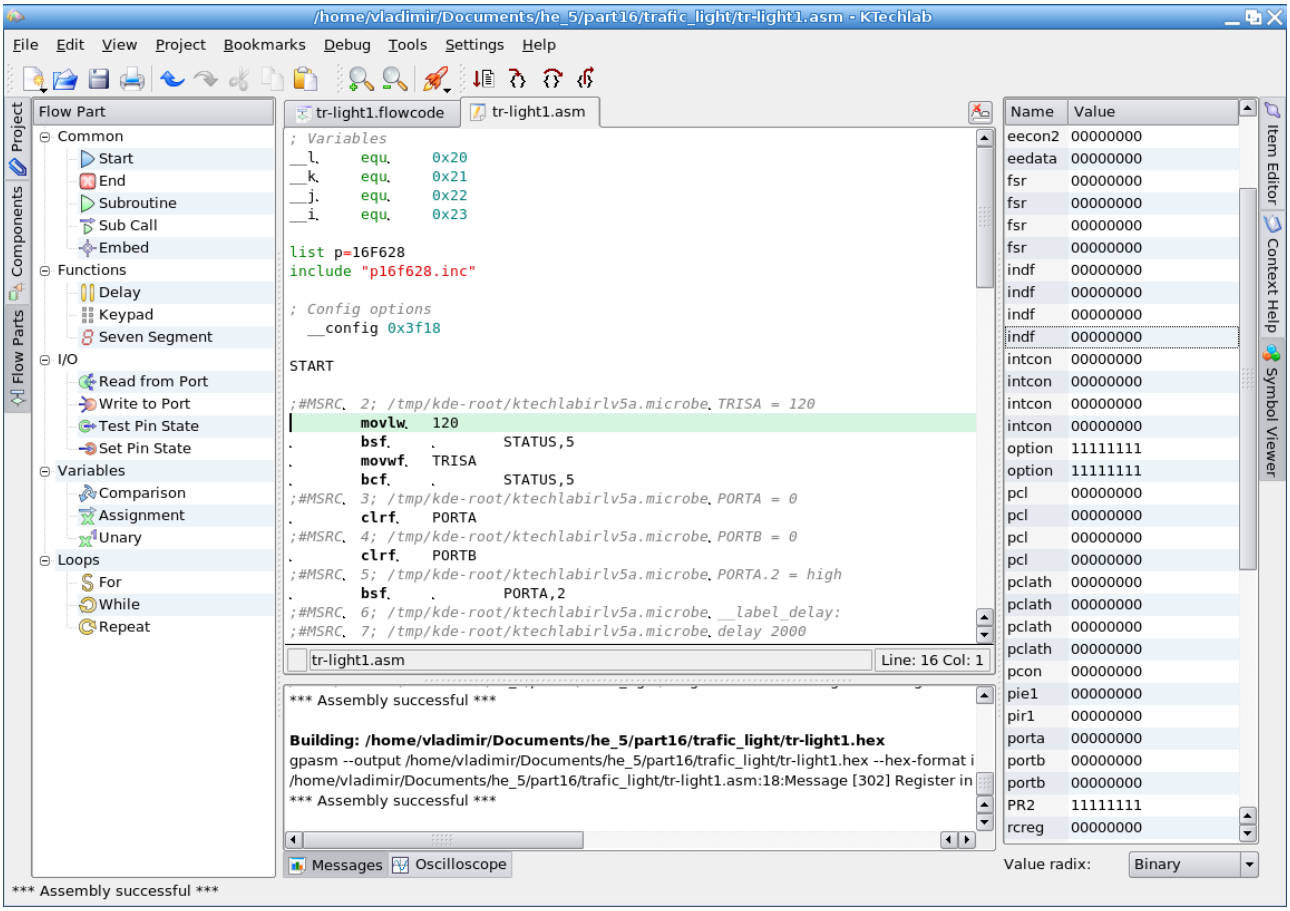


Рис. 16.13. Режим отладки в программе KEechlab

Этим, правда не завершаются возможности отладки программы. Если запустить отладчик *gpsim* самостоятельно из терминала, можно получить более наглядное представление о работе вашей программы. После появления интерфейса отладчика в его основном меню выбирается пункт *File*, для загрузки файла с расширением .cod (у меня это файл tr-light1.cod). Желательно предварительно, или это только у меня, выбрать в пункте *Windows* окно *Source*. При трансляции программы в KTechlab файл с расширением .cod создается автоматически, и хотя отладчик может работать без него, но с ним, можете убедиться, с ним он работает не в пример лучше. Визуальное наблюдение за выполнением хода программы, после того как вы нажали клавишу **Run** основного меню, не исчерпывает дополнительных возможностей отладчика.

Есть удобное средство отладки под названием макетная плата (breadboard), которое вы найдете в том же пункте основного меню *Windows*. На макетную плату можно добавить множество компонентов, предусмотренных программой. Для этого на макетной плате есть клавиша **Add library**. Нажав на нее, вы попадете в окно диалога выбора библиотеки, а поскольку у пока библиотека только та, что пришла с программой, следует воспользоваться подсказкой и ввести в окно запроса: libgpsim_modules. Для подключения модулей служит понятие узла (клавиша **Add node**), к которому можно подключать выводы микроконтроллера и выводы модуля, который вы выбрали. Сейчас посмотрите, как выглядит работа отладчика, а потом я немного больше расскажу о работе макетной платы.

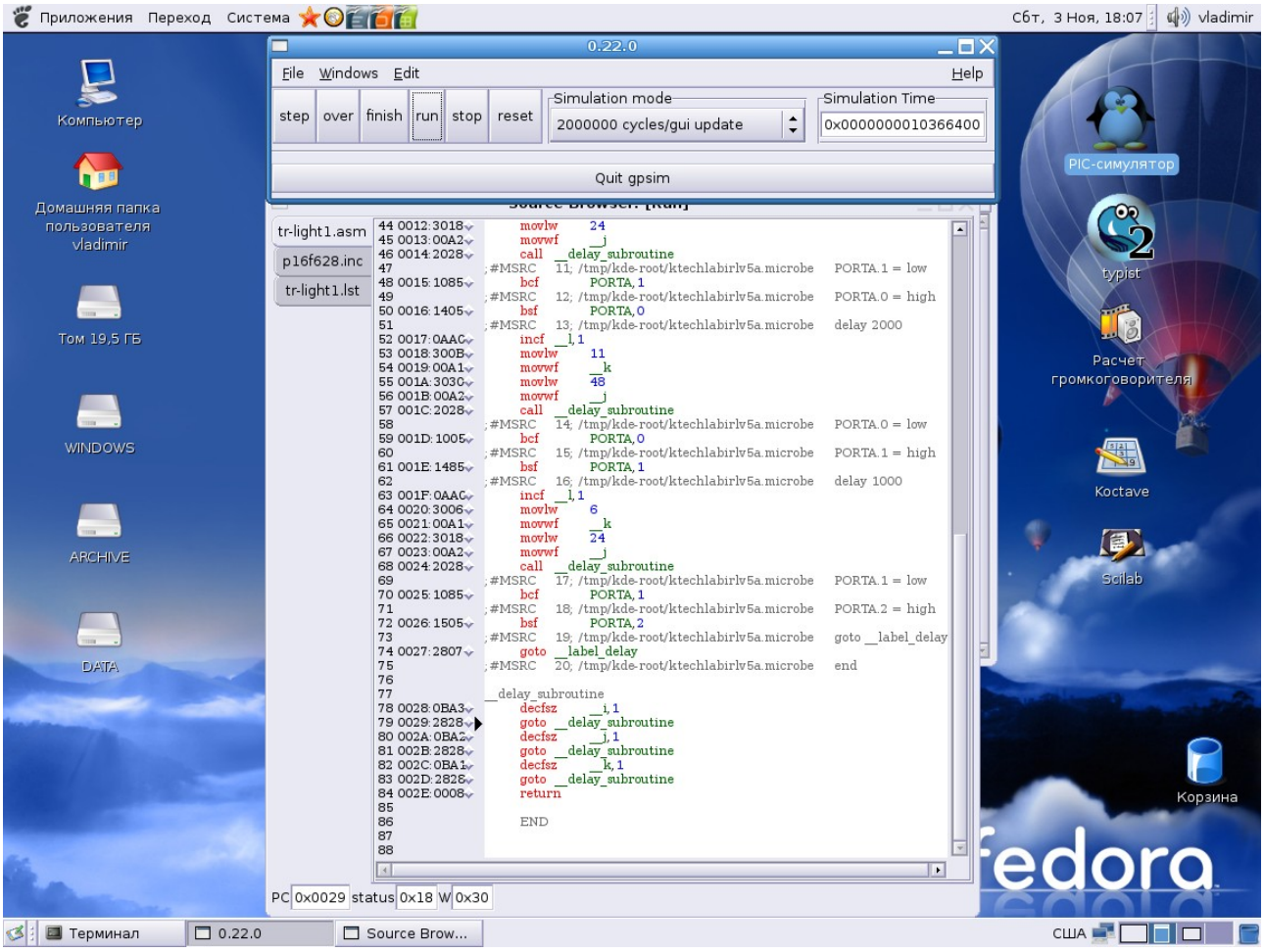


Рис. 16.14. Более полное использование отладчика gpsim

Кроме основного окна отображения программы есть окна, в которых можно посмотреть состояния регистров, памяти и т.д., есть даже встроенный, авторы проекта еще не завершили эту часть работы, осциллограф.

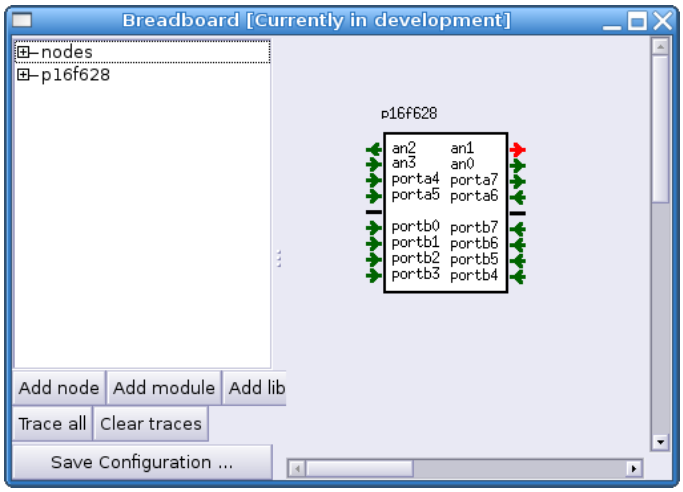


Рис. 16.15. Вид макетной платы отладчика gpsim

Запустив отладку еще раз клавишей **Run**, можно видеть, как зажгутся светодиоды на

реальной макетной плате.

А вот как выглядит набор дополнительных модулей к макетной плате.

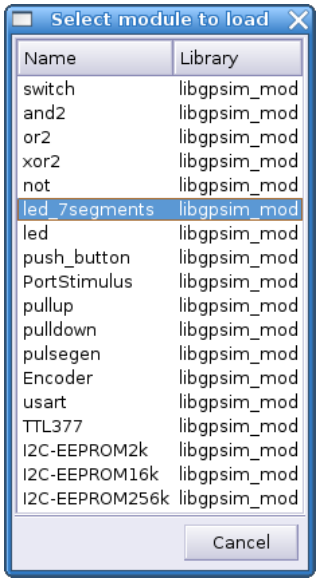


Рис. 16.16. Набор модулей макетной платы отладчика gpsim

После того, как вы добавите модули на макетную плату, после того, как соедините их с микроконтроллером, очень не лишним будет сохранить конфигурацию с помощью клавиши **Save Configuration...** на макетной плате. В предыдущих версиях были некоторые «заморочки» с этим, о которых я писал в предыдущих книгах, но, если вы не сохраните конфигурацию, после новой загрузки вы получите чистую макетную плату, и все операции по заполнению ее придется повторять с самого начала, думаю, вам это не понравится.

Завершение проекта «Автомат для рассеянных»

Познакомившись с работой в программе KTechlab вы, надеюсь, понимаете, почему я каждый раз, когда пытался сделать что-то на цифровых микросхемах, с большим трудом мог продвигаться в работе — мне мешала мысль, что это легко и просто можно сделать с помощью микроконтроллера. Если возможностей KTechlab вам покажется мало, мне так не показалось, ведь остались не рассмотрены возможности microb, составной части этой программы, остаются полностью открыты все возможности программирования на ассемблере и языке «С», но тем не менее, если... то можно работать с программой Piklab.

Полагаю, что с завершением проекта «Автомат для рассеянных», вернее с построением его на микроконтроллере, вы готовы справиться самостоятельно. Не буду портить вам удовольствие своими советами.

Тем не менее, я хочу сказать, что выбор микроконтроллера, я выбрал PIC16f628A из соображений его доступности и стоимости, выбор микроконтроллера зависит в первую очередь от задачи, которую он должен решать. Если контроллер прекрасно справляется с задачей, нет смысла искать ему замену. Я использовал один и тот же контроллер при написании нескольких книг, и не все проекты были так же просты, как «Светофор». Контроллер прекрасно справлялся со всем, что я просил его сделать.

Если на макетной плате («физической», не программной) поставить панельку для микроконтроллера, то на одной макетной плате можно проверить десятки, если не много больше, решений с одной единственной микросхемой. С двумя микросхемами и двумя

панельками, тоже очень интересный эксперимент, можно заставить микроконтроллеры общаться между собой. Дело в том, что у PIC16F628A есть встроенный USART — готовый интерфейс, например, для использования RS232 или, что мне кажется предпочтительней, RS485. Первые эксперименты можно проделать на столе без дополнительных микросхем, а с двумя микросхемами, двумя контроллерами на разных макетных платах, можно продолжить эксперименты с линией в несколько сот метров.

Завершение проекта «Электроника для начинающих»

Книга еще не завершилась, но мне хочется досрочно написать заключение. Я не буду скрывать его до конца книги, как положено по правилам, я напишу его сейчас.

Тот подход к освоению электроники, который описан в этой книге, не базируется на методических указаниях или опыте преподавания. Нет у меня такого опыта. Да и не хотел я, как сказал в самом начале, учить кого-либо. Я хотел написать «роман в технической прозе» о том, что есть такой интересный предмет для увлечения, как электроника.

Я всегда старался дополнить изложение советами перейти к «г-ну Макету» после предварительного рассмотрения схемы в программе EDA, не только по причине недоверия к этим программам, но потому что физическое воплощение своих идей — это неотъемлемая часть обыденной работы даже профессионалов. Умение паять, работать с приборами — этому тоже надо учиться, сразу может и не получиться. У меня, например, не получалось, приходилось набраться терпения и разбираться со всеми проблемами. Очень помогают книги. К сегодняшнему дню хороших книг написано столько, что, думаю, невозможно их все прочитать. И не всегда есть возможность купить то, что хотелось бы. Хорошая помощь в этом — радиолюбительские сайты, на которых можно найти и книги, и схемы, и полезные советы. На сайте сети магазинов «Чип и Дип» можно найти описания многих электронных компонент: транзисторов, микросхем, трансформаторов, громкоговорителей...

В предисловии я написал, что надеюсь сам узнать что-то новое, пока пишу эту книгу. Действительно я познакомился с двумя Александрами, от одного узнал, насколько сильно изменились интересы радиолюбителей, а второй привлек мое внимание к современным преобразователям и проблемам симуляции импульсных схем. Я никогда не специализировался в этой области, и еще раз убедился, сколько интересного можно найти в электронике.

Если у вас хватило терпения дочитать книгу, если у вас хватило сообразительности самостоятельно проверить все написанное в ней, то в этом месте для вас точно закончился славный период радиолюбительского детства. Вы возмужали, окрепли, да и просто выросли. Вы теперь взрослый любитель, а если вы молоды и электроника вам понравилась, вы, возможно, захотите ее сделать своей профессией. Не знаю, насколько это решение практично, и оно явно не из категории «брака по расчету», но оно достаточно... пусть будет, «романтично». Это хорошее слово. Для меня оно означает воспоминания о хороших песнях — а я еду за туманом... — и хороших людях, и, в общем-то, интересной работе.

Глава 17. Измерения в электрических цепях

Ах, когда-то все было не так, как сейчас. Как славно было когда-то: детекторный приемник, как предмет вожделения, и собственные уши, как единственный измерительный прибор.

Измерения — единственный способ узнать достоверно, что происходит в электрической схеме, независимо от того, проста она или сложна. Но измерения могут стать источником ошибки, если не учитывать реальных свойств измеряемой цепи и параметров приборов. Например, вольтметр имеет некоторое сопротивление. Его достаточно во многих случаях для того, чтобы в работу схемы не вносилось изменений, но во всех случаях. Если сопротивление вольтметра соизмеримо с сопротивлением в месте измерения, то сам вольтметр может заставить схему работать правильно, и вы увидите правильное значение напряжения. Или осциллограф, щуп которого имеет определенную емкость, может исказить сигнал, заставив вас думать, что виновата в этом схема.

Время меняет интересы, потребности и возможности. Сегодня радиолюбителю не обойтись без множества приборов в своей лаборатории. Однако минимальный набор приборов доступен, практически, каждому, если к минимальному набору приборов отнести цифровой мультиметр. Имея в своем распоряжении этот прибор, вы можете создавать, по мере необходимости, новые.

Возьмем такой пример: у вас нет осциллографа, но вы хотите собрать усилитель. Без осциллографа и генератора трудно даже сказать, работает ли усилитель? В таком случае можно выбрать несколько путей к цели.

Можно использовать вместо генератора естественные источники сигналов. Например, динамический микрофон развивает напряжение порядка 0.5 — 1 мВ, линейный выход, если он есть у радиоприемника или музыкального центра, даст сигнал с напряжением около 250 мВ. А в качестве измерительного прибора можно использовать слух — наушник с последовательно включенным резистором в 1 кОм и конденсатором 5 — 10 мкФ, лучше не электролитическим, позволит услышать и оценить результат.

Достаточно просто собрать генератор прямоугольных импульсов на цифровых микросхемах. При выходном напряжении, скажем, 5 В импульсы в своем составе будут иметь широкий спектр частот до значений в сотню раз превышающих исходную частоту. Делитель напряжения из нескольких резисторов позволит регулировать величину сигнала. Сигнал можно услышать, а можно измерить мультиметром. Правда, показания мультиметра при измерении переменного напряжения относятся к измерениям синусоидального напряжения, но такой параметр как коэффициент усиления по напряжению, оценить можно.

Мультиметр измеряет переменное напряжение в довольно ограниченном частотном диапазоне. Но, чтобы оценить высокочастотный усилитель, можно использовать детектор, подобный детектору радиоприемника, который при достаточно большом сигнале вполне может помочь с оценкой ряда параметров усилителя.

Сегодня, как мне кажется, можно найти микросхему NE566 (LM566, SE566). На микросхеме можно собрать генератор импульсов и пилообразного напряжения. Следующая схема из книги Р. Фелпса «750 практических электронных схем».

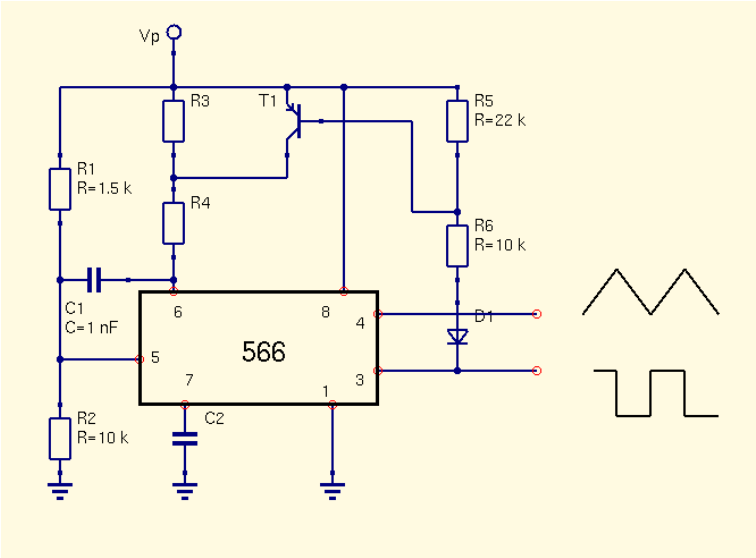


Рис. 17.1. Генератор импульсов и пилообразного напряжения

Частота генератора (в отсутствии T1) определится из соотношения F = 1/3(R3+R4)C2. Еще более интересную схему функционального генератора можно найти в той же книге, схему построенную на специализированной микросхеме фирмы EXAR XR-2206.

Если полистать журналы «Радио», если побродить по Интернету, можно найти множество достаточно простых схем функциональных генераторов. Одной из трудностей при сборке своего генератора раньше была необходимость рисовать шкалу, что сегодня не составляет труда, если есть принтер. Вторая трудность в том, чтобы откалибровать генератор. И с этим сегодня проще — при покупке мультиметра можно выбрать такой, у которого есть встроенный частотомер. Он может помочь и в калибровке, и он может использоваться как удобная шкала генератора.

Очень полезным и нужным прибором, особенно на начальном этапе, является осциллограф. Даже при измерении постоянного напряжения, если вы измеряете напряжение мультиметром, вы можете неверно оценить ситуацию. У сетевого блока питания выходное напряжение может иметь пульсации, которые мультиметр не заметит, а именно они могут стать причиной проблем в работе схемы. В этом смысле осциллограф незаменим. Если реализовать один из вариантов, о которых я писал в главе об осциллографе, не получается, можно использовать программу Qucs на первом этапе, а затем попробовать измерить мультиметром, включив его через конденсатор на измерение переменного напряжения, величину этих пульсаций. Это не будет точным измерением, но позволит вам оценить меру воздействия этого паразитного явления на работу схемы.

Если в распоряжении радиолюбителя оказывается осциллограф, исключая, скажем, приставку к компьютеру, о которой я писал выше, то, как правило, это довольно примитивная по сегодняшним меркам модель. Но недавно, перелистывая связку старых журналов «Радио» при написании книги «Наглядная электроника», я обнаружил, сколь много полезных и несложных схем можно там найти для повторения. Например, коммутаторы, превращающие одноканальный осциллограф в двухканальный, или многоканальный для наблюдения сигналов в цифровых устройствах. Или функциональный генератор на одной цифровой микросхеме. В своем архиве я нашел эту схему и описание ее работы, приведу их.

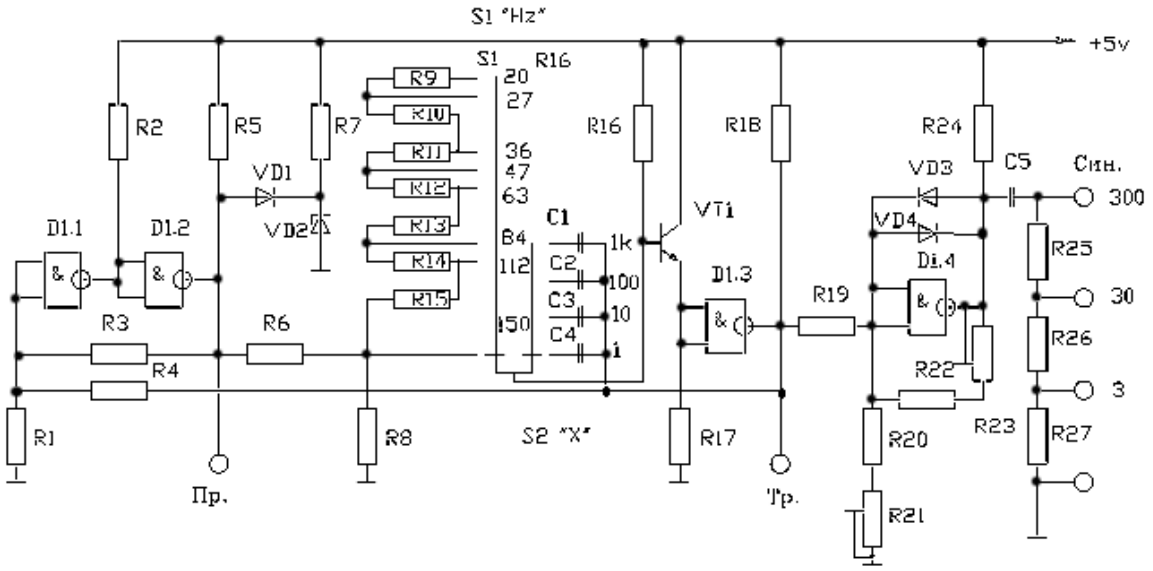


Рис. 17.2. Функциональный генератор на цифровой микросхеме

Схема этого функционального генератора опубликована в журнале «Радио» No11 за 1980 г. Схема разработана Л. Ануфриевым по заданию редакции . В целях упрощения генератор разработан как генератор дискретных частот, перекрывающий диапазон от 20 герц до 150 килогерц. В первом поддиапазоне это частоты 20, 27, 36, 47, 63, 84, 112, 150 герц, остальные поддиапазоны в десять, сто и тысячу раз выше. Выходное напряжение синусоидального сигнала 300мВ, 30мВ, 3мВ.

На элементах D1.1 и D1.2 выполнен компаратор. На транзисторе VT1, элементе D1.3 и конденсаторах C1 — C4 построен интегратор, а на элементе D1.4 и диодахVD3 и VD4 преобразователь треугольного напряжения в синусоидальное.

Как возникают колебания в генераторе?

Положим, на выходе D1.2 высокий уровень напряжения (переключатели S1 и S2 в положении, показанном на схеме), конденсатор C4 начинает заряжаться через резистор R6 напряжением с выхода D1.2, а напряжение на выходе интегратора начнет линейно спадать. Это напряжение через резистор R4 поступает на вход компаратора, и как только оно достигнет величины 0.5В (определяется сопротивлением R3) компаратор переключится в другое устойчивое состояние с низким уровнем на выходе D1.2. Но напряжение на базе транзистора VT1 выше этого напряжения, что заставит конденсатор C4 разряжаться через резистор R6 и выходное сопротивление элемента D1.2. Напряжение же на выходе интегратора при этом будет линейно нарастать, пока не достигнет величины в 3.7В, при которой компаратор вновь вернется в исходное состояние с высоким уровнем на выходе D1.2, а напряжение на выходе интегратора начнет опять линейно спадать. Изменяя зарядное сопротивление (R6, R8 — R15) или емкость (C1 — C4), можно изменять скорость заряда, а, следовательно, и частоту генератора.

Функциональный преобразователь сигнала треугольной формы в синусоидальный — это обычный усилитель на элементе D1.4, охваченный линейной (через R22 и R23) и нелинейной (через VD3 и VD4) обратной связью. Открываясь на вершинах треугольных импульсов, диоды «скругляют» треугольное напряжения, превращая его в близкое к синусоидальному.

Используемые компоненты:

DD1 - К155ЛА8, VT1 - КТ315Б, VD1, VD3, VD4 - КД522А, VD2 - КС133А.

Резисторы МЛТ 0.125Вт, 5%: R1 - 1.3кОм, R2 - 7.5кОм, R3 - 5.6кОм, R4 — 5.1кОм, R5 - 430, R6 - 3.9кОм, R7 - 510, R8 - 27кОм, R9 - 6.8кОм, R10 - 5.1кОм, R11 - 3.9кОм, R12 - 3кОм, R13 - 2.2кОм, R14 - 1.6кОм, R15 - 1.2кОм, R16 - 240кОм, R17 - 1.1кОм, R18 — 390, R19 - 2.4кОм, R20 - 820, R22 - 910, R24 - 1.3кОм, R25 - 8.2кОм, R26 - 820, R27 - 91.

Резисторы СП3-1Б: R21 - 1кОм, R23 - 470.

Конденсаторы МБГП, МБМ, КСО, К40П-2 (5%): C1 - 470пФ, C2 - 4700пФ, C3 - 0.047мкФ, C4 - 0.47мкФ.

Конденсатор C5 - 100.0х10В типа К50-3, К50-6.

Переключатели S1 - ПГГ 11П1Н; S2 - ППГ 5П2Н.

Налаживания генератора начинают с проверки напряжения питания, которое должно быть равно 5 вольтам. Подключив осциллограф к выходу сигнала прямоугольной формы «Пр», устанавливают переключатель S1 в нижнее по схеме положение, соответствующее максимальной частоте, а S2 в среднее. При отсутствии на экране осциллографа прямоугольных колебаний проверяют уровень напряжения на выходе «Тр». Если он низкий (0.2В), то необходимо уменьшить сопротивление резистора R1, если высокий — увеличить. С появлением треугольных импульсов следует отрегулировать резистором R1 их размах от 0.5В до 3.5-3.7В. Симметричности треугольных импульсов добиваются резистором R8. Увеличение этого резистора приводит к увеличению скорости спада треугольного напряжения. При напряжении стабилизации стабилитрона VD2 3В резистор R8 возможно исключить из схемы, а, если симметрии импульсов не удается добиться, то следует попробовать резистор R8 подключить к положительному полюсу. После симметрирования импульсов на высокой частоте диапазона переключатель S1 следует переключить в верхнее по схеме положение, перейдя на нижние частоты диапазона. Здесь симметрии импульсов добиваются подстройкой резистора R16.

После этого замеряют период треугольных импульсов, который при подключении конденсатора C3 (диапазон «х10») должен быть 5мС (с точностью 10%). Подстройку следует вести резистором R4 — увеличение резистора приводит к увеличению периода и наоборот.

Теперь осциллограф переключают на 300мВ выход синусоидального напряжения и резистором R21 добиваются симметричного ограничения сигнала, а резистором R23 изменяют порог ограничения, добиваясь наилучшей формы сигнала.

Осталось проверить работу генератора на всех диапазонах и всех частотах.

Особенностью выбора ряда частот в пределах диапазона является то, что при построении АЧХ в логарифмическом масштабе (самый удобный, как правило, вариант) расстояние на графике между значениями этих частот одинаково.

Нагрузочная способность выходов прямоугольных и треугольных сигналов невелика. Входное сопротивление подключаемых устройств должно быть не менее 10кОм.

Ранее в книге я упоминал коммутатор для обычного осциллографа, превращающий его в двухлучевой. Схему я нашел на сайте www.radio-portal.ru, приведу ее вместе с приложенным к ней описанием и ссылкой на источник.

Схема коммутатора к осциллографу

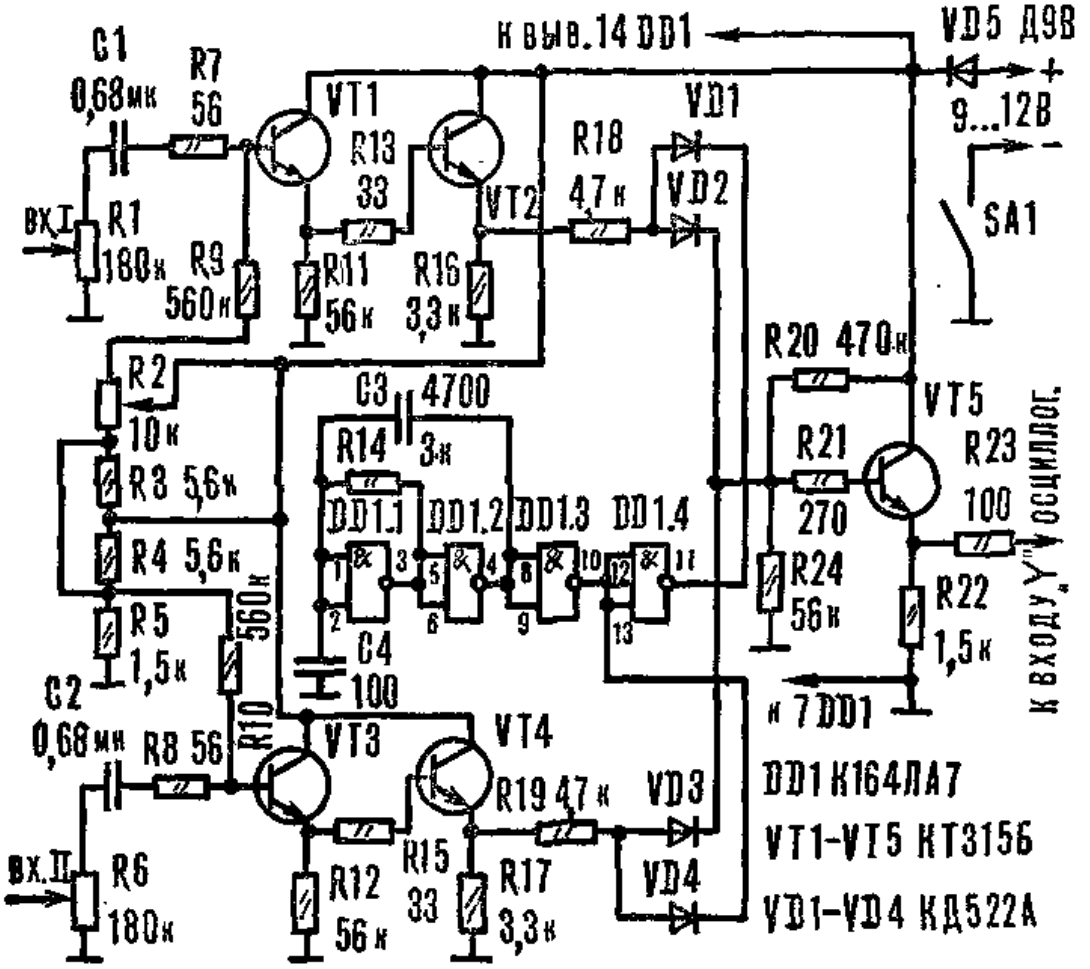


Рис. 17.3. Схема коммутатора для осциллографа

Познакомимся с работой устройства. Тактовый генератор, собранный на логических элементах DD1.1 и DD1.2, генерирует прямоугольные импульсы с частотой 25 кГц и скважностью, равной 2. С выхода генератора (вывод 4 DD1.2) импульсы поступают на буферные элементы DD1.3 и DD1.4, с выходов которых противофазные напряжения поочередно закрывают диоды VD2, VD3, включенные в цепи прохождения исследуемых сигналов. Поскольку база транзистора VT5 эмиттерного повторителя соединена через ограничительный резистор R21 с катодами этих диодов, на резисторе R22 выделяются прямоугольные импульсы тактового генератора. Их амплитуда (расстояние между средними линиями) зависит от положения движка переменного резистора R2 — в верхнем по схеме положении амплитуда импульсов наибольшая, в нижнем — наименьшая. Если теперь на входы I, II подать исследуемые сигналы, то на экране осциллографа (при длительности развертки 10 мкс) появится импульсное напряжение.

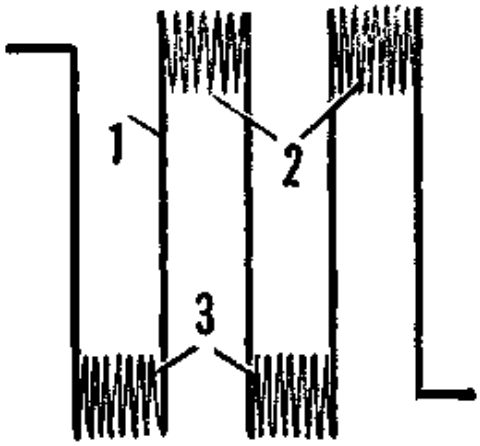


Рис. 17.4. Осциллограмма напряжения

1 — импульс тактового генератора, 2 — первый исследуемый сигнал, 3 — второй исследуемый сигнал.

При большей длительности развертки (1 мс) на экране осциллографа можно наблюдать оба исследуемых сигнала. Их амплитуды и расстояние между сигналами устанавливают изменением положения движков переменных резисторов Rl, R6 и R2 соответственно. Частота тактового генератора зависит от емкости конденсатора С3 и сопротивления резистора R14.

Микросхему К164ЛА7 можно заменить на МС К164ЛЕ5, К564ЛА7, К564ЛЕ5, а также на К176ЛА7, К176ЛЕ5. К561ЛА7, К561ЛЕ5. При использовании последних элементов печатную плату под выводы микросхем придется изменить. Вместо транзисторов КТ315Б можно применить любые полупроводниковые приборы той же серии, подойдут и КТ3102А. Коэффициент передачи по току для всех транзисторов — 40—100. Желательно, чтобы параметры VT1, VT3 и VT2, VT4 были одинаковыми. Диоды КД522А можно заменить на КД521А или КД5ЮА, VD5 — серии Д9 или Д2. Постоянные резисторы — МЛТ-0,125 или ВС-0,125, переменные — СПО-0,5 или СП-1 группы А. Конденсаторы КМ или К10-7В.

Коммутатор питается от батареи «Крона» или автономного источника. Потребляемый ток не превышает 10 мА. Если радиоэлементы исправны, устройство начинает работать без настройки.

А. ПРОСКУРИН «М-К» № 6/87

Схему, приведенную выше, я заранее скачал, изменил формат файла, чтобы его было удобно вставить в текст, но... файл отчего-то оказался испорчен. Пришлось повторно искать эту схему в Интернете, а пока я этим занимался, я наткнулся на другую схему на сайте http://cxem.net, которую не собирался первоначально включать в эту главу. Немного поразмыслив, я решил ее включить, а причины этого поясню после приведенной схемы и ее описания.

Схема приставки к мультиметру для измерения L и C

Цифровой измерительный прибор в лаборатории радиолюбителя теперь не редкость. Однако не часто им можно измерить параметры конденсаторов и катушек индуктивности, даже если это мультиметр. Описываемая здесь простая приставка предназначена для использования совместно с мультиметрами или цифровыми вольтметрами (например, М-830В, М-832 и им подобными), не имеющими режима измерения параметров реактивных элементов.

Для измерения емкости и индуктивности с помощью несложной приставки использован принцип, подробно описанный в статье А. Степанова «Простой LC-метр» в «Радио» № 3 за 1982 г. Предлагаемый измеритель несколько упрощен (вместо генератора с кварцевым резонатором и декадного делителя частоты применен мультивибратор с переключаемой частотой генерации), но он позволяет с достаточной для практики точностью измерять емкость в пределах 2 пф ...1 мкф и индуктивность 2 мкГн ... 1 Гн. Кроме того, в нем вырабатывается напряжение прямоугольной формы с фиксированными частотами 1 МГц, 100 кГц, 10 кГц, 1 кГц, 100 Гц и регулируемой амплитудой от 0 до 5 В, что расширяет область применения устройства.

Задающий генератор измерителя (рисунок 17.5) выполнен на элементах микросхемы DD1 (КМОП), частоту на его выходе изменяют с помощью переключателя SA1 в пределах 1 МГц - 100 Гц, подключая конденсаторы С1-С5. С генератора сигнал поступает на электронный ключ, собранный на транзисторе VT1. Переключателем SA2 выбирают режим измерения «L» или «С». В показанном на схеме положении переключателя приставка измеряет индуктивность. Измеряемую катушку индуктивности подключают к гнездам Х4, Х5, конденсатор - к Х3, Х4, а вольтметр - к гнездам Х6, Х7.

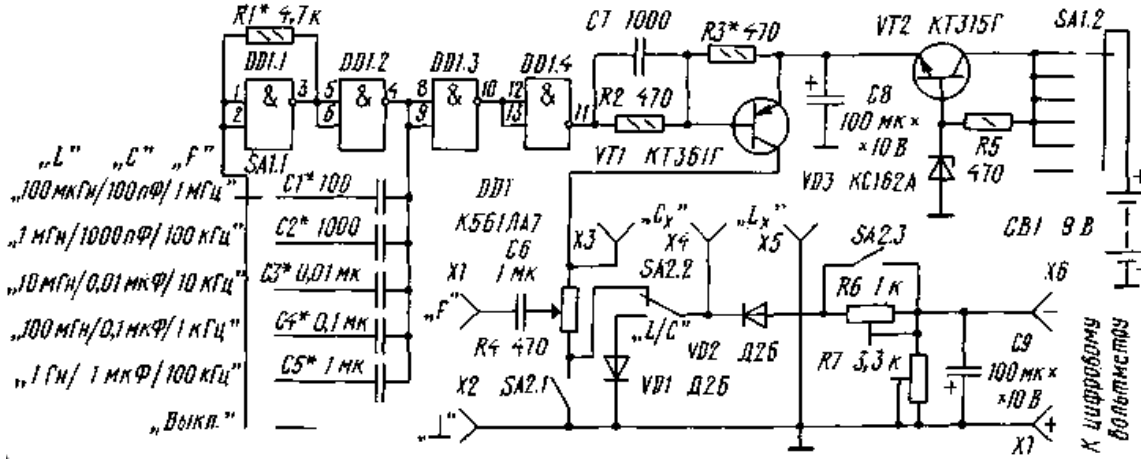


Рис. 17.5. Схема приставки для измерения LC

При работе вольтметр устанавливают в режим измерения постоянного напряжения с верхним пределом 1 - 2В. Следует учесть, что на выходе приставки напряжение изменяется в пределах 0... 1 В. На гнездах Х1, Х2 в режиме измерения емкости (переключатель SA2 - в положении «С») присутствует регулируемое напряжение прямоугольной формы. Его амплитуду можно плавно изменять переменным резистором R4.

Питается приставка от батареи GB1 с напряжением 9 В ("Корунд" или аналогичные ей) через стабилизатор на транзисторе VT2 и стабилитроне VD3.

Микросхему К561ЛА7 можно заменить на К561ЛЕ5 или К561ЛА9 (исключив DD1.4), транзисторы VT1 и VT2-на любые маломощные кремниевые соответствующей структуры, стабилитрон VD3 заменим на КС156А, КС168А. Диоды VD1, VD2 - любые точечные германиевые, например, Д2, Д9, Д18. Переключатели желательно использовать миниатюрные.

Корпус прибора - самодельный или готовый подходящих размеров. Монтаж деталей в корпусе - навесной на переключателях, резисторе R4 и гнездах. Разъемы ХЗ-Х5 - самодельные, изготовлены из листовой латуни или меди толщиной 0,1...0,2 мм клиновидной конструкции. Для подключения конденсатора или катушки необходимо ввести выводы детали до упора в клиновидный зазор пластин; этим достигается быстрая и надежная фиксация выводов.

Налаживание прибора производят с помощью частотомера и осциллографа. Переключатель SA1 переводят в верхнее по схеме положение и подбором конденсатора С1 и резистора R1 добиваются частоты 1 МГц на выходе генератора. Затем переключатель последовательно переводят в последующие положения и подбором конденсаторов С2 - С5 устанавливают частоты генерации 100 кГц, 10 кГц, 1 кГц и 100 Гц. Далее осциллограф подключают к коллектору транзистора VT1, переключатель SA2 - в положении измерения емкости. Подбором резистора R3 добиваются формы колебаний, близкой к меандру на всех диапазонах. Затем переключатель SA1 снова устанавливают в верхнее по схеме положение, к гнездам Х6, Х7 подключают цифровой или аналоговый вольтметр, а к гнездам ХЗ, Х4 - образцовый конденсатор емкостью 100 пФ. Подстройкой резистора R7 добиваются показаний вольтметра 1 В. Потом переводят переключатель SA2 в режим измерения индуктивности и к гнездам Х4, Х5 подключают образцовую катушку с индуктивностью 100 мкГн, резистором R6 устанавливают показания вольтметра, также равные 1 В.

На этом настройка прибора заканчивается. На остальных диапазонах точность показаний зависит только от точности подбора конденсаторов С2 - С5.

От редакции. *Налаживание генератора лучше начать с частоты 100 Гц, которую устанавливают подбором резистора R1, конденсатор С5 не подбирают. Следует помнить, что конденсаторы СЗ - С5 должны быть бумажными или, что лучше, металлопленочными (К71, К73, К77, К78). При ограниченных возможностях в подборе конденсаторов можно использовать и переключение секцией SA1.2 резисторов R1 и их подбор, а число конденсаторов надо уменьшить до двух (С1, СЗ). Номиналы сопротивлений резисторов составят в этом случае: 4,7; 47; 470 к0м.*

И. ПОТАЧИН, г. Фокино Брянской обл. (Радио 12-98)

Так вот, почему я решил добавить эту схему — она, конечно, просто полезна. Кроме того, работу этой схемы, как и предыдущих, очень полезно было бы рассмотреть в программе Qucs. И, наконец, можно использовать генератор, как генератор прямоугольных импульсов, а добавив, измеряя с помощью приставки L и C, колебательный контур на частоту основного сигнала, получить синусоидальный сигнал. Кроме полезности эти схемы позволят провести ряд интересных, как мне кажется, и поучительных экспериментов.

А вот экспериментировать со следующей схемой я очень не советую. Наоборот, вспомните о мерах безопасности при работе с напряжением и постарайтесь, собрав схему на плате, тщательно проверив правильность монтажа, установив плату в предназначенную для нее коробку и подключив нагрузку, еще раз проверить, все ли закрыто и нет ли открытых проводов, и только после этого подключайте схему к силовой сети.

Собственно сама схема не предназначена к тому, к чему я ее предлагаю предназначить.

Схема регулировки яркости светильника

В предлагаемом устройстве используется так называемый фазоимпульсный способ регулирования среднего тока через нагрузку. Он изменяется благодаря тому, что нагрузка-светильник подключается к сети не непосредственно, а электронным ключом через некоторое время после появления очередной полу-волны сетевого напряжения. Изменяя это время, потребляемую нагрузкой от сети мощность можно регулировать практически от нуля до максимума. Для лампы светильника это означает изменение яркости ее свечения. При замыкании контактов выключателя S1 лампа L1 включается не сразу, а плавно в зависимости от емкости конденсатор C2. Это увеличивает срок службы самой лампы, т.к. мы знаем, что лампы обычно сгорают при включении - резком подключении напряжения.

Лампа L1 (220V 100W) собственно и является светильником. Все резисторы на 0,25W, кроме R8, который на 2W. При монтаже расположите этот резистор в 2mm над поверхностью платы, чтобы не нагревались остальные детали. Конденсатор C1 пленочный, тринистор КУ202Л можно заменить на КУ202К, КУ202М, КУ202Н. Соблюдайте условия его включения в схеме. Подключите неправильно - работать не будет.

В корпусе, в котором Вы разместите устройство, обязательно просверлите отверстия для вентиляции, т.к. элементы R8, VS1 в процессе работы нагреваются.

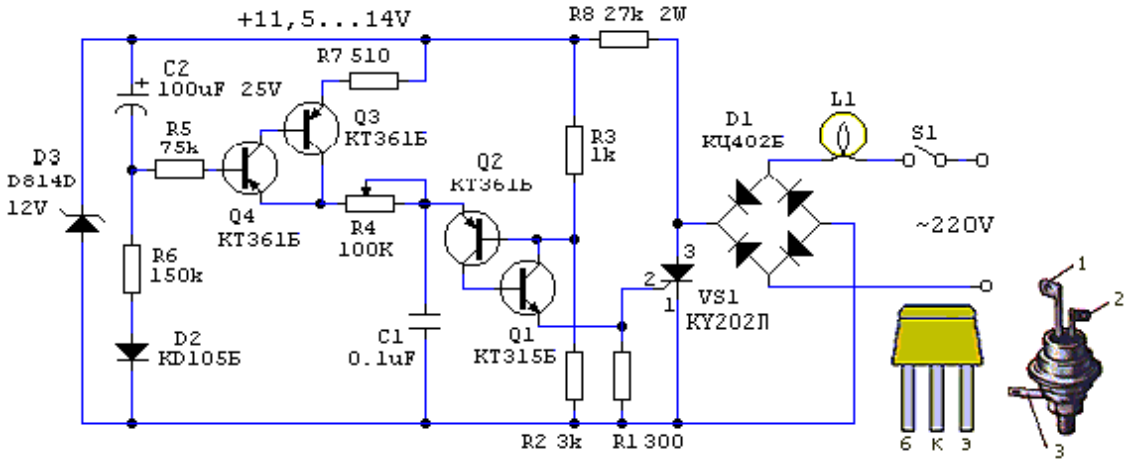


Рис. 17.6. Схема регулировки мощности в нагрузке

Источник: www.rp.butovonet.ru

К чему я в первую очередь вам предлагаю применить эту схему? К регулировке температуры паяльника, если вы пользуетесь обычным паяльником на 220 В. Хотя в оригинале нагрузкой служит лампа L1, мы уже договорились, что и лампа, и паяльник — не более, чем разного рода резисторы. А если так, то схеме все равно, какой резистор вы подключите. Работу этой схемы, я думаю, вполне можно проверить, если не в программе Qucs, хотя и там можно многое увидеть, то в программе PSIM. Очень советую для работы с паяльником между нагрузкой, паяльником L1, и выключателем S1 добавить предохранитель. Рабочий ток предохранителя, уверен, вы легко найдете, зная мощность вашего паяльника.

Если вы присмотритесь к этой схеме, то, как мне кажется, у вас должно появиться искушение заменить тиристор на триак. Прекрасный проект для работы и в Qucs, и в PSIM. Переносить его на макетную плату... а зачем? Схема должна работать и в ее первозданном виде. Но проверить разные варианты в программе, другое дело.

Глава 18. Организация собственной разработки

В профессиональной практике разработка начинается с создания технического задания. Оно должно определить, что в конечном счете должно получиться.

Радиолюбитель, начиная деятельность с ремонта своего плеера, рано или поздно приходит к необходимости собственной разработки, хотя он считает, что собирался только чуть-чуть подправить схему, взятую из журнала. Схему он повторил, схема работает, но немного его не устраивает.

И не столь важно, планируете вы использовать компьютер или нет, в любом случае начинать работу лучше с записи о том, что вы хотите получить в итоге. Это можно записать на листе бумаги, в блокноте, но если у вас есть компьютер, можно это сделать в любом текстовом редакторе.

Еще больше пользы принесет использование специализированных программных средств. В Linux для графической среды Gnome есть программа управления проектами *Planner*.

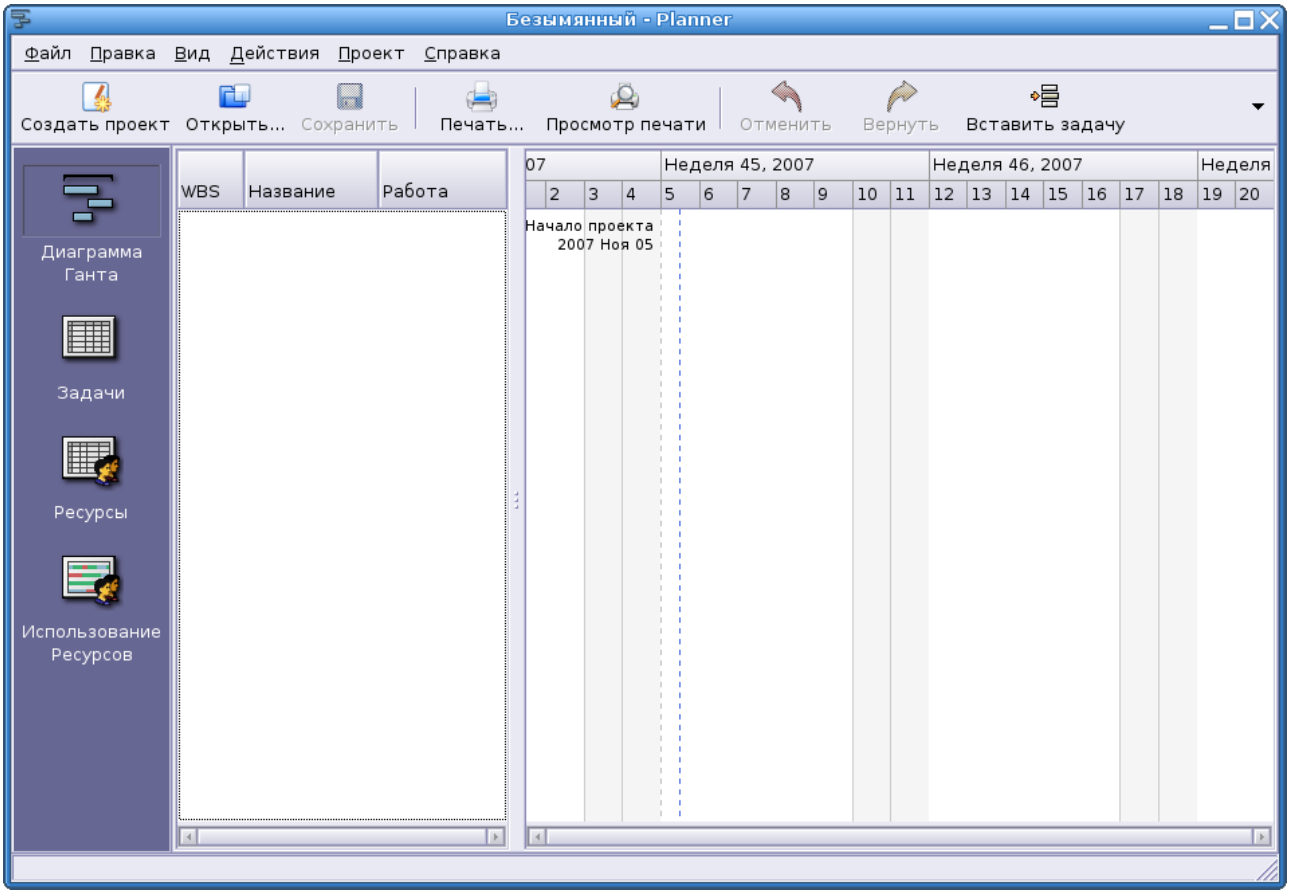


Рис. 18.1. Программа управления проектами

Это не единственная программа, аналогичная входит в состав, например, Koffice. Koffice — полный набор программ для офисной работы, разработанный для оконного менеджера KDE. Его можно использовать и в KDE, и в Gnome. Для офисной работы весьма важна ее организация. Но разве для любой работы это не так?

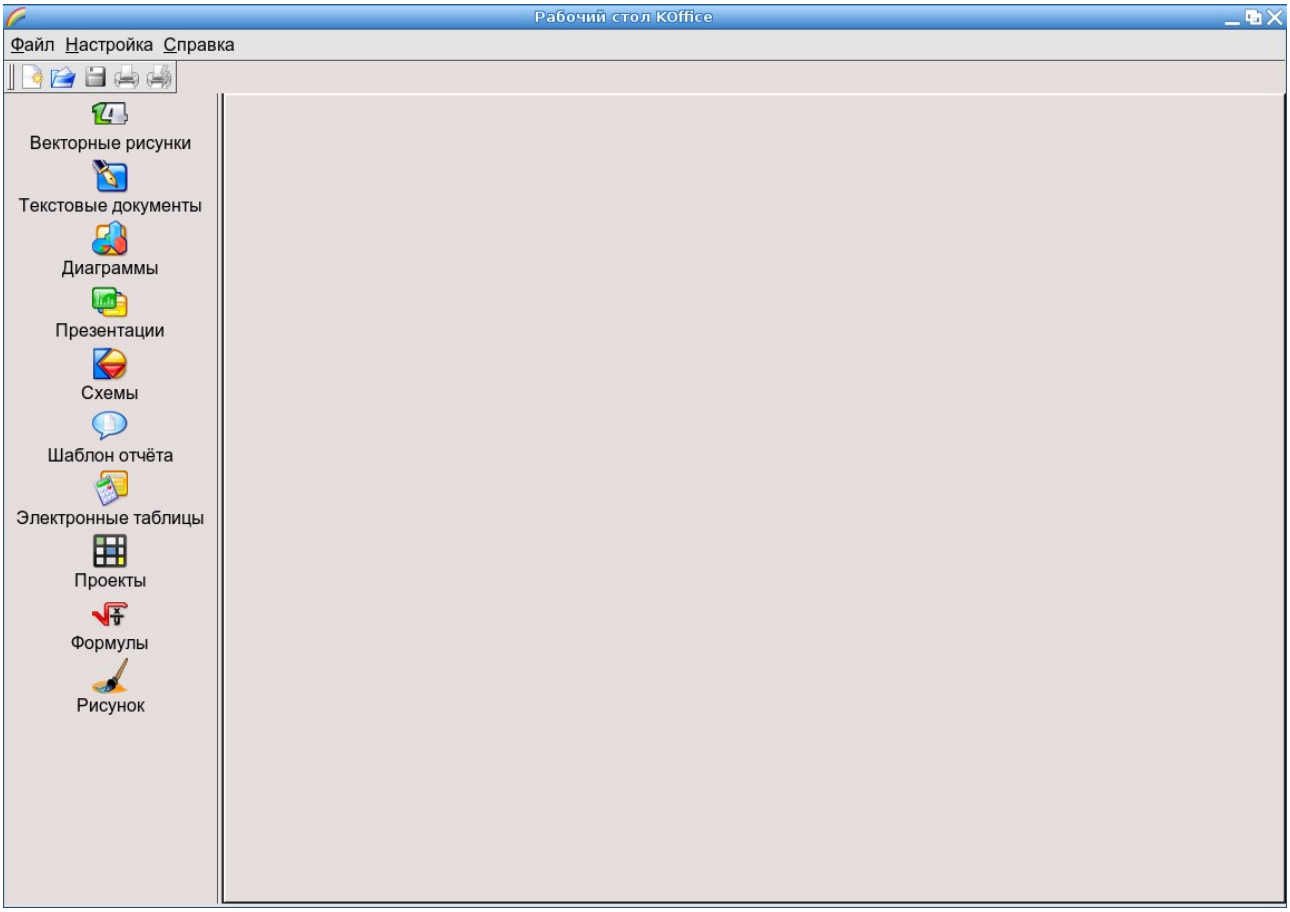


Рис. 18.2. Среда Koffice

В этом отношении, как и во многих других, любитель находится в более выгодном положении, чем профессионал. Последнему предписывается и форма, и порядок работы, принятые на предприятии. Тогда как любитель волен выбирать любые средства достижения цели, которые ему покажутся пригодными. Если вам кажется обременительным осваивать программу работы с проектами, используйте работу с текстовыми документами. Мне обычно бывает достаточно текстового процессора OpenOffice.org Writer для описания цели работы, создания плана работы и рисования небольших диаграмм. Здесь, пожалуй, самое важное выработать привычку к планированию своей работы. Не то, что без этого ничего не получится, нет. Работу вы все равно планируете, называете вы этот процесс планированием или нет, но вы планируете работу, только не записываете свой план в целях экономии времени. Как человек по натуре безалаберный, я благодарен всем своим учителям, научившим меня уважать процесс организации работы. И когда натура берет верх, когда, поленившись или забывшись, я не сохраняю промежуточных результатов работы, я впоследствии сильно ругаю себя за это. Записать, это несколько минут, а восстановить нужный вариант решения бывает очень трудно, уходят часы бесплодных поисков, особенно по прошествии нескольких месяцев, когда забывается все, что ты делал.

Начинающий любитель, для которого увлечение может превратиться в профессию, будет прав, если порасспросит профессионалов, а как у них организована работа. На многих предприятиях сегодня используются компьютеры и программы, помогающие осуществить работу. Не будет вредным использовать полученные советы. Если вы и передумаете, выберете другую профессию, уверен, сам процесс работы на другом поприще будет мало отличаться от того, что вы уже освоили. Любую работу нужно планировать, так или иначе

описывать цель работы, и в процессе работы составлять отчеты об уже сделанном.

Конечно, вам не нужно составлять отчет, но если вы возьмете за правило иметь рабочий журнал любого вида, куда вы будете заносить и свои задачи, и план работы, и результаты каждого этапа, вы заметите, что потраченное на это время окупается даже за время короткой разработки, которую вы и не называете разработкой.

Вот пример. Вы собрали схему для питания которой нужен источник напряжения 5 В. У вас есть блок питания от другого устройства, который вы можете использовать на время экспериментов, но на его выходе 12 В. Разумное решение, которое вы нашли сами или получили в подарок от другого, поставить пятивольтовый стабилизатор (готовую микросхему) на выход блока питания. Вы не планируете свою работу в явном виде. Но ваш план таков — поехать в магазин, купить микросхему, и поставить ее между блоком питания и своей схемой. Что вы и делаете.

Но вернувшись из магазина, впаяв микросхему, вы оказываетесь перед выбором: разбирать блок питания, чтобы припаять провода, или «разорить» провода от блока питания, чтобы подключить его к схеме. Или поехать в магазин еще раз, что самое разумное, для покупки разъема для подключения блока питания. Вы принимает правильное решение и едете в магазин.

Не то, что ваш план был плох. Вы сэкономили немного времени над его продумыванием. Если бы вы стали записывать свой план работы, вы непременно пришли бы к выводу о необходимости покупки разъема для подключения блока питания. На повторной поездке в магазин вы сэкономили бы пару часов, потратив несколько минут на продумывание плана работы. И это простой пример. В более сложных случаях все потери множатся многократно.

Работая с резисторами-транзисторами, это из личного опыта, на поиск подходящего сопротивления из собранных в одной коробке разномастных резисторов уходит от десяти минут до получаса. Достаточно несложная работа заставляет повторять этот процесс дважды или трижды. Минимальные затраты времени десятки минут. Разобрать все резисторы по номиналам, по мощности, по типам и разложить их в кассу требует времени. Но не более двух-трех часов. Если теперь поддерживать порядок, не бросать резисторы все вместе, а раскладывать по местам, когда нужда в них прошла, то каждый день вы будете экономить время для занятия более плодотворного, чем перебирание деталей в поисках той, которой может и не быть.

Мне нравится начинать пайку макета — разложить на столе новенькую плату, новенькие микросхемы и детали. В ожидании, когда паяльник разогреется, я с нетерпением ожидаю начала процесса. Я много раз давал себе слово, что даже собирая несложную схему, воспользуюсь программой разводки печатной платы. Я не собираюсь делать именно печатную плату, но рисунок печатной платы и компоновки всех элементов схемы полезен хотя бы в том, что являет собой план работы. Отвлекаясь в процессе пайки на перекур или обед, обязательно делаешь ошибки. На создание монтажного плана в виде рисунка печатной платы уходит пусть час, пусть два. А на отыскание ошибок в монтаже, когда схема вроде бы и работает, а вроде бы и нет, на отыскание таких ошибок может уходить и несколько дней работы.

Монтажная схема, алгоритм программы, план проекта — это инструменты. Они могут быть очень полезны, если вы умеете ими пользоваться, они будут полезны, если вы их используете, даже не освоив их полностью, но уж точно они будут бесполезны, если вы не будете ими пользоваться совсем.

Раскладывая детали на столе, я много раз давал себе обещание, что в следующий раз... Да, любой следующий раз должен начинаться сразу, иначе его не будет никогда. Потерянный

винт, для которого несложно подобрать подходящую коробочку, резистор, который вы смахнули со стола, когда обернулись, вам знакомо это? А достаточно было не в следующий раз, а именно в этот взять небольшой кусочек поролона, воткнуть в него все резисторы и транзисторы, и вы опять бы избежали ненужной поездки в магазин за единственным резистором, которого вам не хватило для завершения монтажа.

Особенно большое разочарование у меня вызывают промахи, связанные со схемами. Зачем вносить исправления по ходу налаживания, если можно исправить все разом, когда схема будет полностью отлажена. Когда схемы существовали только на бумаге подобный подход оправдывался тем, что после нескольких исправлений схема превращается в «грязное месиво» из прошлых ошибок. Но теперь схема чаще всего появляется на компьютере, где исправление занимает не столь много времени. Однако привычка берет верх. А когда схема полностью отлажена, все исправления на макетной плате сделаны, вы готовы внести их в первоначальный вариант? Вы помните их все? Если нет, то теперь уже по готовому монтажу приходится обходить всю схему, чтобы внести правку, но, когда позже вам приходится обращаться к этой исправленной схеме, вы к собственной радости можете обнаружить множество «недобитых» в прошлый раз ошибок, которые заставляют вас заниматься налаживанием уже налаженной и работающей схемы.

В этом смысле рабочий журнал, в виде ли текста выполненного в редакторе, в виде ли вспомогательного файла, который позволяют сделать многие среды разработки, в виде ли тетради в клеточку — в любом виде рабочий журнал, в котором вы записываете план работы на день, в котором записываете то, как и насколько выполнен этот план, свои соображения и идеи, возникающие по ходу любой работы, такой журнал поможет избежать ошибок и восстановить работу, если вы даже выбросите макет или готовое устройство.

Самое полезное, ведь у многих, имеющих компьютер, есть и принтер, самое полезное иметь не только файлы, но и бумажную копию этих файлов. Папки проектов не займут много места, если вы в них вложите схемы, описания работы схемы, заметки по особенностям монтажа и т.д. Да, сегодня вы закончили проект и искренне полагаете, что возврата к нему не будет. Но сегодня, как известно, заканчивается, и наступает завтра.

Так получалось, что временами мне приходилось иметь дело с устройствами, пусть не сложными, но не имеющими ни монтажной, ни принципиальной схемы, не говоря уж про описание работы. Каждый раз я искренне считал, что это разовый случай, что больше я с этим никогда не столкнусь. Все фрагменты схем, сделанные на клочках бумаги, все мои замечания выполненные неразборчивыми каракулями, либо отправлялись в мусор по окончании работы, либо оставались в том виде, в каком появлялись. И сколько раз, и как я себя ругал за это впоследствии, если приходилось вновь сталкиваться с той же проблемой.

Вместе с тем, бывали периоды прояснения в моем отношении к работе. Я помню, сколько удовольствия я получал не от того, что не поленился сохранить все записи, а от самого процесса разумно организованной работы. Я не очень люблю, скажем, пайку. Не потому, что этот процесс мне чем-то неприятен, отнюдь. Мне нравится придумать что-нибудь, быстро «прикинуть», как бы это можно было сделать, после чего интерес к работе пропадает. Любой процесс, отодвигающий конечный результат, как продумывание реализации схемы, или планирование работы, или создание макета, любой «посторонний» процесс в моих глазах приобретает вид личного врага. Не люблю я их всех. Но, когда это первое ощущение уходит, когда ты, поборов свою неприязнь к «канцелярщине» всех мастей, садишься за описание задачи, в виде технического задания или в виде некоторого проспекта, ты начинаешь понимать, что этот процесс имеет самостоятельную привлекательность. По ходу дела возникают новые идеи, соображения, появляется новый взгляд на конечную цель, например, в плане применения того, что задумывалось только для решения одной единственной задачи.

Схожее ощущение появляется у вас и тогда, когда вы начинаете задумываться над монтажной схемой. Особенно, если воспользуетесь программой автоматизированной разводки печатной платы. Особенно, если есть возможность увидеть трехмерное воплощение платы. Вы начинаете понимать, что этот процесс интересен сам по себе. Не только значим, порой от правильной компоновки зависит работоспособность схемы, но, действительно, интересен.

Для среды Linux существует, по меньшей мере, несколько доступных программ автоматической разводки печатных плат, как KiCAD, Eagle и gEDA.

Привычка организовывать свою работу определенным образом приходит быстро, особенно если не утруждать себя выбором между всеми возможными вариантами. Если мне приходится общаться с «живым» устройством я чувствую себя не в своей тарелке при отсутствии осциллографа. Привык. Но, поразмыслив, я понимаю, что далеко не всегда он нужен.

От того, как вы, начиная общение с электроникой, определите для себя необходимые для организации работы средства, зависит главное, получите ли вы, и сколько, удовольствия от этого. Постарайтесь извлечь максимум удовольствия. Для этого есть все необходимое.

Часть 2. Игра в программирование

Не получается у меня задумка. Не работает схема. Все, вроде бы, правильно, но не желает работать это «транзисторно-микросхемное месиво». Раньше было проще – измерил напряжения, посмотрел осциллографом, все понял и исправил. Так то было раньше.

Не то, чтобы я испереживался, не спал по ночам. Сплю, и вам советую. Спите крепко, чтобы проснувшись назавтра порадоваться новому дню и солнцу, если оно светит, или дождю, если на улице пасмурно. Как бы плотно ни укрыли тучи ваш город, солнце-то продолжает светить!

Любая деятельность человека всегда связана с периодами усталости или разочарования, с тем, что предмет его внимания начинает капризничать, становится непостижимым образом похож на дикобраза – кругом колючки, как ни подступись. Даже если то, чем вы занимаетесь, не ваша работа, а ваше увлечение, ведет оно себя временами столь же противно, что и каждодневная рутина. Бесполезно браниться, бесполезно сердится, полезно, но не всегда, к сожалению, помогает, обратиться за советом или помощью. Важнее всего помнить, что как бы плотно ни укрыли тучи...

Программирование, как обязательный курс, по которому приходилось сдавать экзамен, может больше напугать, чем заинтересовать. Между тем, все больше людей, никак с программированием прямо не связанных, или не связанных с ним прежде, подвергается, я бы сказал, «прямому шантажу» со стороны этого явления природы.

Я не программист. И не пытаюсь примазаться к этой славной профессии. Но и нет моей вины в том, что время от времени появляется необходимость возвращаться к давно забытым со студенческих времен понятиям процедуры, функции, цикла... Однако сегодня, когда мне не нужно сдавать экзамены, возвращаясь к этим пугающим нормального человека понятиям, я начинаю с грустью думать об их простой и ясной сущности. И происходит это потому, что сегодняшнее программирование изобилует новыми таинственными сущностями – класс, наследование, контейнер и т.д. Но, признаться, если не обращать на них внимания, то современные системы программирования позволяют сделать легко и быстро многое из того, что прежде отнимало много времени и сил. Сегодня я не могу сказать, что проще – написать небольшую программу, которая решит ваши проблемы, или с карандашом и бумагой проделать эту же работу? Обычно первая мысль о бумаге и карандаше кажется более разумной, но лишь до тех пор, пока не примешься претворять эту мысль в жизнь.

Я не умею программировать, и компьютер для меня сродни кофеварке, удобно и включается быстро, но я хочу предложить такую игру: я сделаю вид, что я умею программировать, что знаком с компьютером не по наслышке, а вы сделаете вид, что верите мне, – посмотрим, сумею ли я рассказать что-нибудь полезное о программировании. Это будет не более, чем игра, и я не буду, выскакивая из-за угла, пугать вас громкими криками: «Грядет итератор!».

Для игры в футбол есть футбольное поле, для игры в гольф – поле для гольфа. Для игры в программирование я предлагаю великолепнейшее поле – язык и среда программирования Gambas. Он имеет в своей основе классический язык Бэйсик. Работает в среде операционной системы Linux. Если по какой-то причине вы не хотите или не можете пользоваться этой операционной системой, то можете пользоваться средой программирования Visual Basic. Они достаточно похожи.

Для тех, кого смущают родственные отношения между Basic и Gambas, я готов привести множество высказываний опытных профессионалов о том, что программирование не соотносится напрямую с языком кодирования. В этом смысле нет ничего зазорного, что

основой языка служит Basic, тем более, что Gambas – это современный, мощный и хорошо построенный объектно-ориентированный язык программирования. А насколько он хорош, можно судить по тому, что вся графическая пользовательская оболочка Gambas написана на самом языке Gambas.

По мне же, чем проще, тем лучше. И лично меня смущает только одно, достаточно ли это просто. С выяснения этого я, пожалуй, и начну.

Спасибо, что дочитали это предисловие до конца, а если нет, то и ничего страшного.

Глава 1. Поиск печки, от которой танцевать

Под печкой в любительской практике будем на протяжении этой главы понимать удобные программные средства, которые легко использовать в практике радиолюбительства с не меньшим успехом, чем в профессиональной.

Два берега

Если сравнивать программирование с рекой, то конкретная платформа или язык программирования – это берега могучей и широкой реки. К какому берегу вам причалить, к какому месту на берегу – ваш выбор! Плыть ли по реке на скутере, побивая рекорды скорости, или неспешно перемещаться, отдавшись на волю течения и уютно расположившись в палатке, установленной посреди плота, и наслаждаться видом окрестностей – ваш выбор!

Я отдаю предпочтение одному берегу – платформе Linux и среде программирования Gambas, основанной на языке Бейсик, вы можете сделать другой выбор. Программы, написанные в среде Visual Basic и Gambas, мало чем отличаются друг от друга. Более того, даже использование Delphi потребует скорее не переделки программы, а некоторой правки текста. Конечно, я не могу сказать этого о любой программе, но только о небольших проектах, с которыми собираюсь иметь дело.

То же будет относиться и к среде программирования микроконтроллеров, когда до этого дойдет дело. В Windows можно использовать MPLAB, в Linux – Piklab. В этом случае даже правки кода не потребуется, обе среды программирования используют одинаковый ассемблер, а программирование на языке «С» можно осуществить с использованием компилятора Hi-Tech, бесплатная версия которого (с ограничением по памяти) есть для обеих сред программирования.

Я уже говорил, что не буду пытаться научить кого-либо программированию, для этого есть опытные программисты, опытные преподаватели, я хочу только рассказать о пользе навыков в программировании с пониманием основных подходов к этому процессу даже для тех, кому нет нужды постоянно заниматься программированием. И, главное, что само программирование – увлекательнейшее занятие для любого, у кого есть компьютер и остается хотя бы немного времени, чтобы включить его и заняться чем-то менее бессмысленным, чем стрельба «из мышки по воробьям».

Много лет назад, когда мой сын был еще ребенком, я собрал для него, с большими приключениями, домашний компьютер, который назывался ZX-Спектрум. Когда сегодня я вспоминаю этот компьютер, то, возможно я ошибаюсь, мне кажется, что он был гораздо интереснее, чем мой сегодняшний рабочий Pentium. Он был гораздо проще, и было много легче разобраться с необходимыми переделками, если такая нужда возникала. Помню, когда появилась возможность подключить к нему дисковод, а первоначально этот домашний компьютер записывал все на магнитофон, и появились программы, которые загружались с дискет, то многие из них были не полностью переделаны под работу с дисководом. Так компилятор языка «С» действительно загружался с дискеты, но при попытке сохранить текст программы обращался к магнитофону. Исправление этого недостатка доставило мне больше удовольствия, чем неприятностей. То же касается и игрушек, которые не работали, но зачем-то обязательно были нужны сыну. Для этого же компьютера мы с ним написали первую законченную программу, которая позволяла научиться печатать «вслепую». Не знаю, остались ли у него приятные воспоминания о программировании, но помню, что завершив работу над программой, мы поняли, что учиться печатать на той клавиатуре, собранной из

«подручных средств», которая у нас была, не имеет смысла, но... позже, когда появился компьютер IBM PC, я восстановил эту программу, правда, убрав все излишества, для нового компьютера, и мы оба учились печатать, используя эту программу, поскольку других обучающих программ тогда не смогли найти. И мы оба по сей день пользуемся полученным навыком, что экономит много сил и времени и мне, и сыну, когда приходится что-то печатать. В тот раз я понял две важные вещи: во-первых, при желании можно написать любую программу, которую есть необходимость написать; во-вторых, что многое отнимает гораздо меньше времени, чем это может показаться на первый взгляд. Это касается овладения печатью «вслепую». Мне казалось, что пройдет не меньше года, прежде чем я смогу печатать глядя на экран, а не на клавиатуру, и использовать все пальцы для этого процесса, а не обходится двумя. Я ошибался. Через две-три недели я печатал, печатал «вслепую», а через месяц печатал и на русском, и на английском, переходя с языка на язык пусть не мгновенно, но без особых проблем. А возможности просиживать за компьютером часами тогда у меня не было. Только по 15-20 минут каждый вечер. К сожалению впоследствии я перестал пользоваться печатью латиницей хотя бы для сохранения умения, да и по-русски писать приходилось крайне редко, но последний навык все-таки сохранился. С программированием, кстати, такая же история. Я не программист, но когда приходилось этим заниматься по тем или иным причинам, я вынужденно овладевал какими-то базовыми подходами. Эти знания впоследствии не единожды меня очень выручали. Мне ни разу не пришлось пожалеть о потраченном на их приобретение времени, а если и приходилось о чем-то пожалеть, так это о том, что я мало времени потратил на изучение программирования.

Я не скажу, что сейчас, если возникает необходимость обратиться к программированию, я испытываю восторг от предвкушения предстоящей работы. Нет. Я отдаю себе отчет, что будет необходимость обращаться к документации, к книгам, к подсказкам среды программирования. Я прекрасно понимаю, что наделаю много ошибок, с которыми придется как-то справляться, что ни раз буду чертыхаться и злиться (на кого?), разочаровываться и начинать все заново, но за этими неприятными эпизодами, как за занавесом театра, будет скрываться увлекательнейшая история. Я точно знаю, что временами очень сердился на все и на вся, но когда позже пытался вспомнить, что же происходило на самом деле, то не мог вспомнить ничего кроме удовольствия от «возни» с непокорными программными сущностями.

У меня нет пристрастия к какому-либо языку программирования в силу дилетантского подхода с одной стороны, и в силу разных обстоятельств с другой. В один из моментов работы потребовалось, а исходные коды имелись в наличии, дописать программу для временной демонстрации работы устройства. Позже это все предполагалось сделать профессионально, иначе, и на другом языке программирования. Программа, написанная на «C++», была документирована, что называется, «для себя», и я точно знал, что с этой задачей мне не справиться. Хотя бы потому, что работать с чужой, достаточно большой программой – занятие безнадежное и очень трудоемкое. Но, начав работать, я с удивлением отметил, что, отдавшись на волю течения, продвигаюсь вперед. Конечно, не дело, «лезть в воду, не зная броду», но компьютер удивительное создание. Если не делать ничего сверх необходимого, он всегда выручит, и при этом можно не опасаться фатальных последствий.

Продолжая аналогию с рекой, но считая рекой электронику, то два ее берега превратятся в программирование и «собственно железо». Достаточно давно электронные устройства имеют в своем составе нечто, что так или иначе программируется. И сегодня, занимаясь электроникой, трудно обойтись без нежданных встреч с программированием. К этому надо быть готовым, и я еще раз хочу подчеркнуть, пусть меня поправят, если я искренне заблуждаюсь, не следует путать программирование с конкретным языком программирования или той или иной средой программирования. Совсем не обязательно быть знатоком языка

«Си-шарп», чтобы иметь возможность начать работу над программой. Совсем не обязательно ставить своей задачей освоить в полной мере ассемблер, если вы хотите использовать микроконтроллер в своей практике. В первую очередь по той причине, что каждый конкретный микроконтроллер будет иметь свой ассемблер. Гораздо проще поискать такой контроллер, такую среду программирования контроллера, для которой есть язык высокого уровня. Часто это язык «С». И из самого языка вы, скорее всего, будете использовать только часть возможностей, понять и освоить которую гораздо легче и быстрее, чем язык полностью. Я не имею ввиду профессионалов, у которых другой подход и иные критерии выбора.

Если продолжать аналогию с рекой, но в качестве реки принять ваше занятие электроникой, то берега превратятся в два основных вопроса, которые всегда будут сопровождать вас – что проще, выполнить работу не прибегая к помощи программирования и элементов, связанных с программированием, или программировать?

В качестве примера того, как подступиться к программированию, я намереваюсь написать (по ходу написания книги) программу, которая будет обслуживать микроконтроллер, который, в свою очередь, должен проверить двоичный счетчик с предустановкой. Это простенькая задачка – проверка счетчика. Давайте попробуем присмотреться к обоим берегам.

Как я стал бы действовать, причалив к «железному» берегу реки?

Для задания числа можно подпаять соответствующие входы к общему проводу и плюсу питания (через «подтягивающие» резисторы). Ко входу переписи припаять любой «микрик» – небольшой кнопочный переключатель, включив его соответствующим образом, чтобы сформировать фронт сигнала переписи. Аналогично можно было бы поступить и с входом счета, если бы не одно но – дребезг контактов. Дребезг вызовет серию импульсов вместо одного единственного. У некоторых импульсных генераторов есть режим генерации одиночного импульса, но такого генератора у меня нет. Придется спаять схему, предотвращающую дребезг контактов. Ну, а паять, так паять! Для проверки состояния выходов я могу на выходы припаять светодиоды, что избавит меня от необходимости проводить каждый раз по несколько измерений мультиметром. Припаяв светодиоды с токоограничительными резисторами, я нарисую в рабочей тетради таблицу-заготовку для полученных значений, и все. Я готов проверить работу счетчика. Не слишком меня это «напрягло». Правда, сделав пару циклов, я опять берусь за паяльник, чтобы изменить число на входе предварительной установки. Но и это не слишком сложно. Беспрограммный берег реки полог, порос зеленой сочной травой, красив и живописен.

Единственное, что вызывает у меня сомнения, а не болото ли это? Я проверил работу счетчика в пошаговом режиме на очень низкой частоте переключений. Не возникает ли проблем при ускорении работы? Проверить этот аспект я тоже могу. Достаточно импульсом обнуления счетчика запустить небольшое устройство, которое сформирует мне импульс переписи, затем засинхронизироваться этим импульсом и посмотреть на осциллографе результат работы счетчика при тактовой частоте, положим, в один мегагерц. Паять, так паять!

Лежа на плоту, плывя по течению реки, я посматриваю то на один берег, то на другой. Один я описал выше. А как выглядит другой?

Крутой и неприступный, отчего он кажется неприветливым, берег порос деревьями столь густо, что сквозь поросль не видно ни одной тропинки. От того, что солнце бьет прямо в глаза, заросли кажутся еще более мрачными, темными и непролазными. Ни птица не порхнет, ни зверь не шелохнется. Только неясные тени виртуальных программистов осмеливаются потревожить покой чащобы.

И что за напасть! Прибило меня течением именно к этому берегу.

Первое знакомство с Gambas

Не будучи представлен человеку, желая набиться ему в знакомые, ты никогда не начнешь с жалоб на свои личные проблемы. Самое лучшее, что можно сделать при первом знакомстве – это узнать у человека, как идут его дела, пригласить его выпить чашечку кофе, а за кофе поговорить о погоде, поискать общих знакомых.

Набиваясь в знакомые языку программирования я хочу следовать тому же правилу. Вначале поинтересуюсь о его делах (законспектирую руководство), потом поговорю об общих знакомых (например, языках программирования, или программах). И только после этого попробую обратиться к своим проблемам с его помощью.

Вы не поверите, но набиваться в знакомые не так-то просто, конечно, если не полагаться на случай, на везение. Для подготовки к предстоящему разговору я, вопреки обыкновению, отправился в книжный магазин. Обилие общих знакомых, дружными рядами оккупировавших полки, не сулило скорого продвижения в работе, предстояло перечитать все названия книг, чтобы попытаться найти ту, которую я хотел купить. А завершив процедуру, я, вдобавок, не нашел нужной. Но это только по причине моей привередливости, книг по программированию очень много. Еще можно поискать в Интернете.

И Gambas. В папке «help» кроме традиционных подсказок большое количество статей, относящихся к языку, к объектно-ориентированному программированию. Много примеров, над созданием которых потрудился не только автор, но и многочисленные добровольцы. Думаю, если бы я искал прямой путь к цели, этого хватило бы, чтобы вымостить широченный проспект, а не только подъездную дорогу.

Немного о Linux. Запуск программы, как во многих сегодняшних, если не всех, операционных системах, осуществляется щелчком левой клавиши мышки в соответствующем разделе меню. В Linux меню выбора программ хорошо систематизировано – офисные приложения в разделе «Офис», у меня там много всего, потому что я люблю, когда всего много, когда есть из чего выбирать.

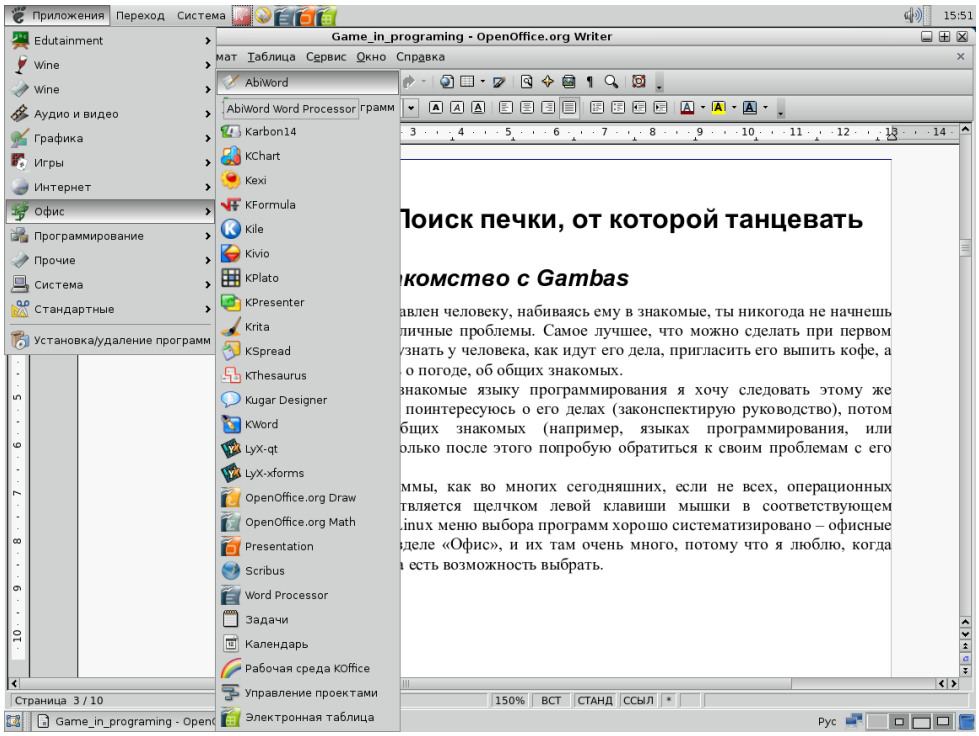


Рис. 1.1. Раздел офисных приложений в операционной системе Linux

Для работы только с текстом я часто пользуюсь текстовым процессором AbiWord. Он легкий и быстрый. Если приходится рисовать что-то в тексте, исключая необходимость создания чертежей профессионального качества, можно использовать KOffice, но чаще я пользуюсь OpenOffice.org.Writer. Из него я могу экспортировать текст с иллюстрациями в форматы «doc», «pdf», «html».

Не менее насыщенным выглядит у меня раздел приложений для программирования. Здесь несколько великолепных сред программирования, позволяющих работать с разными языками, на них я поглядываю с тоской и завистью. Есть программы создания web-сайтов. Свой сайт, в его последней ипостаси, я создавал с помощью Nvu. Времени заниматься сайтом было мало, и не умею я это делать, сайт вышел весь в меня – страшненький, но работающий. Смотрю в меню, и недоумеваю, откуда это накопилось? Недавно обновлял операционную систему, давал себе слово, что все по минимуму! Не понимаю.

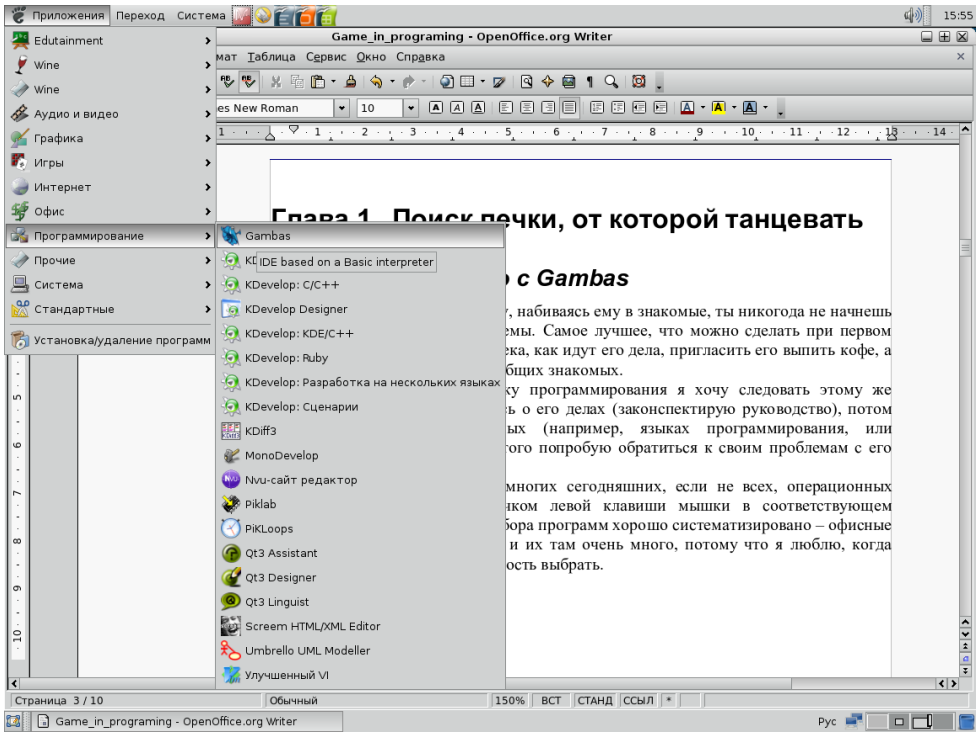


Рис. 1.2. Раздел «Программирование» в Fedora Core 6

И очень мне хочется похвастаться (не могу оказать себе в этом скромном удовольствии) теми возможностями анимации и трехмерного вида рабочих столов, которые есть в последней версии Fedora Core 6 (сегодня, завтра уже Fedora 7). Когда я устаю работать, но еще не начинаю дремать у телевизора, я включаю все эти красоты и начинаю просматривать почту, новости в мире Linux, форумы, где больше читаю, чем пишу, по той простой причине, что едва успеваю прочитать все то полезное, что накапливается за день.

Еще хочу добавить, что трехмерные и возможности анимации не относятся непосредственно к моему дистрибутиву. Нет. Второй дистрибутив на моем компьютере – это Ubuntu. Хотелось с ним познакомиться в работе, чтобы порекомендовать своему другу в качестве основной операционной системы. Так вот, в Ubuntu все эти красоты работают не менее успешно.

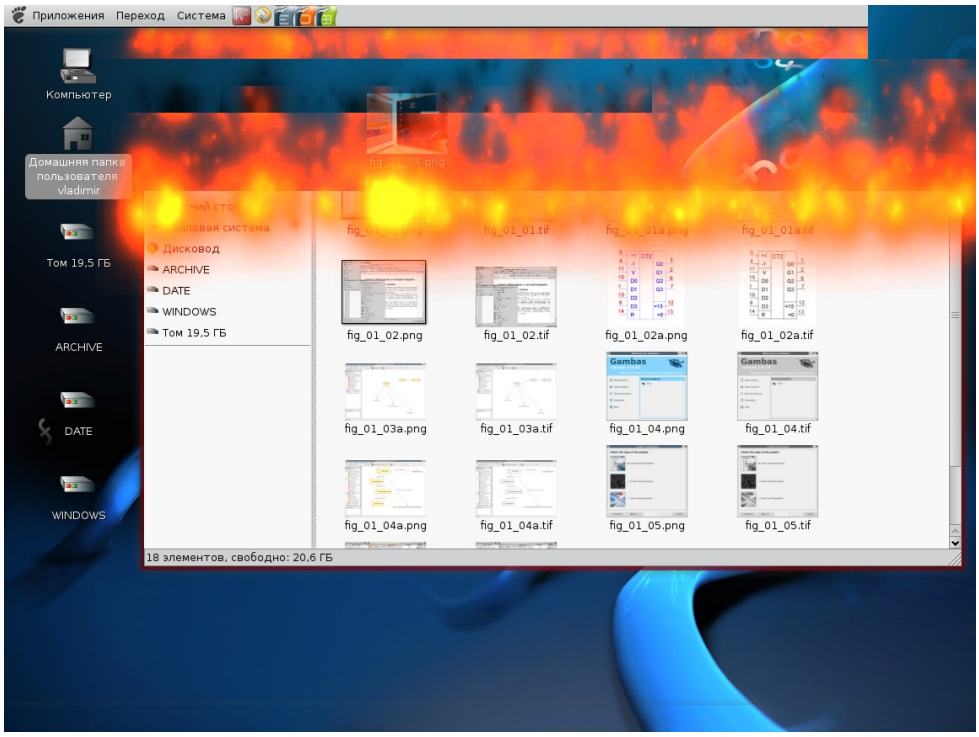


Рис. 1.3. Закрывание домашней папки с поддержкой Beryl

Я бы сказал, что операционная система просто «горит на работе»! И особенно мне этот фокус нравится, когда я закрываю логический раздел диска, где у меня установлен эмулятор Windows для Linux – Wine, закрывая его думаешь: «Вот и сожжены мосты!». В разделе установлено несколько «free» и «demo»-программ EDA для Windows, которые я использовал при работе над предыдущей книгой. По завершении хотел было снести раздел, но программы мне понравились, и я решил оставить их вместе с Wine на разделе Windows. Собственно, я не противник этой операционной системы, просто дороговаты для меня и система, и программы. И поскольку уж зашла об этом речь, все что будет сказано о конкретных программах, которые я, если получится, напишу в Gambas, а язык стоит того, чтобы постараться, так вот, все это можно повторить в Visual Basic, если для кого-то он кажется более подходящим.

Одна из особенностей операционной системы Linux – наличие нескольких рабочих столов, их может быть один или два, часто после установки их бывает четыре, но вы можете легко задать нужное именно вам количество рабочих столов. Я так привык использовать несколько рабочих столов, что не представляю себе, как умещался раньше на одном единственном?

Сегодняшние дистрибутивы позволяют перемещаться между столами, используя трехмерный куб (и количество граней можно, похоже, увеличивать).

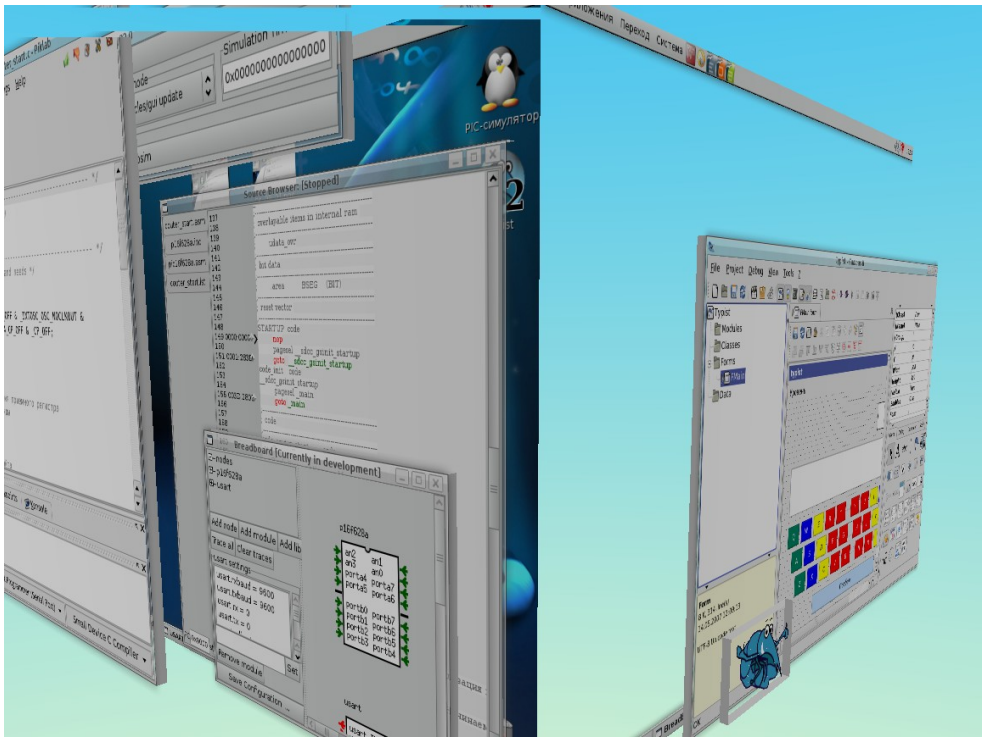


Рис. 1.4. Куб рабочих столов

В глазах многих, как мне кажется, Linux выглядит, как операционная система для профессионалов. Этому представлению может способствовать и посещение форумов, связанных с платформой Linux. Множество советов, как добиться желаемого результата, начинаются словами: «Открой скрипт...», или «Дай команду...». Это правильные советы. Это универсальные (или почти универсальные) советы, обязанные большой гибкости и открытости Linux. В ней можно переделать все, все изменить, настроить, что угодно и как угодно. Важно только знать как это сделать, и разбираться в последствиях своих действий. Многим нравится работать даже с последними версиями без графической оболочки, используя команды, вводимые вручную. Но я, например, только ленивый пользователь. Я не склонен изучать и совершенствовать операционную систему. Я хочу ею пользоваться, выполняя свою работу или потакая своим прихотям. Позволяет ли Linux оставаться пользователем и только пользователем, и никем, кроме пользователя. Думаю, да.

Простой пример: сегодня в Москве многие владельцы компьютеров подключаются к домовым компьютерным сетям, подключение к Интернету проходит через vpn-сервер провайдера. Если в Windows все настройки кажутся простыми (что не совсем так), то создание подключения в Linux может показаться занятием «не для слабонервных». Как выглядят рекомендации, которые в изобилии встретишь на форумах:

> *«Убедитесь, что в системе установлен пакет **pptp-linux-1.2.0-1asp.i386.rpm** и если нет, то установите его со второго диска дистрибутива. Также проверьте, что у вас уже установлено соединение с локальной сетью, используя команду **ifconfig eth0**. Если соединение есть, то вывод команды будет выглядеть примерно так:*

```
[user@localhost user]$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:11:22:A3:B4:C5
          inet addr:192.168.1.20  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:0 (0.0 Mb)  TX bytes:0 (0.0 Mb)
      Interrupt:3 Base address:0xa400
```

*Создайте файл **/etc/sysconfig/network-scripts/ifcfg-pptp0** приблизительно такого содержания:*

```
PEERDNS="no"
DEVICE="pptp0"
ONBOOT="no"
USERCTL="yes"
PERSIST="no"
DEBUG="yes"
DEFROUTE="yes"
PPPOPTIONS="require-mppe require-mppe-128"
MRU=""
MTU=""
IDLETIMEOUT=""
VPN_HOST="host_name"
VPN_USER="username"
ROUTES=""
```

где VPN_HOST – это имя или ip-адрес VPN-сервера, а VPN_USER – имя, под которым вы известны VPN-серверу...», и т.д.

Видимо, это правильные рекомендации. Когда лет пять назад я купил первый дистрибутив Linux, то обратился с проблемой подключения по VPN в отдел технической поддержки дистрибутива, где получил такого же рода советы. Однако, повторяя все операции шаг за шагом, я так и не смог настроить подключение. Позже я нашел удобный графический вариант создания vpn-подключения с помощью программы (или сервиса?) Webmin, что было, возможно, не «по науке», но работало.

Подключение к Интернету в последнем дистрибутиве Fedora Core я делал с помощью Webmin. Создание подключения мало отличается от аналогичного в Windows и требует ввода почти стандартной информации о vpn-сервере (но требует IP адреса сервера), данных доступа и т.п. Но без ложки дегтя не обходится.

Не столь давно, установив в качестве второй операционной системы Ubuntu, еще один дистрибутив Linux, я настраивал подключение по VPN к Интернету в программе Kvpnc. Как это выглядит, чтобы легко было сравнить эти настройки с Windows, я сейчас покажу, предварительно уточнив, что Ubuntu я ставил с LiveCD (один загружающий систему CD-диск с возможностью установки системы в логический раздел жесткого диска) с графическим менеджером Gnome. Для работы Kvpn дополнительно потребовались некоторые библиотеки, относящиеся к другой графической оболочке. Если бы я использовал Kubuntu с графической оболочкой KDE, то установка дополнительных библиотек, возможно, не потребовалась бы.

Основная проблема возникает из-за необходимости скачивания файлов через Интернет, к которому ты еще не подключен. В частности, при установке Ubuntu сама установка застревает на настройке менеджера обновлений, который долго ищет подходящие зеркала для обновления дистрибутива, хотя подключение к Интернету еще предстоит создать. Обычно я отключаю сеть, чтобы закончить установку. Network Manager имеет раздел подключения по vpn, но тоже с дополнительными файлами. Я как-то попытался выполнить эту настройку, успеха, увы, не добился. После установки первой для меня версии Ubuntu я, используя основную свою ОС Fedora, скачал программу pptp-config со всеми необходимыми библиотеками и дополнительными файлами. Удобство этой программы для меня в том, что

дополнительные файлы, которые обычно связаны зависимостями, имеют разрыв в этих зависимостях и позволяют найти конец цепочки, с которого последовательная установка проходит.

Для соединения с Интернетом по VPN можно было бы использовать после настройки этот графический интерфейс. Но программу следует запускать с правами root из терминала, либо сделав значок запуска, и вдобавок, я не знаю, как организовать ввод двух необходимых строк для маршрутизации без использования ручного ввода в терминале после того, как соединение установлено.

После установки, настройки и первого запуска этой программы (с вводом строк маршрутизации) появляется возможность установить Kvpnc. Для этого я вначале устанавливаю Kpdf.

Зачем? Затем, чтобы установились все необходимые библиотеки для работы Kvpnc. Затем Kvpnc устанавливается и можно запустить настройку, как это показано ниже.

После установки программы необходимо создать новый профиль (аналогично новому подключению), что лучше всего сделать с помощью мастера создания этого профиля.

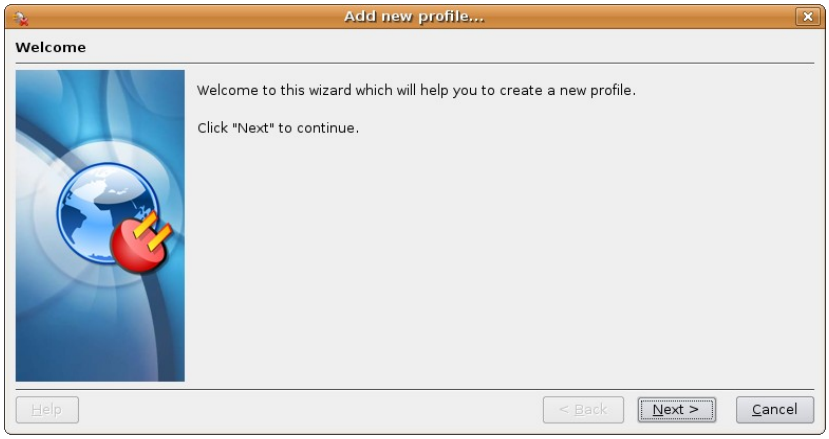


Рис. 1.5. Запуск мастера создания VPN подключения в kpvnc

Рисунки, которые вы видите, относятся к дистрибутиву Ubuntu. Можно бы все проделать в Fedora 7 (последний дистрибутив моей основной ОС), но я придерживаюсь правила, не создавать дополнительных трудностей без особой необходимости. Интернет настроен и работает, зачем переделывать?

Рис.1.6. Продолжение настройки в диалоге создания VPN подключения

Едва ли мастер подключения сильно отличается от аналогичного в Windows. Как видно из рисунка, возможны разные варианты подключения по VPN. Но меня интересует один вариант, определяемый моим провайдером, который я и выбираю. После нажатия на клавишу **Next** попадаешь в продолжение диалога, где и вводишь обычные данные об имени пользователя, пароле (как и в Windows) и т.д. Единственное отличие в добавлении двух строк, первая из которых касается IP адреса VPN-сервера (на рисунке ниже это 1.1.1.1) и IP адреса шлюза (на рисунке это 2.2.2.2), а вторая выглядит одинаково, я думаю, для всех пользователей.

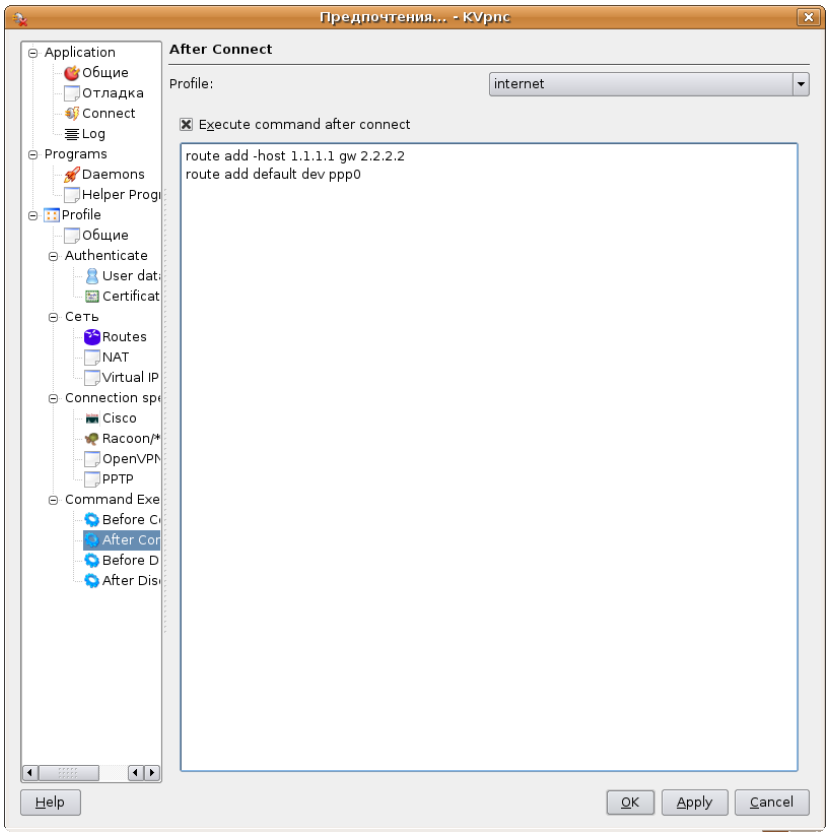


Рис. 1.7. Добавление двух строк в раздел After Connection

Это добавление необходимо сделать после завершения работы мастера настроек, запустив программу Kvpnc, и войдя в подменю After Connection меню настроек Settings. Нажатием клавиши **Apply**, как обычно, мы подтверждаем сделанные изменения, нажимаем клавишу **ОК**, и теперь можем запустить соединение, найдя в разделе Приложения, в подразделе Интернет программу Kvpnc.

Запущенная программа запросит пароль администратора компьютера (или root), а затем выведет окно подключения.

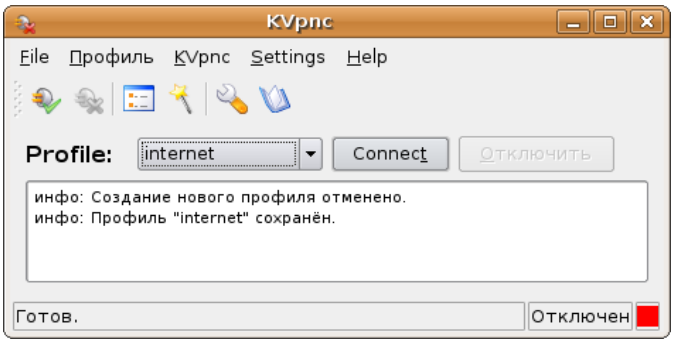


Рис. 1.8. Запуск программы kvpnc после настройки соединения

Осталось нажать клавишу **Connect**. После чего программа выводит сообщение о состоянии подключения (в нижней части экрана).

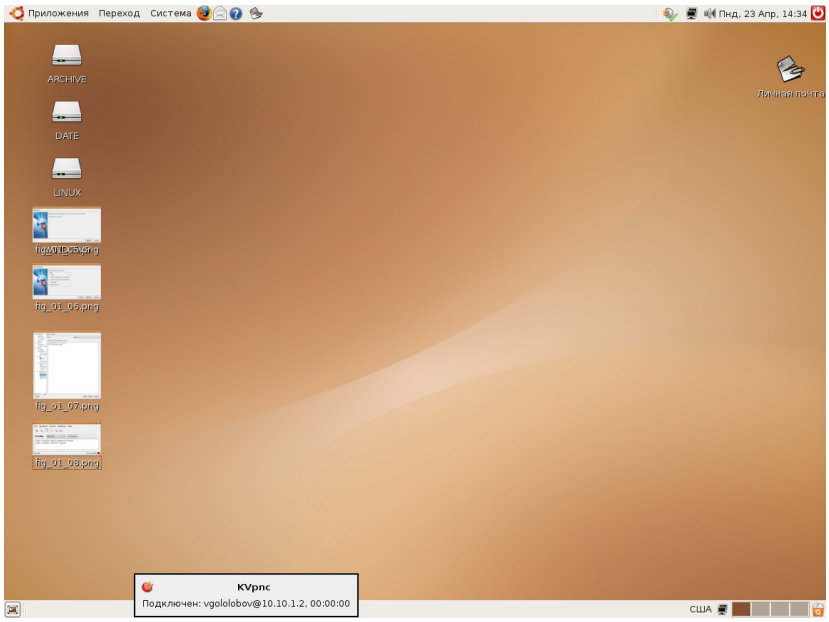


Рис. 1.9. Сообщение о подключении к Интернету в Ubuntu

Как и при настройке vpn-подключения, в Ubuntu я встретил несколько мелких проблем. Например, выбрав русскую раскладку клавиатуры при установке Ubuntu, я получил раскладку, в которой знаки препинания: точка, запятая, двоеточие и т.д., – расположены не вполне удобно для меня. Да, достаточно в файле xorg.config найти строку раскладки и исправить us, ru на us, ru(winkeys), а затем выбрать настройки X в выводимом сообщении после перезапуска X-сервера, то есть, завершения сеанса и начала нового, чтобы проблема была устранена, но приходится вспоминать, как разобраться с этой особенностью каждый раз после установки ОС. Или другая нелепость, появившаяся недавно. Попытка запустить привычный мне почтовый клиент Evolution выводит сообщение о фатальной ошибке, да и то при запуске из терминала, обычный запуск «умирает молча». Проблема решается почему-то весьма своеобразным образом: необходимо изменить часовой пояс в настройках даты-времени, задав, как рекомендовали на форуме, часовой пояс Киева. Запустить Evolution, настроить почту, после чего можно вернуть нормальный часовой пояс. Такие особенности настроек, я думаю, доставляют большое удовольствие любителям «поковыряться» в ОС, но очень мешают остальным пользователям переходить на Linux. Я пользуюсь vpn-сервером, где

для ускорения работы отключается MPPE. Это заставляет меня в файле options.pptp (/etc/ppp) удалять (или закомментировать) строку require-mppe-128. Это моя частная проблема, но есть и еще одна, которая может оказаться тоже частной, не знаю. В том виде, который настройки приобрели после создания vpn-подключения, я могу зайти на многие сайты, но не могу зайти на www.yandex.ru. У меня там почта, у меня там мой сайт, а зайти я не могу. Обидно.

В чем же проблема? В том, что по умолчанию используется MRU (не MTU, заметьте) со значением 1000. Графический интерфейс позволяет задать пользовательское значение MTU, но не MRU. После настроек в папке /etc/ppp/peers появляется файл сохранения настроек Kvpnc, где это отображено. Но попытки ввести исправление в этот файл не дают окончательного и бесповоротного решения. Во всяком случае на моем компьютере. Дело в том, что запуск Kvpnc проходит с вводом пароля root, при этом файл конфигурации переписывается. Чтобы он переписывался с нужным значением MRU необходимо в домашней папке root найти файл в скрытой папке .kde с именем kvpncrc, где и изменить значение MRU (для отображения в файловом менеджере скрытых файлов необходимо установить опцию «Показывать скрытые файлы» в разделе «Вид» основного меню).

Если не придавать значения двум строкам маршрутизации подключения после его установления, и тех правок в файлах, которые мне приходится делать, то формально остальные процессы не сложнее, чем в Windows. Да и вид экрана в операционной системе Linux мало отличен от Windows. А если кого-то это смущает, он может выбрать графический интерфейс KDE, или даже другой дистрибутив, который еще больше приближен по внешнему виду к Windows. Мне как пользователю, гораздо удобнее выполнять, скажем прямо, совсем мне не нужную процедуру создания подключения к Интернету (я предпочел бы иметь выход в Интернет сразу после физического подключения к сети) в графическом диалоге, а не писать коды в нескольких файлах. Тем обиднее, что все-таки приходится записывать что-то для маршрутизации, обидно, что приходится прописывать IP адрес vpn-сервера, что приходится искать, где изменить MRU. Все это наводит на грустные размышления, как то: был бы я столь же доволен использованием Linux, если бы несколько лет назад не получал удовольствия от возни с настройками, поиска решения проблем на Linux-форумах, если бы не привык к некоторым особенностям любого дистрибутива Linux...

Но я совсем отвлекся от рассказа. Красоты последних дистрибутивов, досадные мелочи, всего много, так и не доберешься до дела.

Запускаем Gambas, который находится в разделе Программирование основного меню Приложения. К слову, я года два использую графическую оболочку Gnome, хотя изредка захожу в KDE, прежде основную рабочую графическую оболочку. Gnome установился по умолчанию, я решил посмотреть, насколько сильно он изменился, да так и остался в нем. Но я опять отвлекся:

Рис. 1.10. Первый запуск среды программирования Gambas

Даже первый запуск, кроме традиционных и необходимых действий, таких как: создать новый проект, открыть существующий, использовать недавно использовавшийся проект, - имеет раздел примеров (Examples). И примеров очень много. Я обязательно хочу ими воспользоваться, когда начну работу проектами. А вот первый проект, все-таки не могу я все делать «по правилам», первый проект я возьму из книги М. Мозгового «Занимательное программирование», да простит меня автор, не только потому, что пример сам по себе интересен, но и потому, что автор приводит его в среде программирования Delphi. Я хочу повторить этот пример в качестве первого своего Gambas проекта, что называется «в лоб». В этом я доверюсь автору книги «Занимательное программирование», который не соотносит приведенные им программы с каким-то конкретным языком, он использует Pascal, язык системы программирования Delphi, как знакомый многим по студенческим временам. Мне же крайне любопытно, насколько возможно, не умея программировать, повторить программу, написанную на другом языке программирования. Если ничего не получится, брошу все. Или не брошу.

Я еще подумаю.

После выбора New project... (создание нового проекта) открывается диалог, в котором можно ввести название проекта, его заголовок, указать место, где проект будет располагаться. Gambas хранит все, что относится к проекту, в заданной директории, которая будет носить имя проекта. Для перемещения по диалогу достаточно нажимать клавишу **Next**, но в самом начале следует выбрать, какого рода проект вы намерены создать. На выбор три варианта – графический проект, терминальный проект, использование копии существующего (вашего или нет) проекта:

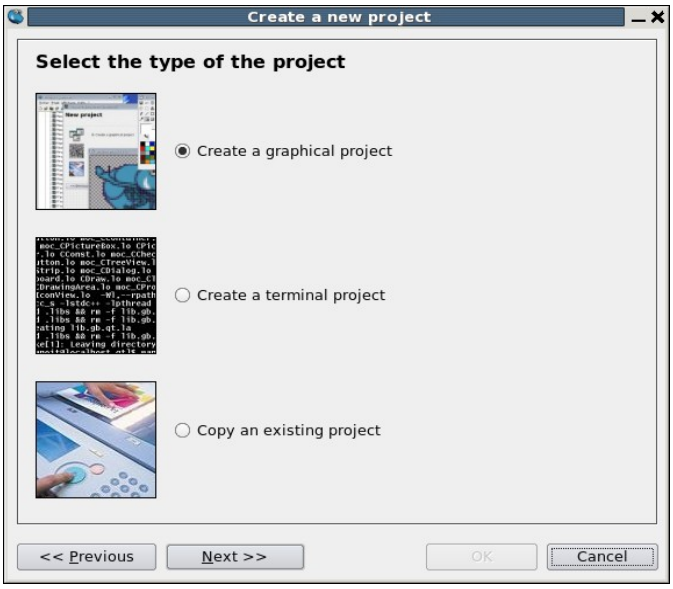


Рис. 1.11. Диалоговое окно выбора типа проекта

Меня интересует графический проект, и я оставляю флажок выбора там, где он есть: «Create a graphical project» – знай наших! Многие расчетные задачи, к чему и должен быть применен компьютер по его замыслу, прекрасно можно выполнить в терминале. При этом вы получите программу, которая будет весьма неприхотлива к возможностям компьютера, будет работать, как я себе представляю, без загрузки графической оболочки, а для удобства использования, если вы, как и я, не очень любите набивать множество команд, можно использовать MC – программу (или графический менеджер), которая называется «Midnight Commander», чтобы его не путали с «Norton Commander», которого в Linux нет. Вообще я давно «подсел» на графические возможности современных операционных систем, и сегодня, каждодневно обновляя систему, не столько озабочен ее стабильностью, давно уже не замечал каких-либо проблем с этим, не столько радуюсь обновлениям, связанным с вопросами защиты системы, вирусы в Linux не очень свирепствуют, сколько жду перемен в ее внешнем облике. Хотя понимаю, что самое разумное, это использовать всю мощь операционной системы без «графических» тормозов. Но...

Итак. Выбрав графический проект, дав ему имя и место жительства, мы попадаем в среду программирования, которая позволит сделать первый шаг – создать форму. Для этого в правом окне менеджера проекта, где отображается «дерево» проекта, я щелкаю правой клавишей мышки по разделу Forms, а в выпадающем меню выбираю раздел New, что приводит меня к меню выбора, в котором есть пункт Form. Этот процесс я понимаю так – создать новую форму. Форма же, насколько мне известно, а я уже немного «законспектировал» пояснения к языку Gamabs, особенно для моего графического выбора, является основным окном программы, которое появится, когда я заполню форму некоторым содержанием, и построю сам проект. В окне диалога создания формы можно задать имя формы, что важно для средних и больших проектов, содержащих много окон, можно установить ряд опций, назначения которых я пока не понимаю, поэтому не хочу их устанавливать:

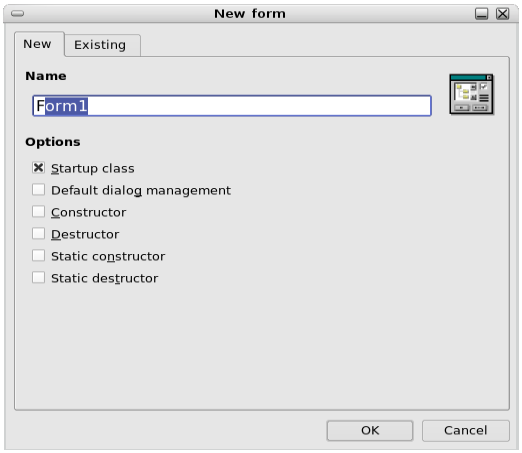


Рис. 1.12. Диалоговое окно создания формы в Gambas

Я не готов создавать средние и большие проекты, поэтому оставлю название формы без изменения и поскорее щелкну по клавише **OK**. В результате появляется форма и редактор программы. На форме, видимо, можно будет разместить разные кнопочки, окошки и другие средства управления программой. А в редакторе поместится все то, что и есть код программы. Следуя совету, прочитанному в книге «Занимательное программирование», я добавлю на форму область рисования (DrawingArea) и клавишу управления (Button). И то, и другое я выбираю на инструментальной панели справа, щелкнув по ним левой клавишей мышки, а затем с помощью мышки «нарисовав» контур окна и клавиши. На рисунке ниже кнопка управления на инструментальной панели изображена с надписью «OK», а область рисования имеет внутри пиктограмму, изображающую цветные геометрические фигуры: квадрат, круг, треугольник.

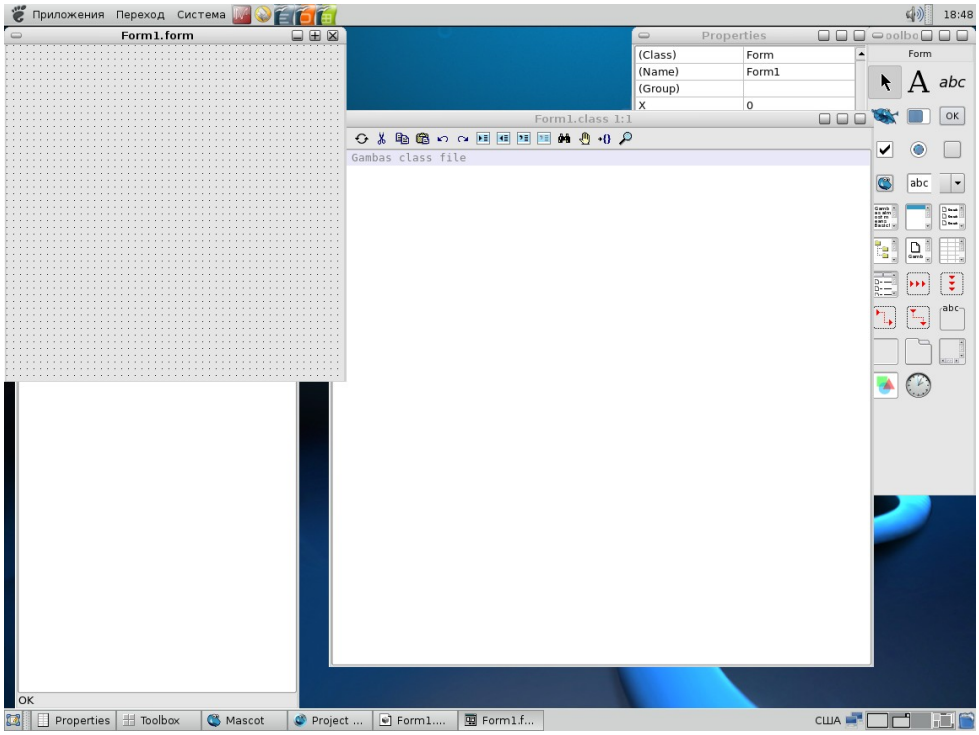


Рис. 1.13. Основная форма проекта и редактор кода

Форма, как и многие элементы управления, имеет свойство Text. Я полагаю, что это текст,

который появится вместо надписи Form1.Form при выполнении программы. Этот текст я заменю своим, и, если программа заработает, то увижу его.

После нескольких входов в Gamabas и выходов из него, я сталкиваюсь с маленькой, но для меня очень неприятной особенностью – я не знаю, как вызвать редактор кода. Мне же понадобится куда-то вписывать код. Я знаю, уже знаю, что двойной щелчок по кнопке управления открывает редактор кода и создает первую и последнюю строку текста события, связанного с щелчком по клавише при работе программы. Двойной щелчок по форме тоже вызывает аналогичное действие. Но это действие мне не нужно, я удаляю ненужный код, а сам прием использую для вызова редактора кода. Но не делать же это всякий раз, когда тебе нужно что-то исправить в программе. Меня это немного раздражает. Но только до тех пор, пока я не догадываюсь щелкнуть по форме правой клавишей мышки. В выпадающем меню есть пункт Code, который вызывает редактор кода и умиротворение в моей душе.

Программа, мой первый «блин комом», выглядит (я скопирую ее из книги «Занимательное программирование», чтобы не наделать ошибок, которые очень люблю делать при переписывании) следующим образом:

Листинг 1.1. Молекула газа в закрытом сосуде (первая версия)[1]

```
procedure TForm1.StartStopBtnClick(Sender: TObject);
    var X, Y   : Integer; { координаты молекулы }
        Vx, Vy : Integer; { составляющие скорости молекулы }
        angle  : Real;    { угол, задающий начальное направление полета }

    const R : Integer = 10; { радиус молекулы }
          V : Integer = 7;  { скорость молекулы }
begin
    Randomize;
    Screen.Refresh;                          { очистка экрана }
    X := RandomRange(R, Screen.Width - R);   { выбор начального (1)[2] }
    Y := RandomRange(R, Screen.Height - R);  { положения молекулы }
    angle := Random(360)* Pi / 180;          { и ее направления (2) }

    Vx := Round(V * Sin(angle));             { получение составляющих }
    Vy := Round(V * Cos(angle));             { скорости молекулы }

    while true do                            { основной цикл }
    begin
        Screen.Canvas.Pen.Color := clBtnFace;    { стираем молекулу (3) }
        Screen.Canvas.Ellipse(X - R, Y - R, X + R, Y + R);

        X := X + Vx;                        { сдвигаем на новую позицию }
        Y := Y + Vy;
                                            { определяем, не вышла ли }
        if X > Screen.Width - R then        { молекула за границы аквариума }
            begin  X := Screen.Width - R; Vx := -Vx; end;   { правая граница }
        if X < R then
            begin  X := R; Vx := -Vx; end;                  { левая граница }
        if Y > Screen.Height - R then
            begin  Y := Screen.Height - R; Vy := -Vy; end;  { нижняя граница }
        if Y < R then
            begin  Y := R; Vy := -Vy; end;                  { верхняя граница }

        Screen.Canvas.Pen.Color := clBlue;              { рисуем молекулу на }
        Screen.Canvas.Ellipse(X - R, Y - R, X + R, Y + R);   { новой позиции }

        Sleep(10);                                    { пауза 10 миллисекунд }
    end;
end;
```

Рис. 1.14. Листинг программы из книги М. Мозгового «Занимательное программирование»

Очень советую купить книгу, а я своими словами опишу программу. Представьте себе двумерный закрытый сосуд прямоугольной формы, в котором мечется, не находя себе места, шальная молекула газа. И молекула, и сосуд – все идеально. С точки зрения программы это означает, что молекула при своем движении сохраняет величину скорости, а ее направление меняется только при столкновении со стенками сосуда. Не думаю, что без посторонней помощи я смог бы написать такую программу. Я и не уверен, что без посторонней помощи смогу воспроизвести ее. Но попробую.

Первым делом я нарисую, как и собирался, область рисования и кнопку запуска программы на основной (в моем случае единственной) форме. Чтобы изменить название клавиши, после того как она появилась на форме под именем Button1, я выделяю эту клавишу щелчком по ней левой клавиши мышки. Клавиша приобретает вид контура с несколькими квадратиками, с помощью «таскания» за которые можно менять размеры клавиши. А окно свойств Properties открывается на свойствах клавиши, что можно сразу увидеть по первой строке (Class) – Button. В этом окне свойств есть пункт под названием Text в левой колонке. Если правее, в следующей колонке, ввести «Старт», то клавиша изменит свое название, видимое на форме. Аналогично я поступлю с областью рисования, которую назову Screen, как в оригинале, но это будет уже не надпись, как на клавише, а, собственно, имя. И меняю я его напротив (Name) в окне свойств после выделения области рисования щелчком левой клавиши мышки по ней.

Теперь можно начать писать код программы (я прочитал в отличие от вас все, что автор говорит о программе). Поскольку программа должна запускаться после нажатия на клавишу «Старт», а в руководстве к Gamabs я нашел, что двойной щелчок по элементам управления вписывает код, который создает заголовок для кода программы, отвечающего на событие – щелчок мышки по клавише управления, я дважды щелкаю по клавише и получаю в редакторе кода первую запись (увы, пока не мою) кода программы. В листинге на рисунке 1.9 аналогичная запись начинается словом **procedure,** а ниже начинается описание переменных и констант. Не буду фантазировать, открою с разделе помощи пункт означенный «Объявление локальных переменных» и объявлю их, а заодно и константы:

```
PUBLIC CONST R AS Integer = 10
PUBLIC CONST V AS Integer = 7

PUBLIC SUB Button1_Click()
      DIM X AS Integer
      DIM Y AS Integer
      DIM Vx AS Integer
      DIM Vy AS Integer
      DIM angle AS Float
END
```

А вот и не заодно. Пришлось константы вынести из блока программы, обслуживающего щелчок по клавише «Старт», иначе возникает ошибка при компиляции программы. Осерчав на себя, из-за того, что не понимаю, почему так, я обозначил их PUBLIC, то есть, доступные всем другим классам (так написано в help). Возможно со временем я и пойму, почему и отчего, но пока мне приходит в голову только одно соображение – константа вещь постоянная и ее следует рассматривать глобально, но это мои фантазии. Лучше продолжим. В листинге первая операция в программе Randomize. Попробуем отыскать аналогичную в описании языка Gamabas:

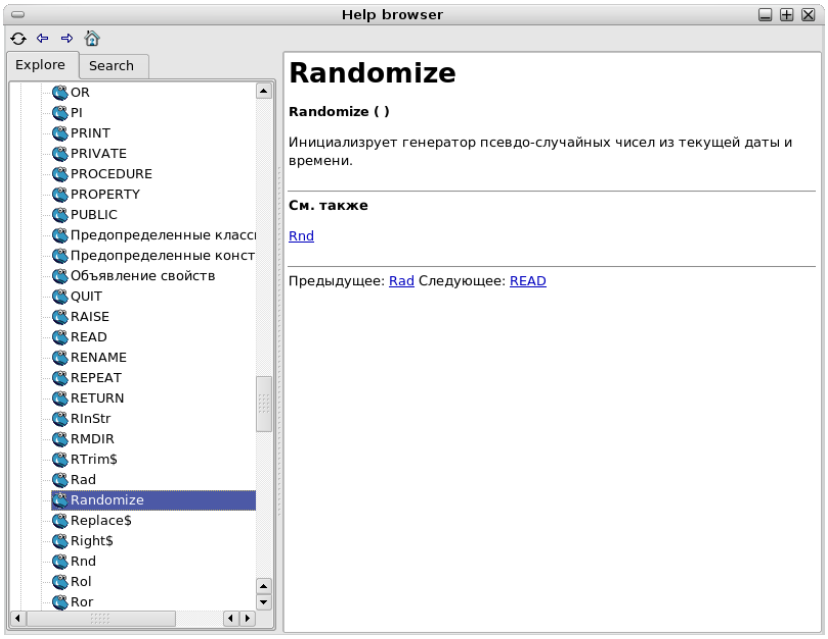


Рис. 1.15. Подсказка по языку Gambas

Следующий оператор в листинге, если я не ошибаюсь, обновляет область рисования. Для получения этого я воспользуюсь свойством редактора кода – после точки, которую можно поставить после имени объекта, открывается окно свойств и методов объекта. Так построены многие среды разработки программ, так построена и среда программирования Gambas. Конечно не слишком трудно вписать операторы, методы или свойства, это так, но мне не хочется начинать с опечаток и ошибок в написании текста кода. Поэтому я воспользуюсь возможностями, предоставляемыми хорошо сделанной системой:

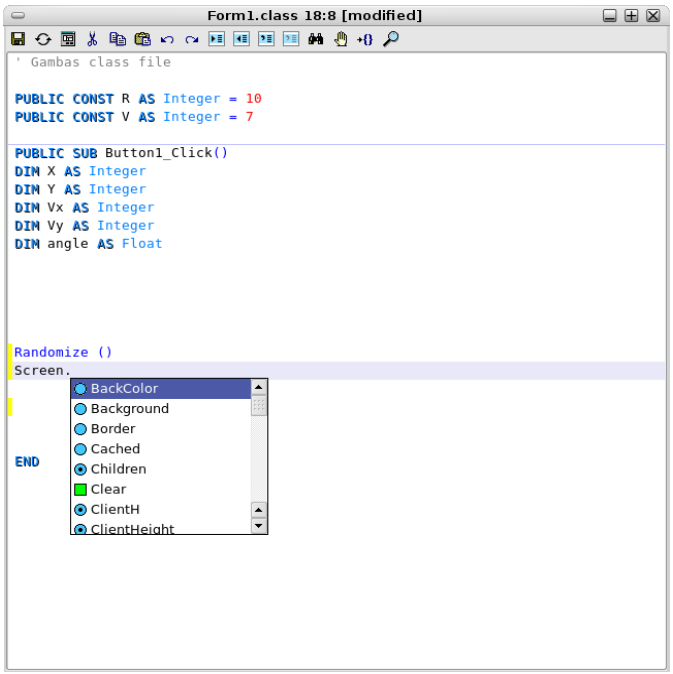


Рис. 1.16. Меню методов и свойств объекта

В открывающемся окне я постараюсь найти что-то похожее. И нахожу. Затем, используя

оператор Rnd, я повторю записи, сделанные в листинге. Теперь программа выглядит так:

```
' Gambas class file

PUBLIC CONST R AS Integer = 10
PUBLIC CONST V AS Integer = 7

PUBLIC SUB Button1_Click()
      DIM X AS Integer
      DIM Y AS Integer
      DIM Vx AS Integer
      DIM Vy AS Integer
      DIM angle AS Float

      Randomize ()
      Screen.Refresh
      X = Rnd (R, Screen.Width - R)
      Y = Rnd (R, Screen.Height - R)
      angle = Rnd (0, 360)*Pi/180
      Vx = Round (V*Sin(angle))
      Vy = Round (V*Cos(angle))
END
```

Теперь осталось вставить цикл. В коде цикла есть несколько записей, которые я не знаю, как выполнить, и пока оставлю программу без них, попробовав ее запустить:

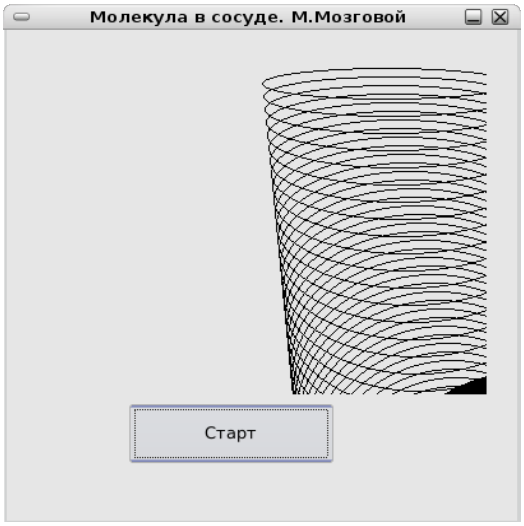


Рис. 1.17. Первый запуск незавершенной программы

Это не то, конечно, что должно получиться. Первое, что я пока не знаю как сделать, это – стереть молекулу. Или, если быть точным, не знаю как ее нарисовать цветом фона, как это сделано в оригинальном листинге, а позже перерисовать в новой позиции синим цветом. И я не знаю, как сделать паузу, поскольку не нахожу подходящего оператора. Кроме того, я ожидал, что молекула будет меньше, и у меня она, похоже, не круглая. Но с этим, я надеюсь, разобраться будет проще. Поищем ответы на основные вопросы. Один ответ нашелся быстро: пауза в 10 миллисекунд получается с оператором WAIT 0.01, где число – необходимая длительность паузы. Теперь рисунок образуется не сразу, как прежде, а постепенно. Устранив ошибку, которой я обязан своей невнимательности, и из-за которой не происходило отражения от правой и нижней границ области рисования (в одном месте вместо Y я написал X), и заменив оба параметра размера эллипса радиусом, отчего «молекула» перестала менять

размер при движении, я получаю результат близкий к ожидаемому. Но я все еще не умею стирать молекулу. Без этого результат работы программы выглядит примерно так:

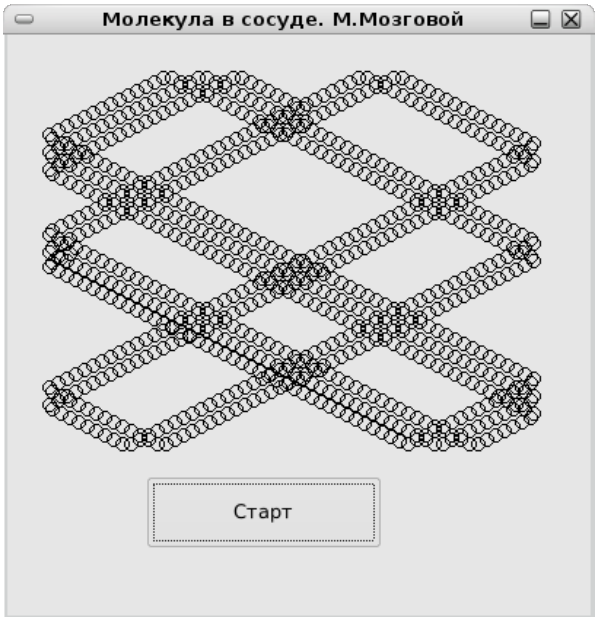


Рис. 1.18. Запуск программы после устранения ошибок

И, наконец, с помощью операции присваивания объекту рисования цвета формы

```
Draw.ForeColor = Form.Background
```

мне удается «стереть» молекулу в начале цикла, а с присвоением синего цвета ближе к концу цикла

```
Draw.ForeColor = 255
```

получить то, что я представляю, как программу из книги «Занимательное программирование». Выглядит она в Gambas так:

```
' Gambas class file

PUBLIC CONST R AS Integer = 10
PUBLIC CONST V AS Integer = 7

PUBLIC SUB Button1_Click()
      DIM X AS Integer
      DIM Y AS Integer
      DIM Vx AS Integer
      DIM Vy AS Integer
      DIM angle AS Float

      Randomize ()
      Screen.Refresh
      X = Rnd (R, Screen.Width - R)
      Y = Rnd (R, Screen.Height - R)
      angle = Rnd (0, 360)*Pi/180
      Vx = Round (V*Sin(angle))
      Vy = Round (V*Cos(angle))
```

```
      WHILE TRUE
            Draw.Begin (Screen)
            Draw.ForeColor = Form.Background
            Draw.Ellipse (X-R, Y-R, R, R)
            X = X + Vx
            Y = Y + Vy
            IF X > Screen.Width - R THEN
                  X = Screen.Width - R
                  Vx = -Vx
            ENDIF

            IF X < R THEN
                  X = R
                  Vx = -Vx
            ENDIF

            IF Y > Screen.Height - R THEN
                  Y = Screen.Height - R
                  Vy = -Vy
            ENDIF

            IF Y<R THEN
                  Y = R
                  Vy = -Vy
            ENDIF

            Draw.ForeColor = 255
            Draw.Ellipse (X-R, Y-R, R, R)
            Draw.End
            WAIT 0.01
      WEND
END
```

И работает это:

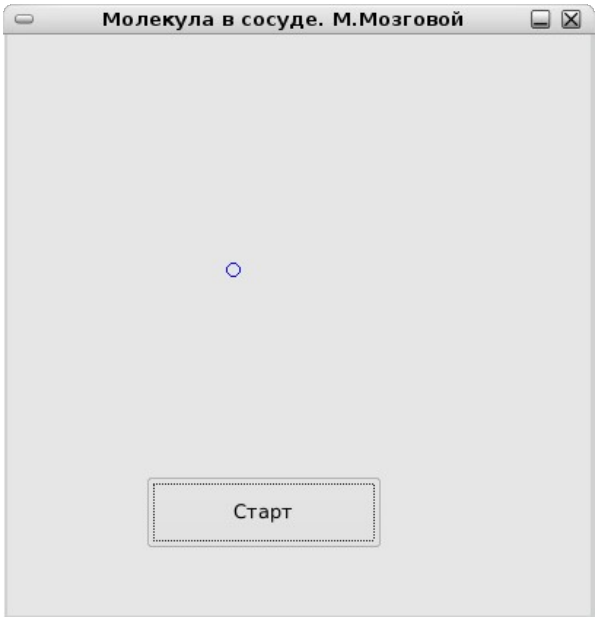


Рис. 1.19. Окончательный результат работы программы

Если отбросить некоторые языковые различия, такие, например, как операторные скобки

рисования:

```
Draw.Begin (Screen)
Draw.End
```

и различия в написании, что и должно было иметь место, то в остальном код программы действительно переписан «в лоб». Следовательно, предположение, что можно на протяжении книги использовать другую среду программирования, достаточно хорошо подтверждается. А сама среда программирования, по крайней мере для небольших проектов, работает не хуже, чем Delphi. Так что, надеюсь, к концу книги, хотя программистом я и не стану, но немного программировать научусь.

А сейчас я вспомнил, что хотел посмотреть раздел примеров в Gambas. Вот и посмотрю. Но прежде хочу добавить в программу, то что предлагает ее автор в самом начале разговора о кодировании программы – добавить возможность останавливать программу не принудительно с помощью средств разработки, нажимая на красную клавишу «Стоп» основного меню, а использовать ту же кнопку управления, что уже есть на форме. В этом случае начало программы в Gambas выглядит так:

```
 ' Gambas class file

PUBLIC CONST R AS Integer = 10
PUBLIC CONST V AS Integer = 7
PUBLIC Running AS Boolean

PUBLIC SUB Button1_Click()
      DIM X AS Integer
      DIM Y AS Integer
      DIM Vx AS Integer
      DIM Vy AS Integer
      DIM angle AS Float

      IF Running THEN
            Running = FALSE
            QUIT
      ENDIF

      Button1.Text = "СТОП"
      Running = TRUE
      ........................
```

Далее программа остается без изменения. Так, действительно, удобнее. А примеры, которые есть в Gambas заставляют меня тихо завидовать тем, кто умеет так программировать.

Рис. 1.20. Пример работы программы из раздела Examples системы Gambas

Часы не только ходят после запуска программы, но и показывают правильное время. Что ж, будем учиться. Стоп.

Компьютер может помочь с программированием

Во всяком случае, так говорят знающие люди. И они же говорят, что программирование – это не написание самой программы, то есть, кода программы на каком-либо языке программирования. Программа – это в первую очередь описание того, что должно происходить с того момента, как программа запущена. Для этого предварительного описания можно использовать, как мне кажется, обычный язык. В одной из книг, которые я сейчас просматриваю, пытаясь научиться программировать, говорится о псевдо-языке программирования, с помощью которого можно описать программу. А на моем компьютере есть программа Umbrello, назначение которой мне давно хотелось понять. У меня эта программа в разделе «Программирование» основного меню «Приложения». Запустить программу не составляет труда – это я уже освоил. И руководство в разделе «Справка» основного меню, то, что сейчас надо.

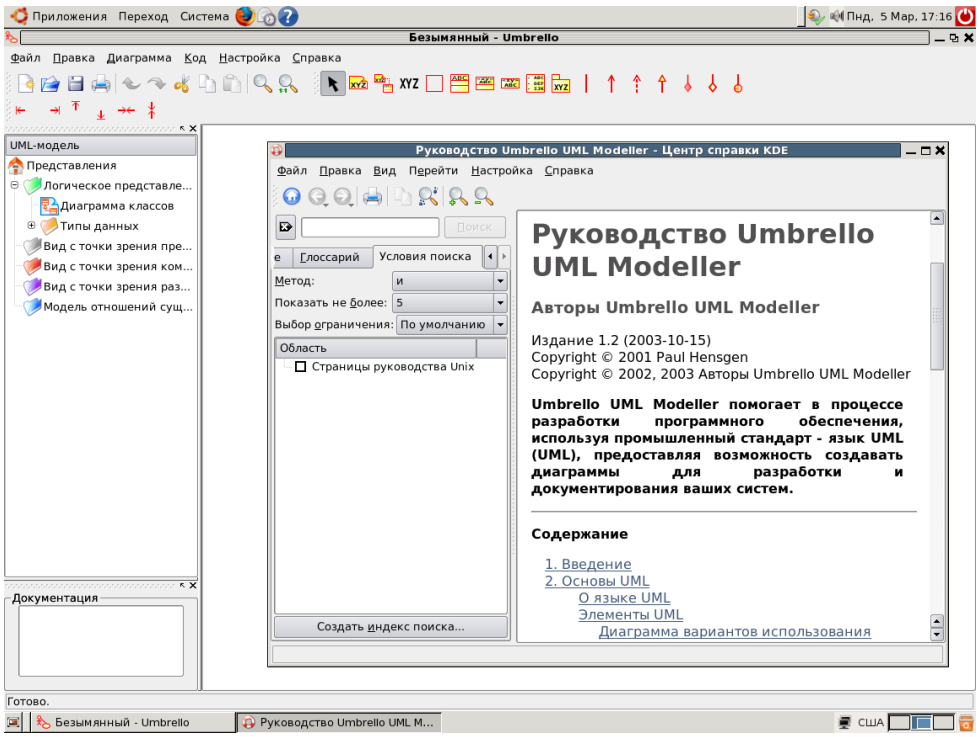


Рис. 1.21. Программа Umbrello

На рисунке можно разглядеть, что Umbrello помогает в процессе разработки программного обеспечения. Это-то мне и нужно. Ничто так ни важно для начинающего, как помощь на первом этапе освоения нового.

Руководство к программе русифицировано, как и сама программа, и против своего обыкновения, надеюсь вы лишены этого недостатка, прежде, чем начать пользоваться программой, я попробую прочитать руководство. Отдельные моменты, чтобы вам не искать, процитирую. Например, вот это:

Umbrello UML Modeller — это средство для работы с диаграммами UML, которое

может быть полезным в процессе разработки программного обеспечения, особенно на стадиях анализа и проектирования...
Хорошая модель очень важна для средних и больших проектов, но также будет полезна и для небольших проектов. Даже если вы один работаете над маленьким проектом, хорошая модель предоставит вам общую картину разработки, что поможет вам сразу написать качественный код.
UML является языком диаграмм, используемым для описания таких моделей. Вы можете выражать свои идеи в UML с помощью различных типов диаграмм...
Унифицированный язык моделирования (UML) является языком диаграмм или обозначений для спецификации, визуализации и документации модели объектно-ориентированных программных систем. UML не является методом разработки, то есть он не определяет последовательность действий при разработке программного обеспечения. Он помогает описать свою идею и взаимодействовать с другими разработчиками системы.

У меня проект маленький. И других разработчиков не предвидится, но это и не обязательно. Первое, что мне хочется сделать, иначе трудно будет разбираться с программой позже, это привести руководство в соответствие с имеющейся у меня версией. Руководство, видимо, написано для предыдущей версии, и, хотя принципиальных различий может не быть, детали, пока не освоился, тоже могут помешать пониманию. Для себя я запишу соответствие названий диаграмм в руководстве и моей версии:

Диаграмма класса - Диаграмма класса
Диаграмма последовательности - Диаграмма последовательности
Диаграмма взаимодействий - Диаграмма отношений сущностей
Диаграмма вариантов использования - Диаграмма прецедентов
Диаграмма состояний - Диаграмма состояний
Диаграмма действий - Диаграмма деятельности
Диаграмма компонентов - Диаграмма компонентов
Диаграмма выпуска - Диаграмма развертки

Есть еще в моей версии «Диаграмма кооперации», но куда ее приткнуть? Почитаем.
В руководстве часто упоминается слово «класс». Запишу-ка я, что это значит:

Класс определяет атрибуты и методы набора объектов. Все объекты класса (называемые экземплярами) имеют одинаковое поведение и одинаковый набор атрибутов (у каждого объекта - собственный набор атрибутов). Иногда вместо класса используется термин «Тип», но важно понимать, что эти термины неодинаковы. Термин «Тип» имеет более общий смысл.

Примеры использования диаграмм в руководстве есть. Пора и нарисовать что-нибудь. Но я на свет, конечно, не вчера родился. Знаю, что первый шаг в любом деле, определение цели. Это потом можно подумать, как достичь намеченного. А моя цель сейчас – разобраться со схемой, которая не желает работать. Сразу разобраться со всей схемой я не осилю, поэтому постараюсь сделать из проблемы маленькую, но чем-то похожую, задачку.

Пусть будет такая маленькая задачка: есть цифровой счетчик с предустановкой и сменой направления счета, как его проверить? Я пытался посмотреть сигналы осциллографом, но ничего не получилось. Не пойму, работает счетчик, нет; может микросхема плохая попалась, или я ее успел «сжечь». Изобразим счетчик.

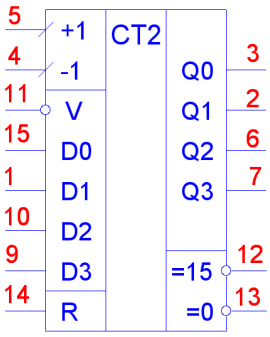


Рис. 1.22. Двоичный реверсивный счетчик с предустановкой

Специалисту ясно, что данные для записи следует подать на входы D0-D3, а перепись данных осуществится при низком уровне на входе V, для прямого отсчета служит вход +1, фронт от низкого состояния к высокому, и т.д. Вход, который не используется при подключении к тактовому генератору, это я прочитал в книге В.Л. Шило «Популярные цифровые микросхемы», много раз она меня выручала, неиспользуемый вход направления счета следует установить в высокое состояние.

Собственно, я пытался проверить это на макетной плате, но с одним генератором и осциллографом... Вот и хочу я, чтобы компьютер заменил мне и все нужные генераторы, и числа для предустановки задавал, и... и потом еще придумаю, чтобы он такое делал.

Безусловно, я понимаю, что сам компьютер не догадается помочь мне, пока не попросишь его с помощью программы. Придется попросить. Для начала нарисую, как я буду работать с программой. Это только первые соображения.

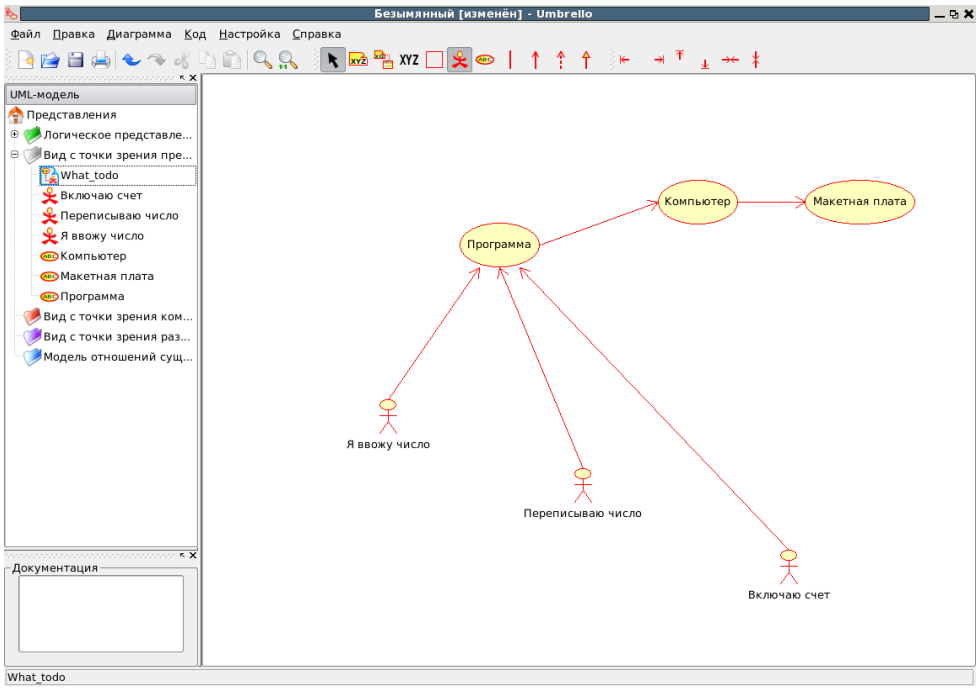


Рис. 1.23. Первые соображения о том, что следует делать

Меня можно спросить, почему «меня» так много на рисунке? Отвечу, так сложилось исторически. Меня можно спросить, а как компьютер будет «общаться» с макетной платой? Отвечу – это провокация, на каверзные вопросы не отвечаю!

Очень часто устройства, добавляемые к компьютеру, подключают к его шине внешних устройств, но я не намерен разбирать компьютер. Очень часто внешние устройства подключают к порту принтера. Такое подключение обеспечивает достаточно быстрый обмен информацией между компьютером и подключенным устройством, дает возможность, переделав программу, привести характер работы порта в соответствие с текущей необходимостью, однако я предпочитаю поступить иначе. Мне нет необходимости в быстром обмене информацией, скорее, наоборот. Использование порта принтера потребует его защиты от случайного повреждения, то есть, необходимости что-то спаять. Не то, чтобы паять я не умел, но если паять, то нечто нужное неоднократно.

В предыдущей серии экспериментов я использовал последовательный порт компьютера, и у меня остался интерфейс RS485 и макетная плата с панелькой для микроконтроллера. Пусть микроконтроллер обслуживает проверяемую схему, а с компьютера будут отправляться команды микроконтроллеру.

Есть несколько соображений в пользу такого решения. Первое из них – мне хочется рассказать о программе Piklab больше, чем я это сделал в прошлый раз. Программа работает с PIC-микроконтроллерами в Linux, как MPLAB в среде Windows. И если о работе MPLAB я уже рассказывал, то о Piklab, скорее, упоминал.

Кроме того, задумав сделать из компьютера нечто в роде испытательного стенда, я подумал о том, что не всегда у меня будет желание переносить его к рабочему столоу, на котором у меня разложены макетная плата, лабораторный блок питания и прочие аксессуары «железячного» бытия. Бросить же на время экспериментов от компьютера к макету тонкий и легкий четырех-жильный телефонный провод можно просто по полу через все помещение.

Но самым существенным аргументом для меня в подобном решении будет один – если мне понадобится посмотреть осциллографом сигналы, то мне понадобятся короткие и «жесткие» относительно периодичесности последовательности импульсов. Микроконтроллер справится с этой задачей лучше компьютера. И в этом смысле, если я не ошибаюсь, а если ошибаюсь пусть опытные программисты меня поправят, я буду использовать программу, работающую на компьютере, в основном в качестве удобного графического интерфейса пользователя. А вся аппаратная часть задачи будет реализована на макетной плате, возглавляемой микроконтроллером. Но это пока мечты.

Если запускаешь программу, то обычно открывается окно, где сверху есть меню, а в окне можно писать, или рисовать, или что-то строить. Мне строить еще рано, а на первых порах нужно только одно маленькое окошко и одну-две кнопки. В окошко я собираюсь вписывать число для предустановки, одна кнопка мне нужна для установки этого числа на выводы микросхемы, вторая кнопка для переписи числа и начала счета.

После этого программа должна заставить компьютер осчастливить меня в качестве генератора всех необходимых мне сигналов. Сильно подозреваю, что если я положу макетную плату на компьютер, то она не найдет общего языка с ним. Придется что-то еще добавить. Скорее всего выглядеть моя задумка будет как-нибудь так:

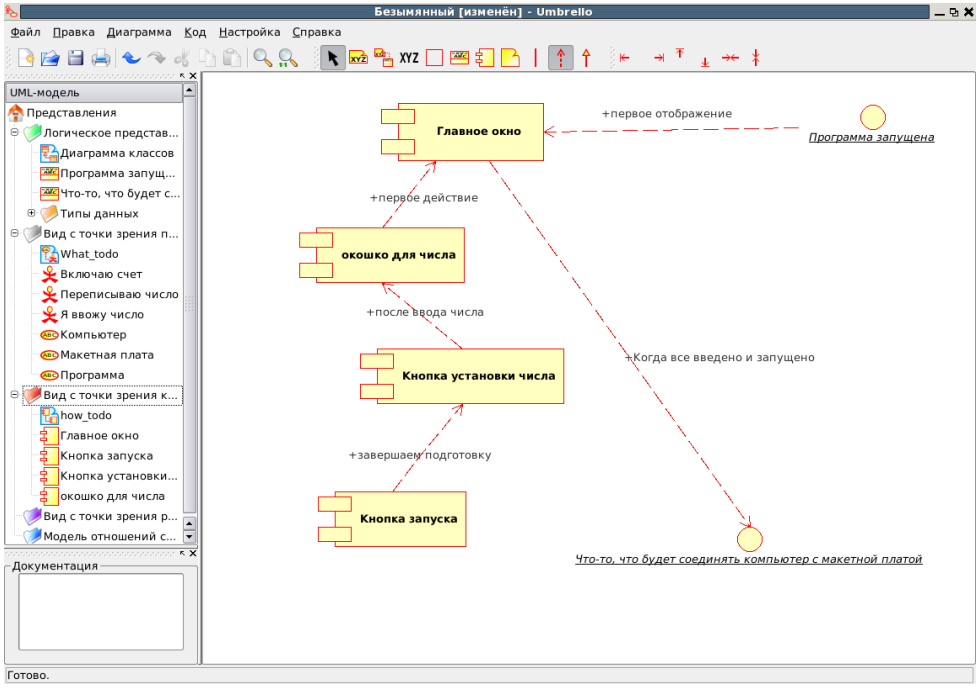


Рис. 1.24. Окончательные соображения перед началом программирования

Я люблю рисовать квадратики, кружочки, рожицы, когда пытаюсь сообразить, что же делать дальше. Я понимаю, что программа Umbrello не для того предназначена, но я еще вернусь к ней. А сейчас у меня план такой – познакомиться с таинственными сущностями, что окружают программиста.

Почему Емеля ездил на печи?

Думаю, и это только мое личное мнение, что тогда он еще не заработал на приличный «Мерседес».

Примерно так же я представляю себе соотношение процедурного и объектно-ориентированного языков программирования, и это тоже только мои предположения. В моих ветхозаветных воспоминаниях о программировании, если отбросить вопрос о типах данных, таких как целое, символ, число с плавающей точкой и т.д., не обращать внимание на переменные, без которых я и сегодня не представляю, может ли существовать программа, и операторах языка, если забыть о подпрограммах, функциях и процедурах, то программа сводится к набору последовательных инструкций, предназначенных для компьютера (или процессора). Эта последовательность может прерываться, когда нужно что-то проверить: выходя из дома, проверяешь, а взял ли ты ключи, надел ли башмаки? Если надел, то можно идти дальше, иначе нужно вернуться и сменить обувь. Иногда последовательность прерывается неким образом, похожим на тот случай, когда уходя из дома, задумываешься, погасил ли ты свет? Ты возвращаешься, проверяешь, уходишь, но по дороге тебе приходит в голову, а вспомнил ли ты о том, что надо проверить, погасил ли ты свет? Этот круговорот рассеянности заканчивается только тогда, когда все у тебя сходится: и проверил, и не забыл, и выключено все.

В таком программировании мне понятно назначение подпрограмм – если часто делаешь одно и то же, как, например, поездка на работу, нет смысла каждый раз расписывать это в последовательных инструкциях. А компьютеру нужно все объяснить до мелочей. Можно один раз эти мелочи расписать, назвать этот набор инструкций каким-то образом, скажем,

«На работу», и впоследствии, как каждое утро, позавтракав, радостно восклицать: «На работу!», – вызывая одну и ту же рутинную последовательность действий.

Не могу сказать, что все было ясно и просто, если и прежде приходилось сталкиваться с программированием, одни только указатели в языке С, или передача параметров функции по значению или ссылке, вызывали у меня тихую внутреннюю дрожь. А запись программы, как стенография, заставляла открывать справочники и за каждым действием отправляться в долгое путешествие по этим полезным книгам. Но если код программы был мне не ясен, то сущность процесса, в какой-то мере, была ясна.

Теперь же, объектно-ориентированное программирование лишает меня даже этих скромных познаний: классы, их дети и родители, наследство и права на него...

Вместе с тем, без понимания того, что такое объект и класс, может статься, что и не получится у меня ничего с Gambas.

Одно из определений класса я приводил, взяв его из описания программы Umbrello, приведу второе, взятое из help'a Gambas:

Класс - разновидность шаблона (template), он не используется непосредственно. Для его использования создается его копия. Эта копия называется объект. Вы можете сделать разные объекты того же самого класса.

Класс - это тип данных, так что объявление объекта похоже на обычное объявление переменной:

object_name AS Class_name

но объект должен быть подтвержден с помощью NEW:

object_name = NEW Class_name

Запишу еще несколько определений, прежде чем начну попытки разобраться с этим. Следующее взято мною из книги С.Липпмана:

Двумя первичными характеристиками объектно-ориентированного программирования являются наследование и полиморфизм. ***Наследование*** *позволяет нам группировать классы в семейства связанных типов, давая возможность разделять общие операции и данные.* ***Полиморфизм*** *позволяет нам программировать эти семейства, как объединения несколько иные, чем индивидуальные классы, предоставляя нам больше гибкости в добавлении или удалении отдельных классов.*

Наследование определяет взаимосвязь предок/потомок. ***Предок*** *определяет общий интерфейс и частную реализацию, общие для всех его потомков. Каждый* ***потомок*** *добавляет или изменяет, что ему наследовать для реализации его собственного уникального выполнения.*

Взаимосвязь между родительским или базовым классом и его потомками называется ***иерархия наследования.***

В объектно-ориентированном программировании мы косвенно манипулируем объектами класса нашего приложения через указатель или ссылку на абстрактный базовый класс вместо прямого управления объектами реального производного класса нашего приложения. Это позволяет нам добавлять или удалять производный класс без необходимости каких-либо переделок нашей существующей программы.

Второй уникальный аспект объектно-ориентированного программирования - это ***полиморфизм****: возможность указателю на базовый класс или ссылке явно ссылаться на любой из объектов его производного класса.*

Если вам что-то стало понятно, то я вам завидую. У меня полнейшая каша в голове. А еще

есть контейнеры и много еще более загадочных сущностей! Попробую найти еще что-нибудь полезное в Интернете.

Объектно-ориентированное программирование - технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами.

Отыскалось много интересного и на сайтах университетов. Вот например:

Программы - это совокупность взаимодействующих объектов. Каждый объект отвечает за конкретную задачу. Вычисление осуществляется посредством взаимодействия объектов... Поведение объекта диктуется классом. Данные и поведение представлены в виде классов, экземпляры которых - объекты. Все экземпляры одного класса будут вести себя одинаковым образом в ответ на одинаковые запросы.

Объект проявляет свое поведение путем вызова метода в ответ на сообщение. Интерпретация сообщения зависит от объекта и может быть различной для различных классов объектов.

Для удобства создания нового типа из уже существующих типов, определенных пользователем используется механизм наследования. Классы могут быть организованы в виде иерархического дерева наследования.

Таким образом, Объектно-ориентированный язык должен обладать свойствами абстракции, инкапсуляции, наследования и полиморфизма.

***Инкапсуляция** с сокрытием данных - способность отличать внутреннее состояние объекта и поведение от его внешнего состояния и поведения.*

***Абстракция** - расширяемость типов - способность добавлять типы, определяемые пользователем для того, чтобы дополнить ими встроенные типы. Один из принципов ООП заключается в том, чтобы типы, определяемые пользователем, должны обладать теми же привилегиями, что и встроенные типы.*

***Наследование** - способность создавать новые типы, повторно используя, описание существующих типов.*

***Полиморфизм** с динамическим (поздним) связыванием - способность объектов быть ответственными за интерпретацию вызова функции.*

Принципы Объектно-ориентированного подхода:

- *Действие в объектно-ориентированном программировании инициируется посредством передачи сообщений объекту. Сообщение содержит запрос на осуществление действия. В качестве реакции на сообщение получатель запустит некоторый метод, чтобы удовлетворить принятый запрос.*
- *Все объекты являются экземплярами, классов. Все объекты одного класса используют одни и те же методы в ответ на одинаковые сообщения.*
- *Принцип наследования. Классы могут быть организованы в иерархическую структуру с наследованием свойств. Дочерний класс наследует атрибуты родительского класса.*
- *Принцип полиморфизма. Объекты реагируют на одно и тоже сообщение строго специфичным для них образом.*

С удовольствием, и без зависти, я прочитал все найденные мною статьи. И слова, и даже почти все термины в них понятны, но лишь пока читаешь. Стоит на мгновение отвлечься, как, возвращаясь к прерванному чтению, ты уже не так уверен в том, что тебе все понятно. В этом смысле, боюсь, мне более всего подойдет следующее:

Для тех, кто не имеет об этом ни малейшего понятия, мы не будем подробно объяснять, что такое объектно-ориентированное программирование. В этот вопрос и так уже внесено достаточно путаницы. Поэтому забудьте о том, что люди говорили вам об объектно-ориентированном программировании. Наилучший способ (и, фактически, единственный) изучить что-либо полезное об объектно-ориентированном программировании - это сделать то, что вы уже почти сделали: сесть и попытаться узнать все самостоятельно.

Последую этому совету. А чтобы не сожалеть о времени, потраченном на чтение книг, статей и методических указаний, на первом этапе я определю для себя, что объектно-ориентированное программирование отлично от традиционного в первую очередь тем, что его основным элементом является объект с набором свойств и возможных действий для общения с другими сущностями программы. Это не определение – это маленький островок в той каше-размазне, что у меня сейчас в голове.

По мере работы с языком Gambas я постараюсь возвращаться к найденным мной статьям, проверяя не рассеялся ли туман в голове? А для начала поробую перевести все то немногое, что удалось понять (или кажется, что удалось) на область, которая мне привычнее – электронику. Как и программирование, электроника развивалась, и если когда-то, как при процедурном программировании, схема строилась на отдельных элементах: это были резисторы, конденсаторы, транзисторы, – то позже появлись микросхемы, а с их появлением и другой подход к созданию схем. Схема, как и раньше, рассматривается в функциональном плане. Но если раньше эти функции реализовывались с помощью построения схемы из отдельных элементов, то сегодня для выполнения необходимых действий (операций с сигналами) выбирается подходящая микросхема. Она имеет определенные свойства, как питающее напряжение, цепи коррекции или задания режимов, уровни входных и выходных сигналов, и имеет возможность действовать, преобразуя входные сигналы в выходные, генерируя выходные сигналы при определенных условиях, за которыми микросхема следит и т.д.

Интересный, как мне кажется, пример видения программы в традиционном подходе и с «правильной» точки зрения я встретил в книге С.Н. Лукина «Понятно о Visual Basic.NET». Хотя программа относится к Visual Basic, ее без труда можно реализовать в среде Gambas, что я и хочу сделать. Речь в книге идет о создании программы собственного калькулятора. В моей операционной системе Linux есть штатный калькулятор, приводимый к виду научных расчетов, калькулятор, который позволяет записать достаточно сложное выражение и произвести расчет нажатием на клавишу «равно», и я часто им пользуюсь, но... Но в некоторых ситуациях даже он не спасает меня от противного даже самому ворчания пополам с четырханием. Однако вначале о программе. Запускаем Gambas. Кстати о Gambas. Почти готова вторая стабильная версия программы, и, после некоторого размышления о быстротечности всего сущего, я решил, что далее буду использовать ее. Это версия 1.9.48 (разрабатываемая). Итак, запускаем все-таки Gambas:

Рис. 1.25. Gambas 2, окно запуска

По всей видимости новая версия среды программирования претерпела существенные измения, и к моменту выхода стабильной версии эти изменения могут еще в большей мере коснуться программы, но это важно, с моей «колокольни», для профессионалов, а не для таких дилетантов, как я. По этой причине, имея первый опыт работы в интегрированной разработки проектов Gambas, я выбираю раздел **New project**, получая следующие возможности для разработки программы:

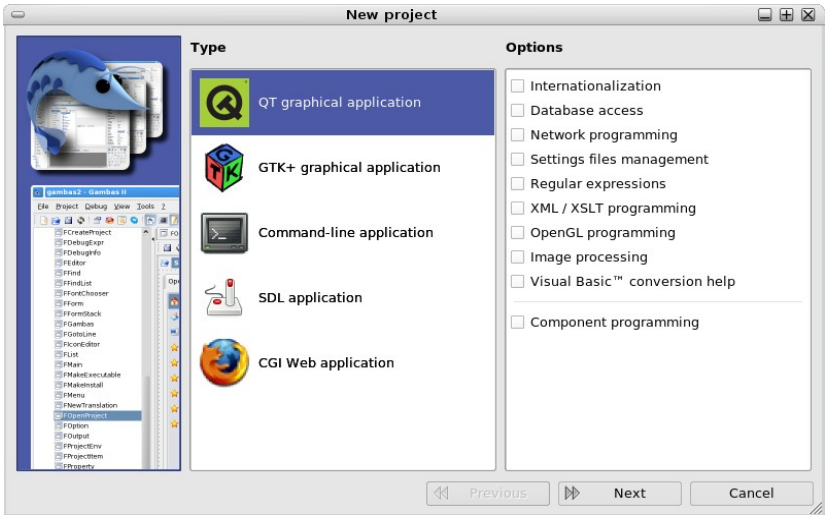


Рис. 1.26. Возможные виды проектов в Gambas 2

Как и ранее я выбираю для своих целей графический проект с применением средств Qt. Пока я не знаю назначение опций, находящихся в правом окне, и пока не использую их.

Проект я назову «Calc» или как-нибудь похоже, пока это неважно. Попадая в рабочее окно, я получаю основную форму, для вызова на экран которой дважды щелкаю в левом окне «дерева» проекта. Что мне нужно для демонстрации, так это три окошка **TextBox** и одна клавиша **Button**. Окошки позволяют ввести в них текст, а клавиша нужна мне для выполнения расчетов – нажал клавишу, и посчитал. В первое окно буду записывать первое число, во второе окно – второе. Результат отобразится в третьем. Для получения доступа к объектам я открываю **Toolbox** в разделе основного меню **View**. Можно нажать клавишу F6, как во многих программах. Затем, используя окно диалога свойств, оно открылось при запуске, но можно его вызывать из того же пункта основного меню **View (Properties)**, я нахожу в свойствах окошек TextBox1 – TextBox3 пункт **Text**, где и записаны эти имена, и удаляю ненужный мне текст. Это не обязательно, но тогда в работающей программе это придется делать вручную каждый раз, когда программа запускается. Но это все не столь важно, хотя на рисунке ниже можно увидеть, где удаляется лишняя надпись. Одновременно можно заметить, что изменился вид рабочего окна.

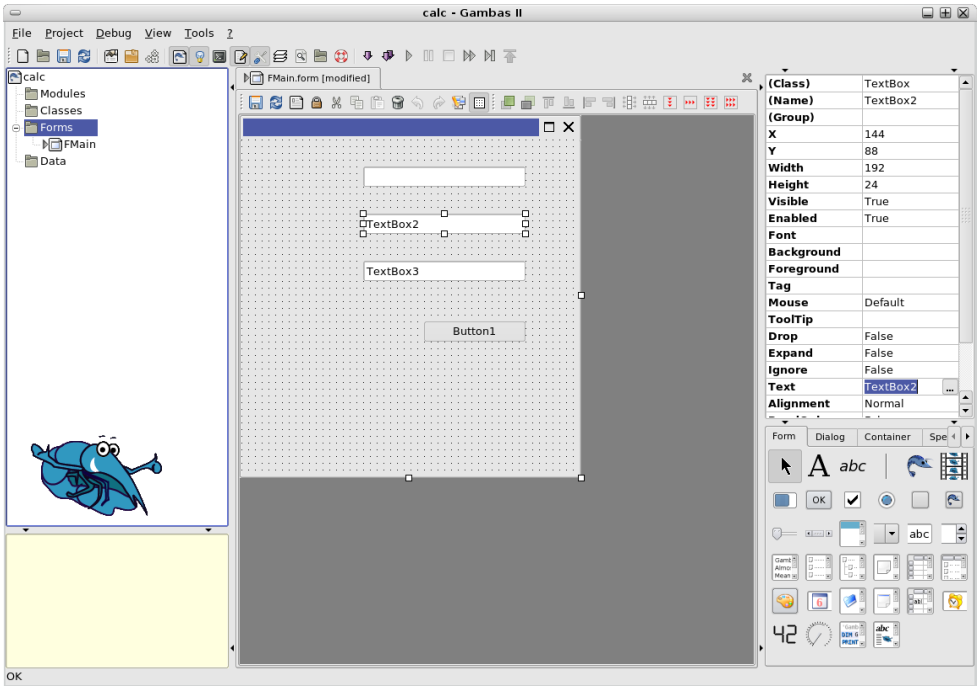


Рис. 1.27. Подготовка формы программы калькулятора

Проделав подготовительные операции по расчистке формы, можно, дважды щелкнув по клавише **Button1**, открыть редактор кода с готовым заголовком процедуры. Альтернатива двойного щелчка, и для меня единственная находка в поисках того, как добиться других возможностей ввести события, это щелкнуть по клавише **Button1** правой клавишей мышки. Выпадающее меню содержит много полезных пунктов, и, в частности, первый из них для меня полезный пункт **Code**, всегда ломал голову, как перейти в редактор кода без добавления ненужных процедур, и пункт **Event**, который и позволяет выбрать множество других событий, а не только щелчок клавишей мышки по кнопке действия в программе.

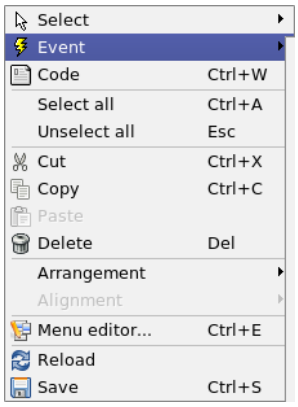


Рис. 1.28. Выпадающее меню после щелчка по объекту Button

И пункты выбора и упорядочивания следовало бы, и хотелось бы, попробовать, но не сейчас. Я думаю, что еще очень много интересного и полезного встретится по пути. А пока, вот как выглядит набор возможных событий, связанных с любой кнопкой, помещенной на форму:

Рис. 1.29. Перечень возможных событий, связанных с объектом Button

Если выбрать событие Click, то получится такой же результат, как и после двойного щелчка по кнопке Button1 на форме. Но и это сейчас не важно, а важна строка, которую я впишу в редакторе кода:

```
TextBox3.Text = Val(TextBox1.Text) + Val(TextBox2.Text)
```

Как можно убедиться ниже это единственный текст кода, который я ввожу в программу, да и то с большими подсказками со стороны Gambas, как и в других, впрочем, интегрированных средах разработки.

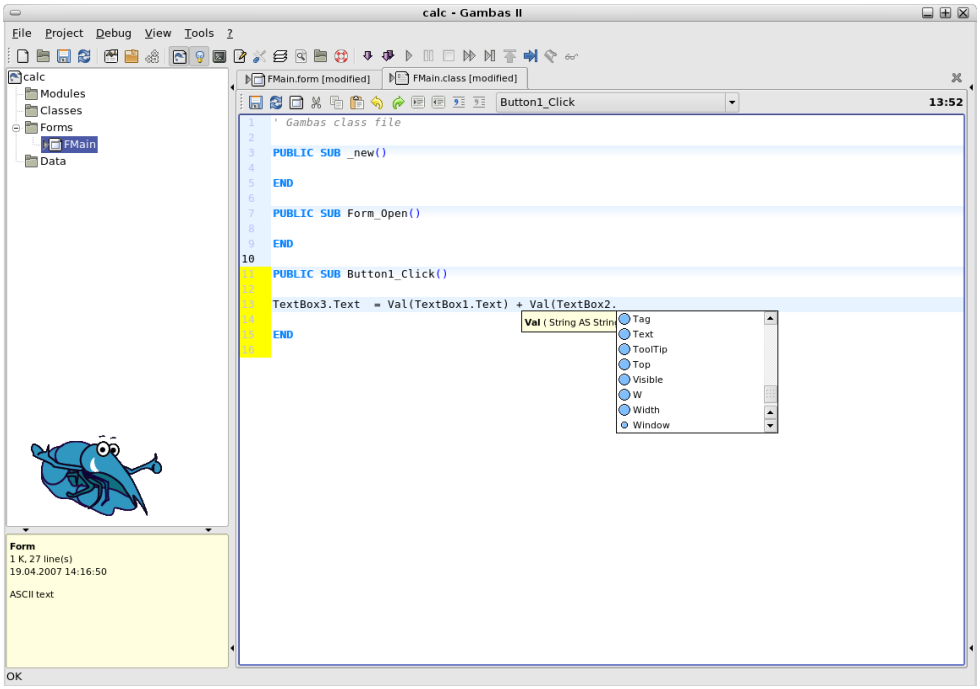


Рис. 1.30. Ввод кода программы в редакторе кода Gambas

Этой строки достаточно для того, чтобы запустив проект получить возможность ввести любое число в верхнее, а второе в окошко ниже, если это нецелые десятичные числа, то следует использовть запятую при вводе числа, а затем, нажав на кнопку Button, получить результат операции, в качестве которой я выбрал сложение, но мог выбрать и другую операцию.

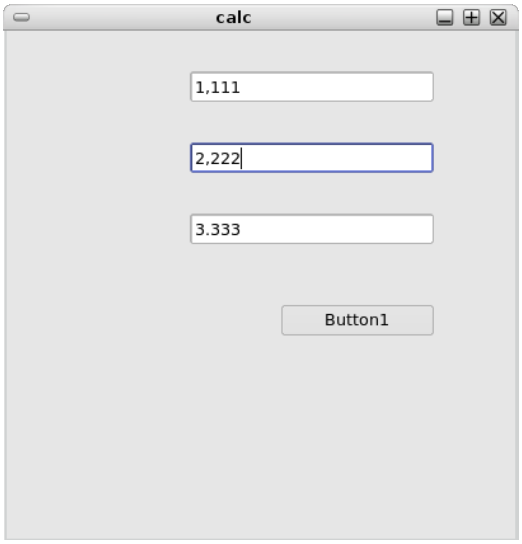


Рис. 1.31. Работа «калькулятора» в Gambas

Вот тот удивительнейший для меня результат, на возможность получения которого открыл мне глаза С.Н. Лукин. Всего одна строка кода, немного перетаскивания картинок «с места на место», и работающая программа. И никогда бы я не догадался, как написать эту строку кода. Я уверен, что стал бы создавать переменные, в которые передавал бы введенные числа, складывал бы их, а потом пытался отобразить в треьем окне. И уверен, что подобная

возможность, не что иное, как свойство объектно-ориентированного программирования. Мне не надо заботится о множестве процедур, мне достаточно добавить к программе нужные объекты – окна ввода чисел и вывода результатов, и записать что я хочу от программы: «Получить результат сложения двух чисел, которые введены в первые два окна». Здесь **Val ()** – это преобразование строки, а ввожу я в калькулятор текстовую строку, в число. При этом мне даже не надо заботиться о типе получаемого числа. Если я ввожу целое число, то получаю тип **Integer**, ввожу дробное число, получаю тип **Float**, с плавающей точкой. По свойствам введенной мной строки Gambas определяет тип данных.

Даже в простом случае, когда приходится подбирать сопротивление нужной величины, а под рукой нет нужного номинала, перебирая сопротивления приходится либо без конца измерять мультиметром результат выбора «на вскидку», либо щелкать по клавишам обычного калькулятора. Создать же работающую программу, которая поможет выбрать резистор нужной величины из тех, что есть в наличии, дело нескольких минут. Вот и подумаешь, а не проще ли потратить немного времени на освоение любого языка программирования, а затем активнее использовать свои знания в практической работе, чем следуя свойству своей натуры (будем называть вещи своими именами – лени), тратить много больше времени и сил, пытаясь делать это по старинке, вручную.

А можно с этого места поподробнее?

Можно. Если имеется ввиду подробнее о среде программирования Gambas. Или если имеется ввиду подробнее о том, зачем вообще программировать, когда перечень готовых программ, как платных, так и бесплатных, с краткими аннотациями займет множество пухлых томов. Или, наконец, подробнее о выборе языка и среды программирования.

Всем этим вопросам я намерен посвятить оставшуюся часть книги.

Чуть подробнее о Gambas. Язык и среда программирования написаны молодым французом Бенуа Минисини (Benoit Minisini) из предместий Парижа для собственного удовольствия, которое он получает как от создания компиляторов, так и от игры на флейте, и от обучения актерскому мастерству, и от работы в качестве профессионального программиста. На работу над проектом в течение нескольких лет его подвигла не только любовь к программированию, но и те идеи, то воплощения языка Бейсик, что были реализованы в Visual Basic. Ему настолько это все не понравилось, что он решил создать свой собственный вариант современного языка Basic. Что и осуществил. Логичным завершением компилятора стало создание графической среды программирования, написанной уже на языке Gamabas. Рост популярности операционной системы Linux, как рабочей среды персональных компьютеров, привлекает все большее внимание профессионалов к языковым возможностям, существующим для этой платформы. Среди профессионалов не только программисты, но и педагоги. Если первые используют любой язык, подходящий к их задаче, то вторые больше связаны с истоками любого языка программирования, и охотнее обращаются к языкам, задуманным именно для обучения. В итоге, если заглянуть в документацию по Gambas, то можно найти множество имен, так или иначе связанных с развитием этого проекта. А что проект бурно развивается можно судить даже по тем рисункам, что приведены выше.

Как любая программа, предназначенная к работе в составе операционной системы, Gambas поддерживает привычный для пользователя интерфейс – ту графическую пользовательскую среду, а сегодняшние операционные системы не обходятся без нее, которая привычна для пользователя, которая помогает ему быстро найти нужный раздел в меню, быстро начать работу над собственной задачей.

Рис. 1.32. Основное меню и основная инструментальная панель Gambas

Пункт меню **File**, который обычно служит для работы с файлами, в данном случае служит для работы с файлами проекта: **New project...** (создание нового проекта), **Open project...** (открыть проект), **Open recent** (открыть недавний проект), **Open example** (открыть пример), **Save project** (сохранить проект), **Quit** (и выйти).

Для работы над проектом, помимо множества возможностей явно не выведенных в области основного меню, служит раздел **Project**:

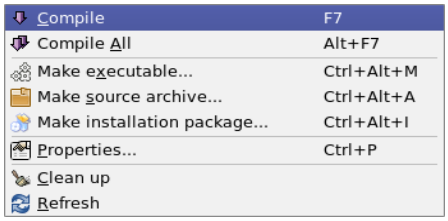


Рис. 1.33. Подменю раздела Project

Это, соответственно, компиляция файла, компиляция проекта, создание исполняемого файла, создание архива исходного текста, создание установочного пакета, свойства проекта, очистка и обновление проекта.

Два пункта меню могут быть непривычны для пользователей Windows – создание архива исходных текстов и установочного пакета. Но не следует забывать, что и программы и сама операционная система всегда существуют в виде открытых исходных кодов. А свободное распространение программ обеспечивается не мощью современных средств «промывания мозгов» под условным названием «реклама», а необходимостью, полезностью и доступностью программ. В Linux можно легко установить программу двумя способами, либо используя исходный код и средства операционной системы для создания и установки программы, либо используя готвый установочный пакет, подобный Setup в Windows. Правда в разных дистрибутивах Linux такой установочный пакет может быть разным и не всегда его можно установить дважды щелкнув мышкой по установочному пакету, но это сегодня скорее исключение из правила, чем правило. Но если вам очень повезло и вы столкнулись с этим исключением, тогда вам поможет предыдущий вариант установки программы из исходных кодов. Это значительно дольше, но универсально.

Следующий пункт меню относится к отладчику вновь создаваемой программы на Gambas, и имеет более или менее стандартный набор средств отладки, таких как запуск, пауза, остановка работающей программы, шаг программы и передвижение вперед, и очистку всех точек остановки.

Далее следуют разделы вида – **View** (открывающий и закрывающие все окна диалога рабочей области), инструментов (**Tools**) и подсказок (**?**). В последнем разделе достаточно подробный **Help browser**, основанный на Wiki в последней версии программы. Мне, конечно, не хватило терпения «прощелкать» все пункты подсказки, но предыдущая версия имела **Help**, состоящий более чем из 3000 файлов, среди которых ряд статей и множество примеров.

Основная инструментальная панель, как это принято, повторяет наиболее часто используемые разделы основного меню, имеет всплывающие подсказки назначения клавиш и

дает возможность таким «торопыгам», как я, быстрее нестись к завершающей стадии своей работы – мучительным раздумьям: «Почему ничего не получается?». Очень быстро нестись.

В новой редакции программы графическая работа над формой проекта и кодом программы может вестись быстрым переключением с помощью ярлычков, на рисунке ниже, Fmain.form и Fmain.class, щелкая мышкой по которым попадешь в графический редактор или редактор кода. Каждый из них имеет свое основное инструментальное меню. На рисунке ниже – инструментальное меню графического редактора.

Рис. 1.34. Инструментальное меню графического редактора и ярлычки перехода

Кроме обычных средств, повторяющих средства основного меню, таких как сохранение и обновление, откат и возврат, удаление и вставки из буфера, панель имеет и некоторые специфические компоненты, такие как вход в редактор меню. Ведь почти все формы, которые при работе программы станут окнами программы, должны иметь основное меню. Именно редактор меню позволяет создавать иерархию основного меню с его разделами и подменю.

Остальные клавиши позволяют управлять размещением объектов на форме, выравниванием их размеров. Кроме переноса объектов на форме с помощью мышки, захватив объект после его выделения и размещения курсора над ним, вернее с помощью удержания левой клавиши мышки и перемещения с помощью мышки, так вот кроме этого объекты можно перемещать с помощью клавиш перемещения клавиатуры, что очень помогает, когда нужны точные движения. У меня, например, мышка обычно либо убегает в одну строну, либо в другую, я долго прицеливаюсь, а когда, наконец, попадаю в нужное место, то от движения при отпускании клавиши мышки, та сдвигается с места и все нужно начинать сначала. Если бы ни клавиатура, я бы всю жизнь создавал кривые рисунки и схемы.

Я думаю, что по мере создания сложного проекта со множеством форм и собственных классов, это дополнительное меню будет расширяться, позволяя быстро перемещаться по проекту. Но для этой же цели можно использовать и окно «дерева» проекта.

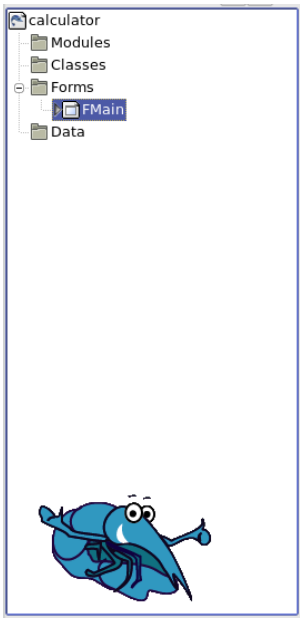


Рис. 1.35. Окно дерева проекта

У меня оно почти пустое, что, впрочем, в полной мере может служить отражением моих познаний в части программирования. Но это так, к слову. И пока не забыл, хочу еще привести одну концепцию, почерпнутую мною из книги С.Н. Лукина, которой постараюсь следовать сам, не следует пытаться сразу создать весь проект, лучше делать это частями, открывая новые проекты (из старых) по мере продвижения, отлаживать и сохранять каждую часть, продвигаясь пусть скромными, но методическими шажками к своей цели. Отлаживая процедуры, если это возможно, «в гордом одиночестве». Я назвал бы это «штучным производством». Он совершенно прав. Но лучше почитать самому. Книжка не дорогая, и, думаю, весьма будет полезна и в начале изучения языка, и в дальнейшем.

Отдельную область занимает инструментарий проекта – **Toolbox**. Построенный аналогично рабочей области, он тоже имеет различные ярлычки, открывающие страницы, на которых размещаются средства создания проекта. Моему разумению пока доступны такие графические элементы как клавиши, текстовые окошки, переключатели и иже с ними. Но если вас не пугают такие ужасы, как контейнеры, можете пощелкать по закладкам и рассмотреть все страницы.

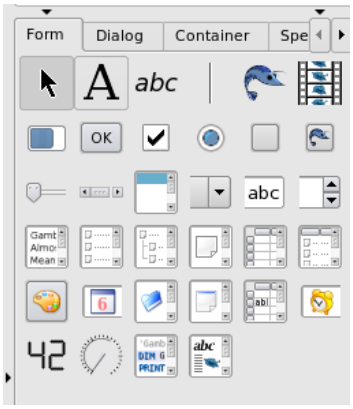


Рис. 1.36. Вид инструментария проекта

А сейчас я зажмурюсь, у меня высотобоязнь, когда я выхожу на лестничную клетку покурить, то часто присаживаюсь у окна с очень низким подоконником, иначе голова начинает кружиться, а в ногах появляется слабость. Зажмурюсь и отрою свойства проекта. Это сделать несложно – вошел в раздел основного меню **Project**, выбрал пункт выпадающего меню **Properties**, нажал левую клавишу мышки и ты уже в свойствах проекта, где есть закладка **Components**. Вот такой «высоченный» список.

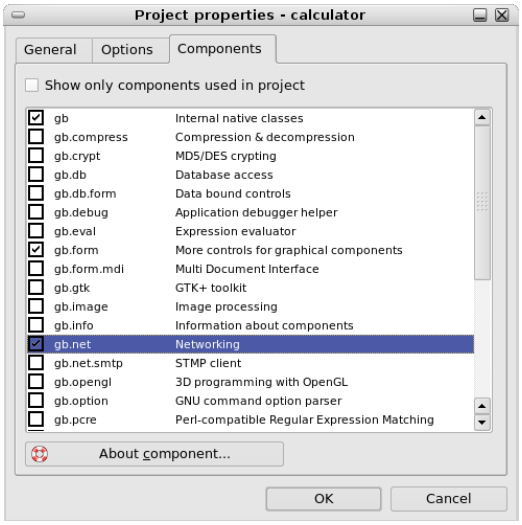


Рис. 1.37. Окно списка доступных компонет Gambas

Если выбрать еще один компонет из списка, то инструментарий пополняется еще одной страницей с закладкой.

Рис. 1.38. Пополнение инструментария проекта после добавления компонента

Но если вы подбно мне не готовы стать монтажником-высотником, всегда можно выключить ненужные компоненты, оставив только то, что ясно и нужно в данный момент.

Первый блин

Даже профессионалы, как правило, специализируются на чем-то. Один предпочитает, если говорить о программировании, работать с базами данных, другой предпочитает системное программирование. Специализация позволяет очертить круг возможных задач и уточнить цель работы в плане выбора и инструментов, и заготовок. Всегда полезно использовать отлаженную и проверенную заготовку, если она подходит для выполнения предстоящей задачи. В этом смысле нам дилетантам приходится тяжелее. И задача не слишком ясна, и цель не видна, и из заготовок – пустой ящик с одним погнутым гвоздем (мои зарисовки в программе Umbrello).

А вот и не так, неправ я. Есть заготовка под названием «Калькулятор». Мало того, я нашел компонент Gb.net, который, надеюсь, даст возможность использовать COM-порт компьютера, как я и намеревался. Попробуем из этих двух находок сделать нечто похожее на заготовку для

проекта «Счетчик». Назову я его, на всякий случай, «Counter». И еще раз, а то стал забывать о чем идет речь, перечислю, что я хочу сделать:

- Создать графический интерфейс с помощью Gambas для проверки двоичного счетчика с предустановкой.
- С помощью интерфейса будет вводиться число, которое при нажатии клавиши, положим, «Запись» будет переписываться в счетчик.
- Для выполнения этой и последующих операций компьютер по COM-порту будет соединяться с микроконтроллером, отдавая последнему соответствующую команду, которую он будет обрабатывать.
- После переписи числа в счетчик микроконтроллер будет прочитывать число на выходе счетчика и отправлять результат на компьютер, а графический интерфейс отображать это число.
- Интерфейс будет иметь клавишу запуска счета, после нажатия на которую счетчик должен начинать отсчет, микроконтроллер должен прочитывать результат и предавать компьютеру.
- Вероятно, нужно будет иметь еще одну клавишу интерфейса для того, чтобы открыть COM-порт и закрыть.

Пока этого хватит. Для выполнения задуманного я, как описано выше, добавлю еще один компонент, сетевой, с помощью диалогового окна свойств проекта, и, как и в калькуляторе, установлю два окна **TextBox**, две клавиши **Button** и еще одну, о свойствах которой пока ничего не знаю. Называется она **ToggleButton.** Если она поможет мне переключать COM-порт, хорошо, нет, заменим. В результате форма выглядит следующим образом:

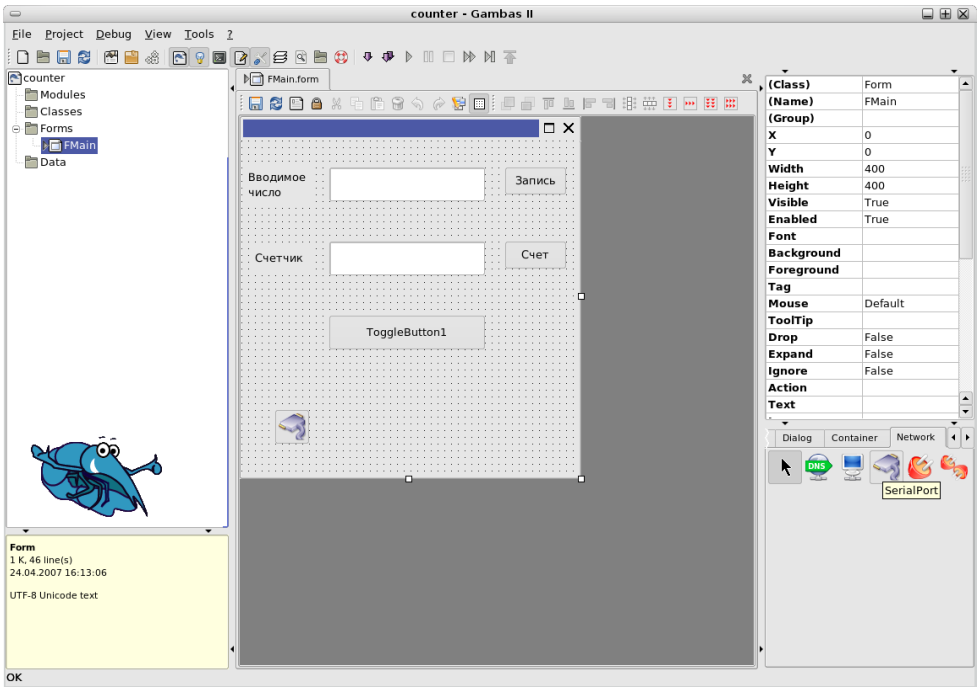


Рис. 1.39. Вид формы для проекта «Counter»

На форме, слева внизу, виден добавленный мной компонент для работы с COM-портом, на инструментальной панели компонент надписью **SerialPort** отмечено, откуда я его извлек. Теперь самое время обратиться к разделу помощи программы, чтобы прочитать, как пользоваться компонентом последовательног порта, и что такое клавиша **ToggleButton**.

Выяснение последнего вопроса с помощью подсказки не прояснило для меня ситуацию, пришлось проверить, как это работает с помощью самой программы добавив в процедуру отработки щелчка по клавише строку вида схожего со строкой калькулятора.

```
PUBLIC SUB ToggleButton1_Click()
      TextBox1.Text = ToggleButton1.Value
END
```

При нажатой клавише она остается «включена», при повторном нажатии «выключается», а ее значение в виде текста при нажатии получается как «T», а при выключении ничего не видно. Пока решим, что при нажатии клавиша получает значени True. Впрочем, это можно проверить, заменив эту строку на такой код.

```
IF ToggleButton1.Value THEN
      TextBox1.BackColor = Color.Blue
ELSE
      TextBox1.BackColor = Color.Red
ENDIF
```

Теперь при нажатии на клавишу верхнее окошко становится синим, а после повторного нажатия становится красным.

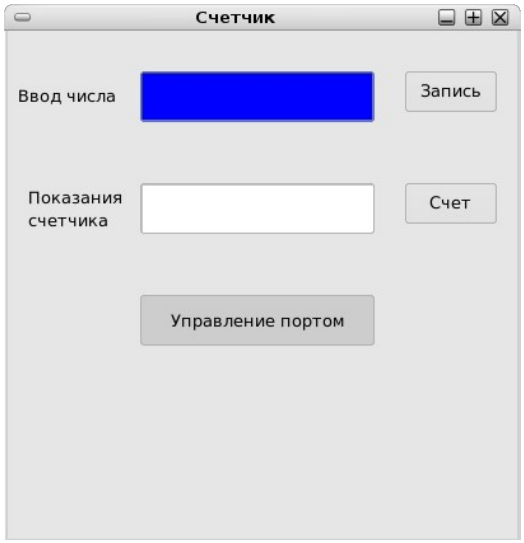


Рис. 1.40. Проверка работы клавиши ToggleButton

Посмотрим, как включать и выключать COM-порт, как задать, какой именно порт использовать, и как установить его свойства.

По началу я хотел задавать все свойства порта перечислением этих свойств в программе. Потом я подумал, что можно использовать окно свойств компонента, где эти свойства, скорость работы и количество бит и т.д., можно установить. Что и сделал. Теперь по нажатию кнопки на форме следовало включать, а при повторном нажатии выключать порт.

Увы! Я перебрал несколько вариантов, как мне казалось разумных строк в программе, но, если отбросить явные ошибки, наилучший результат выглядел сообщением об ошибке при открывании порта. Код программы выглядел с моей точки зрения правильно, но кого волнует моя точка зрения?

```
PUBLIC SUB ToggleButton1_Click()
```

```
      IF ToggleButton1.Value THEN
            SerialPort1.Open
            TextBox1.Text = SerialPort1.Status
      ELSE
            SerialPort1.Close
            TextBox1.Text = SerialPort1.Status
      ENDIF
      IF SerialPort1.Status THEN
            TextBox1.BackColor = Color.Green
      ELSE
            TextBox1.BackColor = Color.Red
      ENDIF
END
```

В первом окне для проверки я хотел бы вывести состояние порта после выполнения операции открывания порта. Затем, в подтверждение этого, если порт открыт, изменить цвет этого окошка на зеленый. При закрывании порта цвет должен смениться на красный.

Не получилось.

Просмотрев все советы, которые я нашел, посмотрев то, что я делал в Visual Basic, когда работал над книгой об «умном доме», я не нашел ничего, что указывало бы на ошибку. Осталось проверить несколько вариантов, первым из которых я решил провести эксперимент по смене операционной системы, благо на компьютере у меня стоит другой дистрибутив – Ubuntu. Перезагрузив компьютер, я открываю программу в Gambas, который у меня установлен в обеих ОС, запускаю программу, нажимаю на клавишу, которая должна открывать и закрывать порт, и получаю результат:

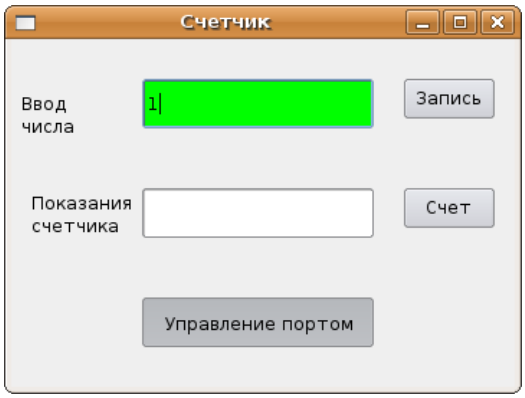


Рис. 1.41. Первая удачная попытка открыть COM-порт

В этом дистрибутиве Linux все работает соответственно моим ожиданиям. Сообщения об ошибке при открывании порта не появляется.

Я вновь возвращаюсь к своему основному дистрибутиву, и вновь получаю сообщение об ошибке, но теперь уже с полной уверенностью, что программа ДОЛЖНА работать. И направление поиска есть.

Я не знаток операционных систем, чтобы утверждать, что это особенность именно Linux – строгое распределение прав доступа ко всем файлам и папкам. Но, если щелкнуть правой клавишей мышки по любой папке или файлу, открыв файловую систему, и выбрать из выпадающего меню пункт свойств, то можно на странице прав доступа увидеть эти права.

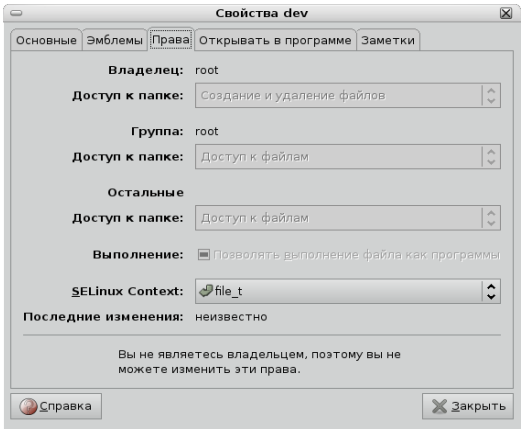


Рис. 1.42. Свойства папки dev на странице прав доступа

Не знаю, как устроен Linux, но для меня вся его файловая структура – это набор папок и файлов. В частности и COM-порт, который имеет название ttyS0, и который расположен в папке dev файловой структуры, для меня просто файл с правами доступа. И, похоже, к этому файлу у меня доступа нет. Результат ли это особенностей дистрибутива, или результат моих экспериментов, не знаю. Но эту ситуацию можно попытаться изменить. Чтобы увериться в правильности выбранного пути, я завершаю свой сеанс работы и вхожу «под root'ом», то есть, вхожу как администратор компьютера. Теперь, запустив программу в Gambas, я не получаю сообщения об ошибке, программа ведет себя ровно так же, как в Ubuntu – работает, выдает все, что я «заказывал».

В основном меню графической среды Gnome, которую я использую, в пункте **Система** есть раздел **Администрирование**, в котором есть пункт **Пользователи и группы**. В открывающемся диалоговом окне можно добавлять и удалять пользователей, можно изменить состав групп пользователей, что меняет их права доступа к различным ресурсам компьютера.

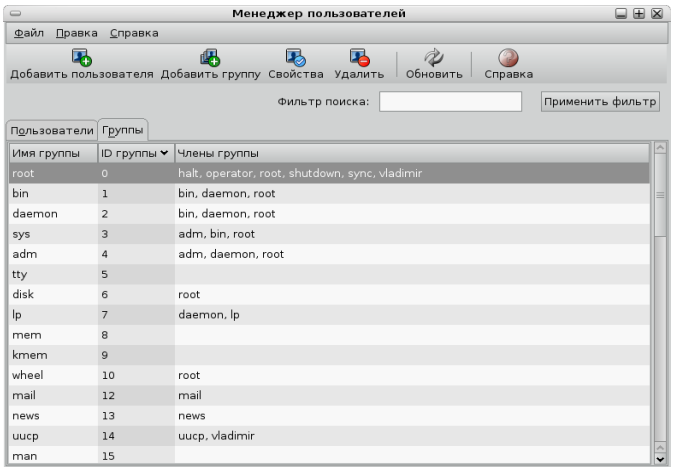


Рис. 1.43. Диалоговое окно управления пользователями и группами

Выбрав пользователя root, администратора компьютера, и нажав на клавишу **Свойства** основного меню дилога, можно войти в диалог свойств пользователя, где есть страница, определяющая, кто входит в административную группу.

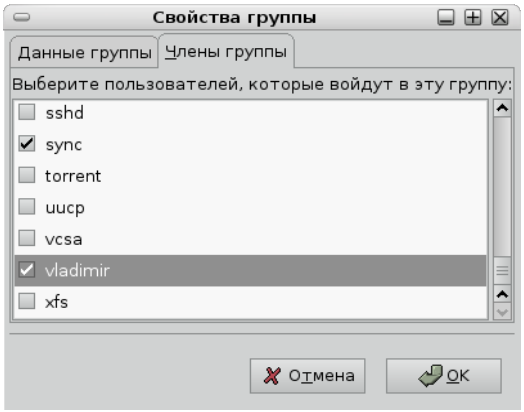


Рис. 1.44. Диалоговое окно определения состава административной группы

В этом окне диалога мне достаточно поставить галочку (щелкнуть левой клавишей мышки по квадратику напротив имени пользователя) рядом с собой, чтобы получить права доступа к тем ресурсам, которые может использовать администратор. Конечно, в рамках того, как обозначены эти права для административной группы, поскольку они могут отличаться от прав главного администратора.

После присвоения себе «повышенных» прав я снимаю вопрос об ошибке при открывании последовательного порта. Порт открывается, но работает ли он, правильно ли он работает, я намерен проверить после того, как завершу ту часть проекта, которая относится к микроконтроллеру. Именно с ним, а не сам с собой, я намереваюсь беседовать при помощи созданного интерфейса. Кроме того, интерфейс придется дополнить чем-то, что переключало бы скорость счета, а код программы дополнить всем необходимым для диалога интерфейса с микроконтроллером. Хотя мне не терпится проверить работу COM-порта, и такая возможность есть – я могу воспользоваться прежней наработкой, достаточно перенести готовый код для микроконтроллера в программу Piklab, запрограммировать контроллер и подправить код в программе «Счетчик», но... Если бы я занимался разработкой, я так и сделал бы, а в рассказе мне это представляется слишком большим жульничеством. Пожалуй, придется потерпеть до конца следующей главы.

Глава 2. Бряцая железом

Можно сколь угодно долго спорить о том, что важнее «курица или яйцо», то есть, программирование или разработка и пайка схем. Но споры, даже научные, носящие благозвучное имя «дискуссия», быстро раздражают собеседников, поскольку основным аргументом в споре всегда является взаимное неприятие аргументов оппонента. Зачастую, сколь бы вдумчиво я ни пытался вникнуть в предмет спора, мне остаются непонятны ни истоки, ни существо спора. Мне кажется, что каждый волен выбирать то, что ему симпатичнее, и кажется правильным. И это совсем не значит, что выбрав что-то одно, он не может, если есть время и желание, попробовать другое. Право выбора остается за человеком всю его жизнь в любой момент. Всегда следует пользоваться этим правом. Но я опять отвлекся.

Хорошее начало

Итак. Первое приближение интерфейсной части проекта готово. Теперь время поговорить о «железной» части проекта. Я предполагаю использовать микроконтроллер PIC16F628A. Два основных аргумента в пользу этого выбора – наличие самой микросхемы, и наличие всего необходимого для работы с микросхемой, то есть, среда разработки с поддержкой языка Си и программатор, который успешно работает в этой среде. Я приведу ниже рассказ о среде программирования и схему программатора.

Но начать я хочу с того, как выглядела бы схема проекта, если бы была реализована вне ухода в программную сторону. Для этой цели, правда, я использую программу EDA, что много проще рисунка на бумаге с последующим сканированием этого рисунка.

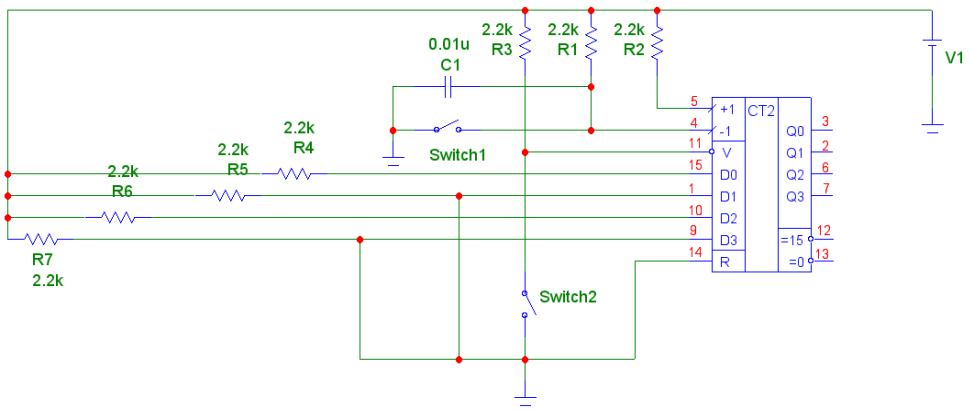


Рис. 2.1. Схема проведения прямой проверки работы счетчика

Теперь о программе Piklab. К сожалению, когда я в предыдущей книге описывал работу с микроконтроллером, я не нашел этой программы, хотя, с другой стороны, программа MPLAB великолепна, ничего плохого сказать не могу.

Для работы с программой, особенно при использовании компилятора языка Си, потребуются дополнительные пакеты. Если программа устанавливается стандартным способом, то пакеты gputils и gpsim, скорее всего установятся по зависимостям. Так установилась программа в моем дистрибутиве Fedora Core 6. Второй дистрибутив Ubuntu не предложил ничего, но есть пакет для Debian, и в этом случае приходится доустанавливать

дополнительные утилиты, хотя этот процесс проходит быстро и безболезненно, в частности требуются пакеты для графической оболочки KDE. Пакет gputils необходим для программирования на ассемблере, gpsim для отладки программы. Для работы с языком Си можно воспользоваться бесплатным компилятором picc-lite, который можно найти на сайте www.htsoft.com. Я использую версию picc-lite_9.50PL2. Компилятор знаком мне по работе с программой MPLAB под Windows.

Схему программатора я выбрал под названием JDM picprogrammer. С программой работают и другие программаторы, хотя я проверял только один, купленный специально для проверки такой возможности при работе над предыдущей книгой. Чтобы не отсылать вас на поиски схемы в Интернет, или к своим прежним книгам, я приведу схему простого программатора в оригинальном виде. При сборке программатора я обнаружил, что забыл купить стабилитрон на 8.2 В, а стабилитронов на 5.1 В купил в избытке. По этой причине я к имевшемуся стабилитрону последовательно подключил в два светодиода (из расчета, что при прямом включении они дадут мне поведение стабилитрона с напряжением стабилизации около 3 В). Такой вариант сработал, и оказалось, что иметь светодиоды в этой цепи даже полезно – при обращении к программатору они светятся, видно, что программатор работает.

На схеме переделанный стабилитрон – это D6.

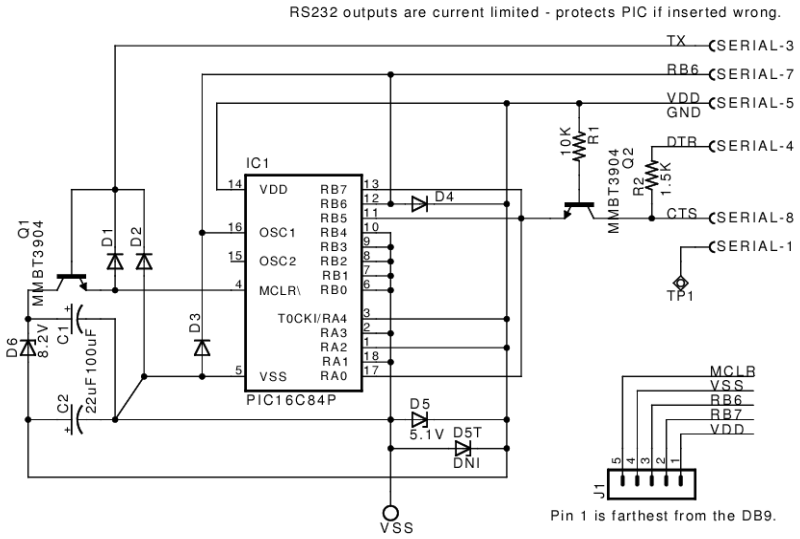


Рис. 2.2. Схема программатора для работы с программой piklab

Сейчас я извлеку программатор из «хламежки», и подключу его к первому COM-порту...

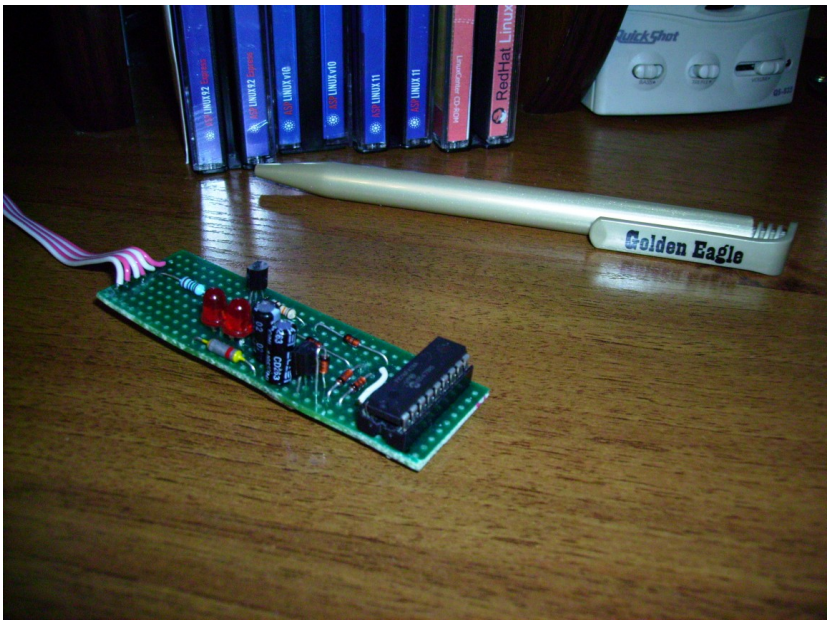


Рис. 2.3 Внешний вид программатора

Итак, первый запуск программы Piklab.

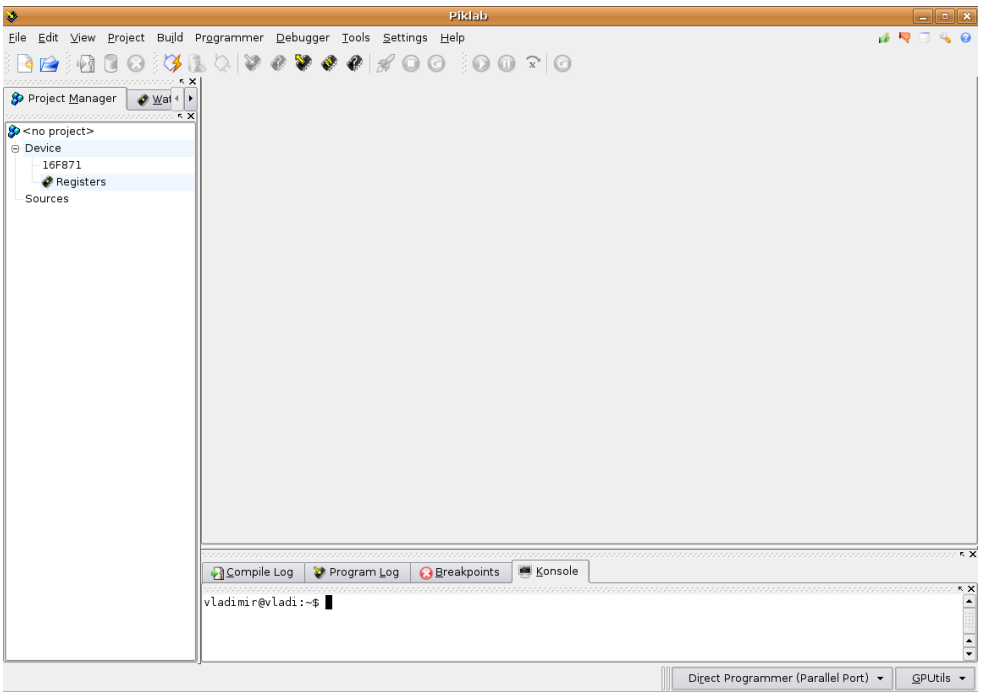


Рис. 2.4. Первый запуск программы Piklab

Внешне среда программирования микроконтроллера, как вы можете судить, очень похожа и на среду программирования Gambas, и на среду программирования KDevelop, если вы знаете такую. Что не удивительно, поскольку обе программы созданы для работы с одной операционной системой. Слева окно менеджера проекта, центральное место занимает встроенный редактор программирования, а в нижней части расположены окна сообщений, включая окно для работы с точками останова и консолью. На рисунке окно консоли открыто, в окне можно ввести любую команду.

Начну с основного меню программы. Кроме обычных для любой программы разделов меню, как раздел работы с файлами, кроме обычных разделов, характерных для любой среды программирования, связанных с отладкой программы, есть и специфический раздел, относящийся к работе с программатором.

Рис. 2.5. Основное меню программы Piklab

Хотя первым в списке разделов меню стоит **File**, относящийся к работе с файлами, я зайду в раздел **Settings**, раздел установок программы.

Рис. 2.6. Подменю раздела Settings

И более всего меня пока интересует установка программатора. По причине чего я захожу в раздел **Configure Piklab**..., чтобы открыть установки, относящиеся именно к программатору.

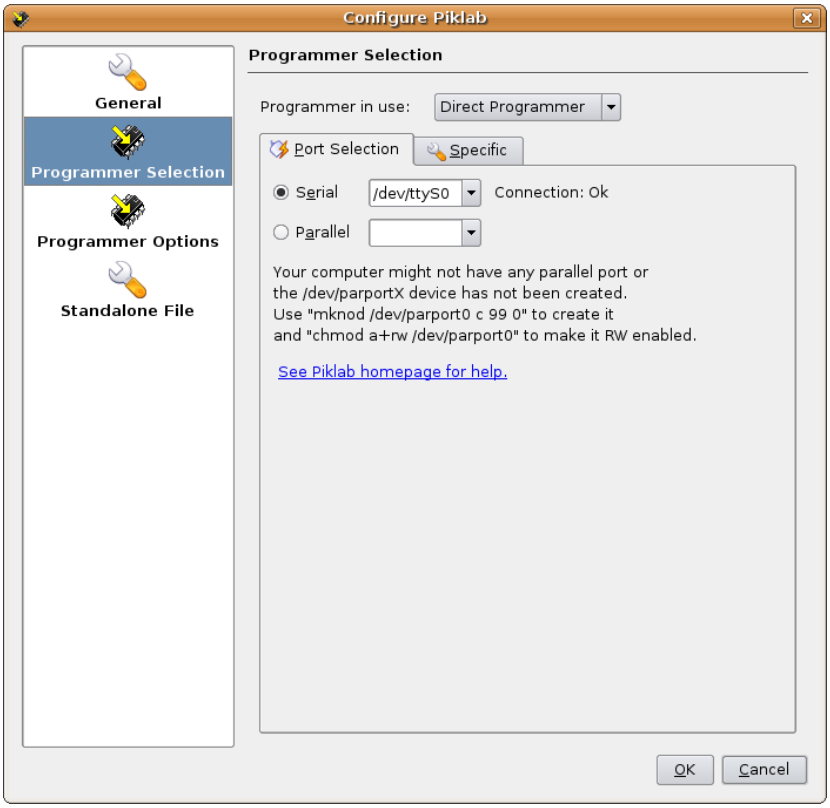


Рис. 2.7. Диалоговое окно установок программатора

После выбора последовательного порта COM-1 (ttyS0 для Linux) и типа программатора

Direct Programmer я сразу вижу, что рядом с надписью **Connection:** появляется ОК, то есть порт обнаружен. А следом за выбором типа контроллера в окне менеджера проекта (**Progect Manager**) я могу прочитать содержимое микроконтроллера, который остался в панельки со времен предыдущих экспериментов. Для этого можно зайти в раздел основного меню **Programmer**, на всякий случай выбрать пункт **Connect**, проверить, чтобы на панели состояния отображался правильный выбор **Direct Programmer (Serial Port)**, и можно прочитать содержимое микроконтроллера.

Успокоенный этими удачными пробами, я собираюсь плавно перейти к рассказу об основном меню программы и соответствующих клавиш основного инструментального меню. Но мне приходит в голову проверить процесс отладки программы. Без достаточно удобной отладки работать с программой будет слишком трудно. Я знаю, что для отладки используется gpsim, но как это осуществить в программе piklab? Файл помощи, который я пытаюсь открыть, показывает, что документация отсутствует. В Интернете на странице проекта есть краткое введение в работу с программой (я ищу работу в графическом приложении), описание, касающееся этой части выглядит весьма кратким:

> *After your successful compilation (as you can see in the Compile Log) you only have to press the button **connect** to "virtually connect" the programmer **Gpsim**, to be able to start the simulation of your program. Just press the blue button with the "play" symbol, to start. After this, the **Program Log** shows some information output from gpsim.*

Что означает, примерно, следующее:

> *После успешной компиляции программы (как можно видеть в Compile Log) вы должны только нажать клавишу **connect** для «виртуального подключения» к программатору **Gpsim**, чтобы иметь возможность начать симуляцию вашей программы. Достаточно нажать синюю клавишу с символом «проиграть», чтобы начать симуляцию. После этого **Program Log** покажет некоторую выходную информацию от gpsim.*

Я повторяю эти действия вначале после трансляции программы на Си (я всегда для проверки использую простую и удобную программу, которую некогда нашел в Интернете, и уже, к сожалению, не помню автора). В окне Program Log появляется сообщение:

```
Connected.
Setting up debugging session.
Ready to start debugging.
Running...
```

И это все, что я вижу. Этот же результат и при трансляции ассемблерной программы. Этот результат дают оба дистрибутива, установленные на моем компьютере. Я прерываю работу на несколько дней, понимая, что если каким-то образом не решить эту проблему, то следует остановиться на этом месте и честно сказать себе: «Строили мы строили, и, наконец, построили!». Книга неожиданным образом завершена, добавить нечего.

Можно попытаться связаться с авторами проекта, но они могут быть либо заняты, либо могут охладеть к работе над проектом. В любом случае ответа от них можно ждать долго и не дождаться. Ассемблерный вариант выглядит несколько лучше в этом плане – при трансляции он дает файл с расширением .cod, который можно загрузить в gpsim, который, в свою очередь, запускается у меня из терминала. Но едва ли вам захочется сразу начать работу с микроконтроллером, используя ассемблер. Это возможно только в том случае, если вас

интересует именно работа на ассемблере. Для остальных удобнее использовать более легкое создание программ на языке высокого уровня. Беда же в том, что компилятор не дает такого файла. Стоп работа!

Возвращение

За это время я еще раз перечитал документацию, перебрал варианты, твердо решил, что бросаю эту книгу, еще раз решил попробовать как-то обойти проблему. Один из возможных вариантов заключался в том, чтобы использовать полученный после трансляции hex-файл для дизассемблирования, а последний файл с программой на ассемблере повторно транслировать, чтобы получить .cod файл, необходимый для загрузки в симулятор работы микроконтроллера gpsim. Сознаюсь честно, что я даже пытался установить программу MPLAB под Wine (эмулятор работы Windows). Но работала только очень ранняя версия для Widows 95, с которой проблем могло оказаться не меньше, если не больше. Последняя версия MPLAB работать не захотела. Чтобы честно сказать, что я перепробовал все, я даже поставил программу VirtualBox – великолепно работающую виртуальную машину для Linux, в которой можно установить из iso-образа на своем компьютере другой дистрибутив Linux или Windows. В ней я установил последнюю версию MPLAB, где попробовал установить последнюю версию транслятора фирмы HI-TECH, обнаружив, что этот транслятор не входит в перечень, устанавливаемых по умолчанию, хотя несколько лет назад это было не так. Я вспомнил, что несколько лет назад, пытаясь работать с микроконтроллером в Linux я, похоже, оказался в сходной ситуации, пробовал установить другой компилятор, но безуспешно, и стал работать в MPLAB. Но с тех пор могло измениться многое.

По умолчанию на моем компьютере загружается дистрибутив Fedora Core 6. Там я и начал поиск другого компилятора для работы с программой piklab. На домашней странице проекта в разделе документации я быстро обнаружил краткое описание настроек sdcc – компилятора, используемого с piklab. Возможность установить этот компилятор в Fedora есть, я его скачиваю, устанавливаю, но не могу настроить. Его не обнаруживает программа piklab.

Порою я бываю упрям без меры. Видимо, это упрямство заставляет меня перезагрузить компьютер и войти в дистрибутив Ubuntu, построенный на основе Debian. В этом дистрибутиве я повторяю установку компилятора sdcc, и он великолепно обнаруживается программой piklab. Мало того, он работает, есть возможность работать с микроконтроллером PIC16F628A, то есть с полной памятью контроллера. И самое главное, он при трансляции дает cod-файл, который можно загрузить в симулятор gpsim.

Порою я бываю упрям без меры. Видимо, это упрямство заставляет меня перезагрузить компьютер, возвращаясь к основной моей операционной системе Linux в дистрибутиве Fedora Core 6 (если этим летом не сменю его либо на привычный мне ASPLinux, либо на Fedora 7). Я загружаю исходный текст компилятора, чтобы использовать штатный режим (в моем понимании) создания программы из исходных кодов. Это не слишком сложно, хотя может занимать достаточно много времени, если программа достаточно большая и сложная. Процедура для меня сводится к тому, чтобы запустить терминал (больше, но аналогично командной строке в Windows). В терминале командой **cd /home/vladimir/Desktop/sdcc/** перейти в папку, где находятся исходные тексты. Эта папка лежит у меня на рабочем столе. Дальше обычная для Linux команда **./configure** и следом **make**. Следом, не значит, сразу. В этом месте, после команды **make** трансляция исходного текста прекращается. Насколько я понимаю нужен дополнительно пакет **yacc**. Такого пакета программа установки не обнаруживает. Но есть пакет **byacc**. Устанавливаю его, вновь запускаю трансляцию командой **make**, трансляция продолжается и успешно завершается. Теперь, поскольку у меня не работает команда **sudo**, опять правовая недостаточность, хоть в евросуд обращайся, хотя что

толку от этих лицемеров, опять приходится обходить проблему. Обойти ее, впрочем, не сложно. Вместо команды **sudo** использую команду **su**, и после ввода пароля даю команду **make install**, как и положено. Как и положено, «оттарахтев» минут пятнадцать, трансляция успешно завершается. И теперь программа piklab обнаруживает компилятор sdcc. И он работает так же удачно, как и в Ubuntu.

Конечно, меня огорчает, что симулятор не запускается, как я ожидал, из программы работы с текстом, что-то я не допонимаю в этом, видимо, но для реальной работы такой вариант даже удобнее – работать на одном рабочем столе все равно не получится, а так можно разложиться со всеми удобствами на двух рабочих столах. Есть еще одна небольшая деталь, касающаяся дистрибутивов. В Fedora не работает механизм добавления модулей, который работает в Ubuntr, но это решается добавлением пакета gpsim-devel, что не трудно сделать с помощью программы установки и удаления приложений.

Моя ошибка, то что я не подготовился в должной мере, несколько выбила меня из колеи, но теперь есть все необходимое для продолжения работы: программатор, компилятор, симулятор. Ничто не мешает продолжить повествование о назначении разделов меню программы piklab.

Хотя первый пункт основного меню привычный, относящийся к работе с файлами проекта, в силу особенностей проекта в данном случае, если открыть этот раздел, то обнаружится отличие предлагаемых действий от обычных.

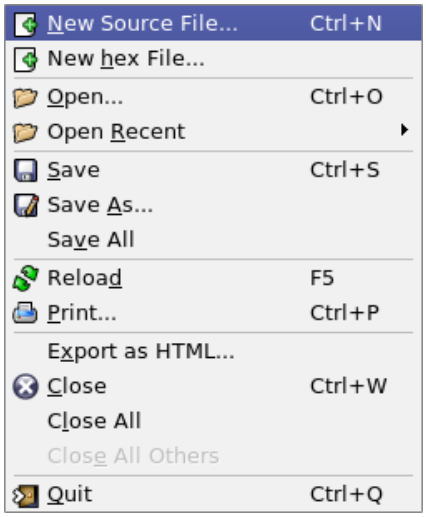


Рис. 2.8. Подменю раздела File основного меню

Первый пункт подменю **New Source File...** подразумевает создание нового исходного файла программы. Однако лучше создать новый проект. Следующий пункт **New hex File...** подразумевает создание нового готового для загрузки в микроконтроллер файла кодов. Это еще рано. Далее следуют два пункта меню, позволяющих открыть файл и открыть недавно использованный файл. Затем три пункта сохранения файлов, практически не отличающихся от аналогичных в любом приложении. Что означает **Reload** я пока не знаю, а остальные пункты меню – вывод на печать, экспорт в виде HTML, закрывание файлов и выход из программы, – говорят сами за себя.

Следующий раздел касается возможностей работы в редакторе текста программы.

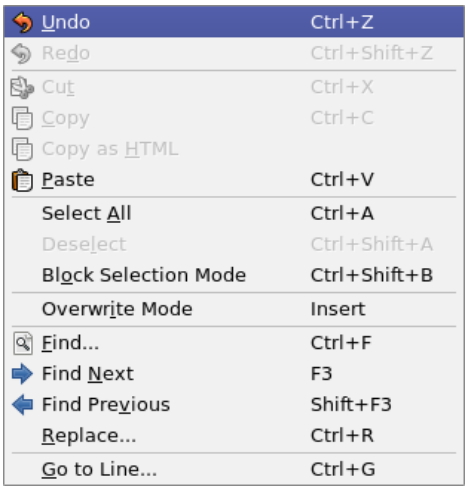


Рис. 2.9. Подменю раздела Edit основного меню программы piklab

Возможности эти соответствуют возможностям любого достаточно мощного текстового редактора, включая отмену последней операции **Undo**, повтор отмененного действия **Redo**, вырезание текста **Cut**, копирования **Copy** и вставки **Paste**. Небольшое отличие – копирование в формате HTML (**Copy as HTML**). Меню чувствительно к операциям с текстом. Например, пока вы не выделите слово или фрагмент текста, операции копирования и вырезания текста не будут активированы. Это удобно, если учесть, что в процессе разработки приходится многократно проделывать одни и те же операции, быстро щелкая по разным разделам меню, и мышка не всегда поспевает за вашими мыслями, отчего, щелкнув по одному из разделов (как вы полагали), вы в действительности щелкаете по другому, который расположен чуть выше или ниже. В подобной ситуации, чем меньше активных разделов, тем лучше.

Редактор можно конфигурировать. При появлении файла с текстом программы подменю раздела **Settings** пополняется пунктом **Configure Editor...**

Раздел выделения кроме обычного выделения всего текста **Select All**, и снятия выделения **Deselect**, имеет режим выделения блока текста **Block Selection Mode**. Чем это удобно? Я говорил, что из-за отсутсвия файла для загрузки в симулятор компилятор HI-TECH заставил меня задуматься о повторной трансляции ассемблерного кода. Но для получения ассемблерного кода из файла листинга, например, удобно применить такой режим, как выделение блока текста, иначе придется вручную удалять много ненужного.

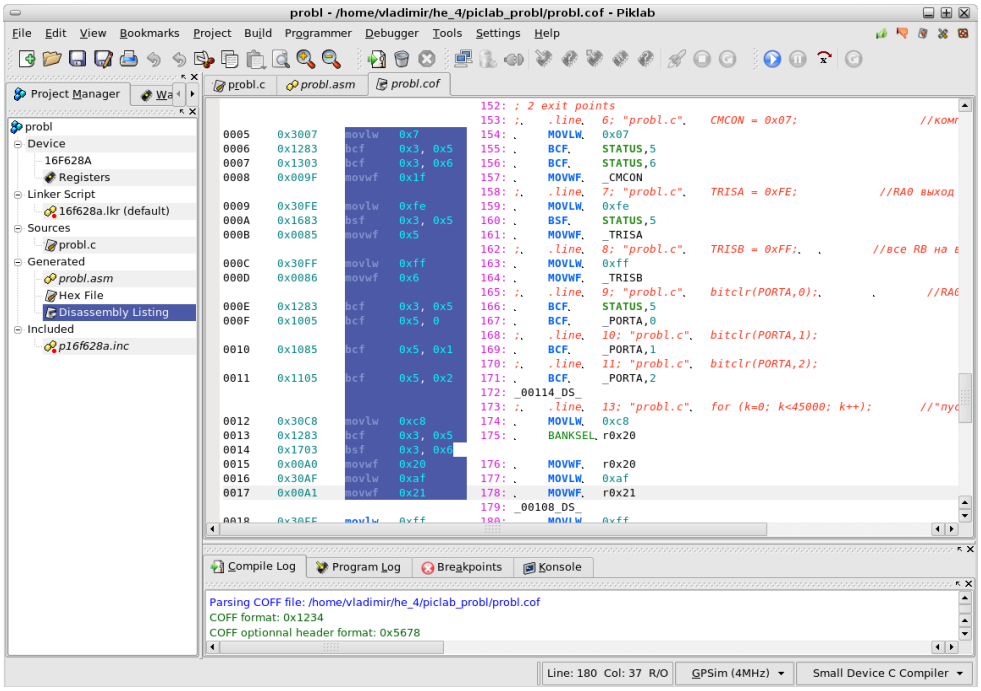


Рис. 2.10. Режим выделения блока текста в редакторе piklab

Выделив в блоке текста только программу, написанную на ассемблере, вы можете вставить ее в новый файл, перенося нужный фрагмент в одно действие. Совсем не лишняя возможность.

Далее в подменю следует изменение режима ввода текста: вставка-замещение. Затем функции поиска **Find** и замещения текста **Replace**. Вы, как мне кажется, не раз пользовались этими функциями при работе в текстовом процессоре, так что ничего нового я вам не скажу. И завершает подменю редактирования функция перехода к номеру строки, **Go to Line...** При отладке программы это может оказаться очень полезным. Забегая вперед хочу сказать, что редактор позволяет включить отображение номеров строк. Сообщения при компиляции, относящиеся к ошибкам, обычно соотносятся с номером строки, и переход по номеру строки не будет лишним.

Подменю следующего раздела основного меню Вид (View) относится к виду открываемых окон редактирования.

Рис. 2.11. Вид подменю View в режиме редактора текста

Так выглядит это подменю, когда вы редактируете текст программы, но его содержание изменится, если у вас открыт hex-файл.

Команды **Back** (Назад) и **Forward** (Вперед) относятся к навигации, как в любом обозревателе, например, web-браузере. Когда работаешь с маленькими проектами дополнительные возможности редко используются. Обычно предпочитаешь обходиться небольшим набором хорошо знакомых операций. Но даже маленькие проекты имеют свойство «разбухать» как-то сами по себе. Ты и не хотел этого, но это происходит. И тогда есть смысл присмотреться к различным возможностям, которые предоставляет тебе программа. Это же замечание я готов отнести к разделу **Switch to...** подменю **View**. Каждому открытому файлу соответствует закладка, что позволяет быстро переключиться с файла на файл. Эту же операцию легко сделать в окне менеджера проекта. Но если вы заинтересуетесь работой с программой, работой с микроконтроллерами, вы не остановитесь на создании только одиночных проектов, вам наверняка захочется создавать удобные, проверенные и отлаженные программные модули, которые вы будете включать в новые проекты как объектные модули. Их может набираться много. Переходы с помощью закладок уже не будут выглядеть такими удобными. Попробуйте переключение с помощью этой команды.

Следующий раздел подменю имеет свое собственное подменю, позволяющее отображать или выключать разные окна проекта. Это особенно полезно, когда щелкая по клавишам управления открытыми окнами закрываешь окно по ошибке, и не знаешь, как открыть его вновь. А это окна менеджера проекта, окно наблюдения, вывод компиляции и программирования контроллера, точек останова и окно консоли.

Следующий раздел основного меню позволяет вам сделать закладки (**Bookmarks**) или удалить их. Текст программы быстро становится достаточно длинным, чтобы «кручение» колесика мышки перестало доставлять вам радость. Проще поставить закладки и переходить к ним при необходимости. Меню тоже чувствительно к вашим действиям. Если вы не делали закладок, то увидите только один активный раздел подменю – **Set Bookmark**. Но вид подменю изменится, как только вы сделали закладку.

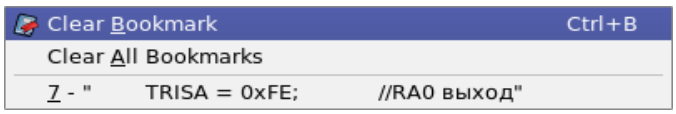


Рис. 2.12. Вид подменю закладок в тексте программы после создания закладки

Следующий раздел меню **Project** напоминает мне о необходимости немного поворчать.

Самое разумное, что следует делать при работе над проектом, это поэтапная его реализация. Начав работу, не стоит пытаться написать код всей программы. Лучше написать часть кода, ограниченную рамками разумной достаточности, проверить и отладить эту часть проекта. После чего сохранить проект, а затем создать новый проект, куда можно включить уже отлаженную часть или отложить это до более подходящего времени. Иначе, как это часто получается у меня, проект, содержащий ошибки, при попытке их устранить обретает множество новых, не менее увлекательных ошибок, а, в конце концов, превращается в полное собрание моих ошибок. Вернуть его к первоначальному виду, который хотя бы как-то работал, задача для такого атлета, как Сизиф, но для меня.

Выход простой – почаще использовать первый пункт подменю раздела **Project** – **New Project...** Далее в подменю есть пункты открывания существующего проекта (**Open Project...**), и недавно использовавшегося проекта (**Open Recent Project**).

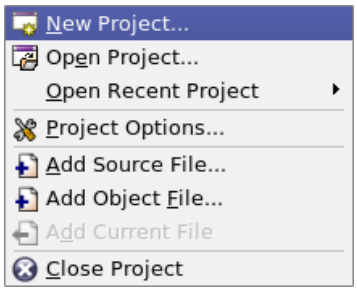


Рис. 2.13. Подменю раздела Project

Перемахнув через один из пунктов, я хочу закончить обзор этого подменю, сказав, что **Add Source File...**, **Add Object File...** и **Add Current File** относятся к заполнению проекта файлами. Первым добавляется файл исходного текста, например, ассемблерного или Си, вторым добавляется объектный файл, а последний добавляет в проект текущий (активный) файл, который вы могли создать с помощью меню **File**. Последний пункт закрывает проект, и если вы не сохранили его перед этим, откроет диалог сохранения файлов. Тот пункт меню, о котором я не упомянул, а именно, **Project Options...**, открывает диалоговое окно установок проекта.

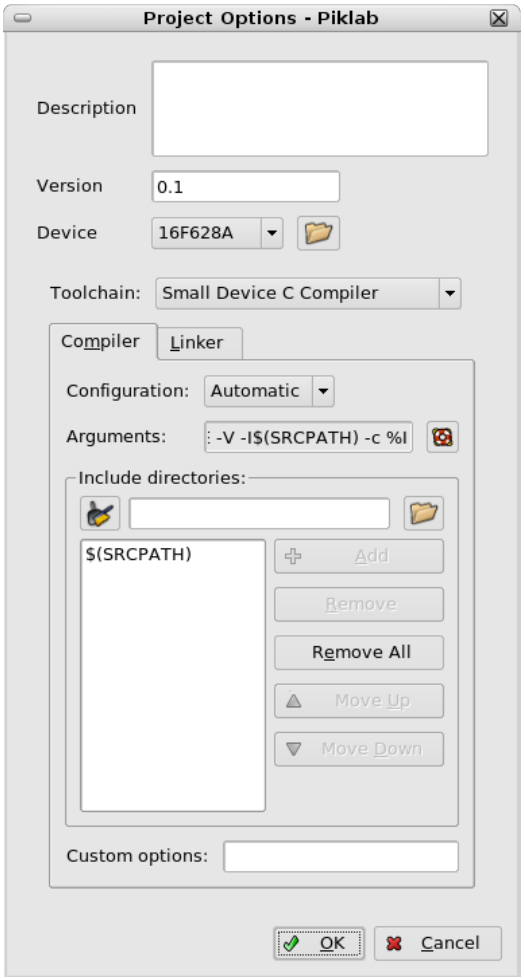


Рис. 2.15. Диалоговое окно установок проекта

Здесь можно выбрать ваш микроконтроллер, транслятор языка, добавить описание проекта. Опции, относящиеся к таким возможностям, как аргументы или пользовательские

опиции, скорее для опытных разработчиков, я лично, предпочитаю их не трогать до тех пор, пока не начну в полной мере осознавать последствия этого эксперимента.

Следующий раздел основного меню **Build** позволяет вам «построить» свой проект: скомпилировать только один файл, обработать весь проект или очистить проект, если вы многократно (и безуспешно) пытались транслировать написанный вами код. Последний пункт **Stop** относится, насколько я понимаю, к останвке процесса трансляции кода. Если текст кода большой, его трансляция может занимать достаточно много времени, а чтобы не ждать завершения, когда вы вспомнили, что собирались заменить имена переменных, но забыли это сделать, можно остановить трансляцию.

Следующий раздел основного меню относится к работе с программатором. Его установку можно выполнить в разделе **Settings**, а работу осуществить либо с помощью меню, либо использовать клавиши основной инструментальной панели. После выбора программатора подменю выглядит следующим образом:

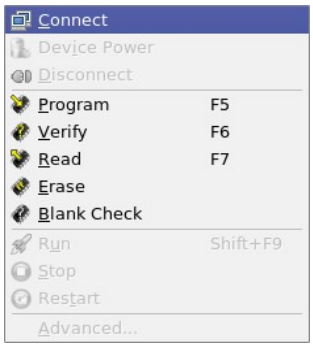


Рис. 2.16. Вид подменю работы с программатором

Первый пункт – подключение программатора, после активизации подключения возможно отключение (**Disconnect**). Процесс программирования (**Program**) отображается в нижней части экрана проекта. Далее следует проверка (**Verify**), чтение (**Read**), очистка (**Erase**) и проверка очистки. К чему относятся остальные, не активизированные пункты меню, я пока не разобрался, но, возможно, они предназначены для дальнейшего расширения проекта.

С программированием микроконтроллера у меня есть небольшая «заморочка». Перед программированием я не могу задать слово конфигурации (по адресу 2007). Чтобы задать его, мне приходится прочитывать микросхему, после чего я могу вписать необходимое мне значение (3F18), которое выглядит теперь как 2118, и записать его. Не самая сложная операция. Но почему это не получается сделать сразу?

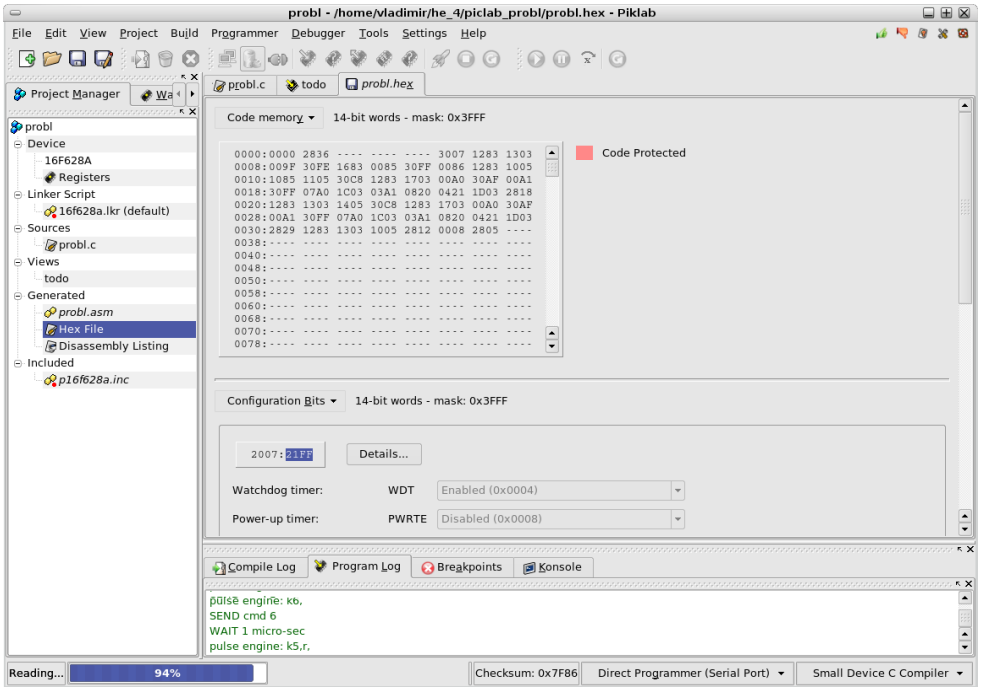


Рис. 2.17. Отображение процесса чтения из микросхемы

Я пропущу раздел **Debugger**, поскольку решил отладку программы осуществлять непосредственно в программе симуляции gpsim, иначе отладка у меня не работает, да и удобнее, мне кажется, работать с двумя программами на разных рабочих столах.

За разделом **Debugger** следует раздел **Tools** – инструменты программы piklab. Количество доступных инструментов и активность разделов зависят от того, с каким файлом вы в данный момент работаете. Наибольшее количество пунктов в подменю появляется при работе с исходным текстом на Си.

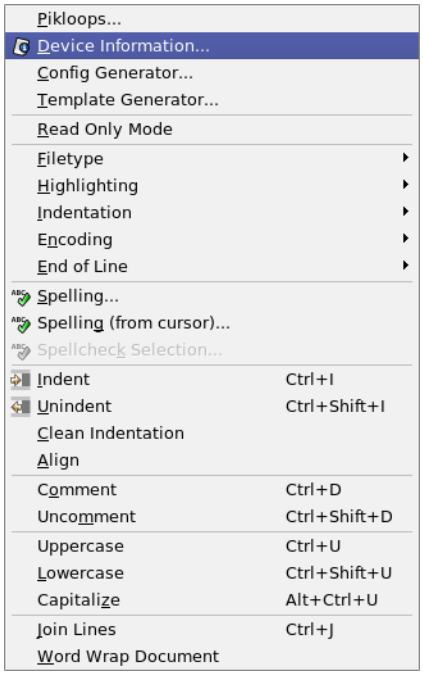


Рис. 2.18. Подменю раздела инструментов программы piklab

Pikloops – это дополнение к программе, позволяющее получать коды на ассемблере для организации временных циклов. Если это приложение установлено, то выбор этого пункта подменю запускает приложение.

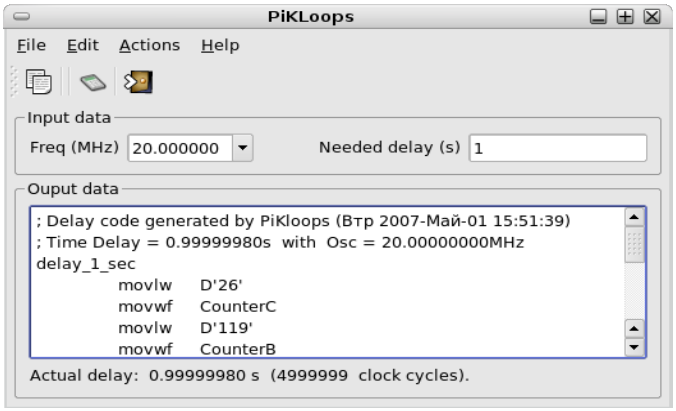


Рис. 2.19. Приложение piktools

Можно задать тактовую частоту микроконтроллера, необходимое время цикла и получить готовый код.

Следующий пункт подменю этого раздела тоже, с моей точки зрения, очень полезен. Я постоянно забываю расположение выводов контроллера, приходится обращаться к документации производителя, или рисовать цоколевку микросхемы. Работая с программой можно обращаться к этому пункту меню – **Device Information...**

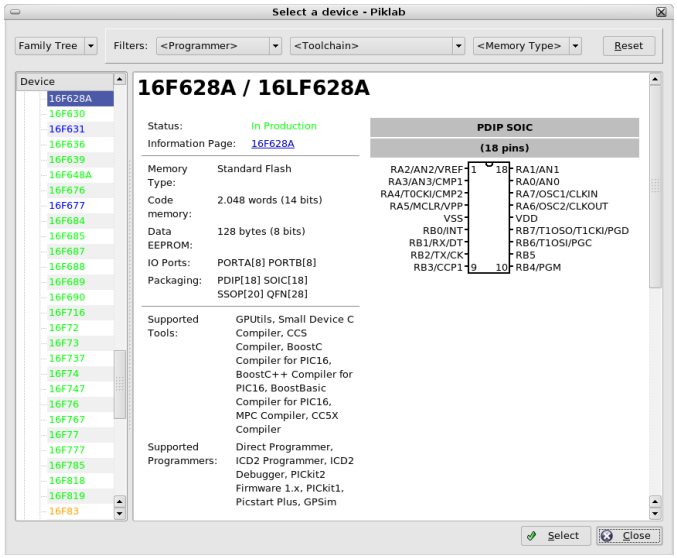


Рис. 2.20. Информация о текущем контроллере

Следующий инструмент раздела **Tools** должен полностью снять вопрос о том, как записать слово конфигурации по адресу 2007. Запуск этого инструмента дает диалог установки всех необходимых бит конфигурации для выбранного микроконтроллера и языка программирования. В окне диалога можно выбрать необходимую конфигурацию, используя клавиши управления установками, а затем скопировать необходимый текст с помощью клавиши **Copy to clipboard**, чтобы, закрыв окно диалога, в текстовом редакторе просто вставить текст в нужное место из буфера обмена.

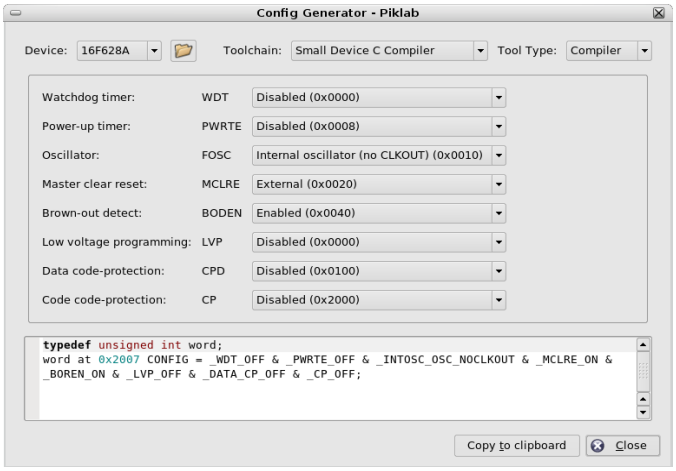


Рис. 2.21. Далоговое окно задания слова конфигурации контроллера

После того, как текст скопирован и вставлен в текст программы, а сама программа откомпилирована, слово конфигурации будет записано при программировании микросхемы. До этого «открытия» мне приходилось поступать иначе. Теперь буду знать.

Хотя я чаще пользуюсь тем, что было сделано ранее, чем начинаю проект «с нуля», видимо есть смысл использовать и те возможности по оформлению кода программы, которые предоставляет piklab. В частности возможность генерировать шаблон, правильно понимаемый компилятором. Для этих целей служит инструмент под назаванием **Template Generator...**

Аналогично тому, как в диалоговом окне генерации слова конфигурации, в окне диалога шаблона для выбранного языка программирования создается шаблон основного текста программы. Его можно скопировать в буфер обмена с тем, чтобы вставить в текст программы.

Можно поступить и иначе, используя мастер создания нового проекта, где на одной из страниц диалога есть возможность сразу установить опцию шаблона для выбранного языка программирования и используемого транслятора кода программы. Оба варианта доступны. Использовать инструментальное меню можно при создании проектов, содержащих более одного файла текста кода.

Использование пункта подменю **Read Only Mode** (режим только чтения) в данный момент не актуально, но при работе над программой может и понадобиться.

Далее следует достаточно длинный перечень средств, которые опытному разработчику сильно облегчат жизнь, но не думаю, что они понадобятся в любительской практике. А если и понадобятся, то все эти возможности можно освоить по ходу дела.

Следующий пункт меню **Settings** я уже упоминал выше. В этом пункте меню можно установить видимость (или невидимость) панелей, задать все свойства прогрммы и программных средств. Этим пунктом меню я пользовался, когда подключал программатор, и тогда, когда подключал компилятор языка Си.

Разделы установок открывают соответствующее диалоговое окно, устроенное достаточно удобно, чтобы не только настроить среду программирования, но и убедиться в том, что настройка произведена. Так, напомню, что когда я немного «заблудился» с выбором компилятора языка Си, когда нашел и установил пакет SDCC (rpm пакет для легкой и быстрой установки), то обнаружил именно в окне диалога, что программа не обнаруживает этого компилятора. И только после повторной установки компилятора Small Device C

Compiler из исходных текстов в разделе **Configure Toolchains...** я увидел, что и сам компилятор, и все необходимые составляющие прогрммой обнаружены.

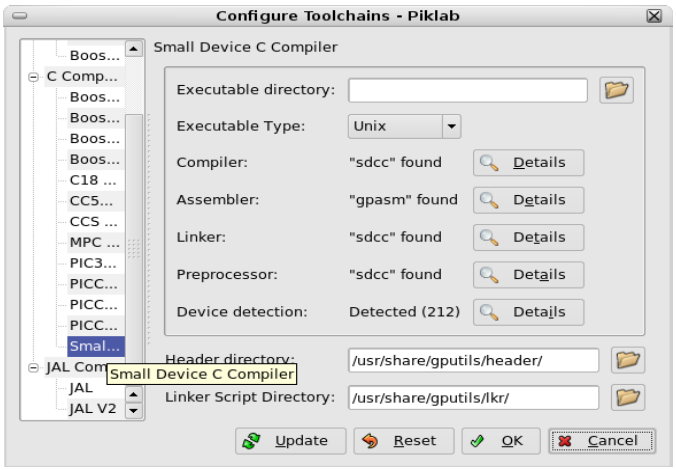


Рис. 2.22. Далоговое окно настройки языка программирования

Как видно из рисунка программа обнаружила и сам компилятор («sdcc» found), и ассемблер, и компоновщик, и препроцессор.

В целом в программе в текущий момент только раздел меню Help оказывается бесполезен. Думаю, это временно, и прежде, чем я допишу книгу, он существенно пополнится.

Основная инструментальная панель программы повторяет наиболее употребительные разделы и пункты меню. Клавиши инструментальной панели активизируются соответственно тому, с каким из файлов в настоящий момент ведется работа, а при отладке активизируются клавиши отладчика, если на нижней панели рабочей области программы выбрать в качестве программатора Gpsim.

Рис. 2.23. Основная инструментальная панель программы piklab

Многие диалоговые окна и подменю основного меню программы чувствительны к типу файла, с которым в текущее время происходит работа. По этой причине я очень рассчитываю на то, что при реальной работе над программой объявятся новые возможности, о которых я буду сообщать по мере их появления. А сейчас пришло время рассказать об отладке программ.

Для этих целей я использую программу gpsim. Как рекомендовано в руководстве, загружать в программу лучше файл с расширением cod, создаваемый при трансляции исходного кода на ассемблере или языке Си.

Отладка в gpsim

Прежде чем перейти к программе отладки, у меня есть желание несколько упростить запуск отладочной программы и необходимость написать подходящий для работы с отладчиком код программы для микроконтроллера PIC16F628A.

Особых трудностей в запуске gpsim из консоли нет, но при многократном повторении это

вызывает некоторое раздражение. Создание кнопки запуска (или ярлычка программы) в Linux не вызывает особенных затруднений – достаточно щелкнуть правой клавишей мышки по рабочему столу и в выпадающем меню выбрать пункт **Создать кнопку запуска...** Как мне помнится, некогда была возможность и включить возможность выполнения команды запуска в терминале. Сейчас такой возможности я не нахожу. Думаю, причина в том, что ранее я в основном работал в графической среде KDE, а не Gnome, как сейчас. Поскольку я использую часть программ, предназначеннных для работы с KDE, базовые пакеты этой графической среды установлены на компьютере, и я могу перейти в KDE, завершив сеанс, и при входе в новый сеанс сменить текущую графическую среду. Что я и делаю.

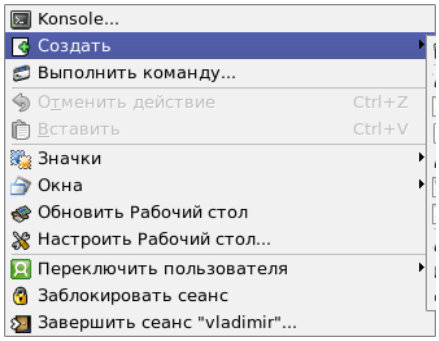


Рис. 2.24 Всплывающее меню в KDE

И при создании кнопки запуска приложения действительно нахожу необходимую мне опцию в разделе дополнительных параметров.

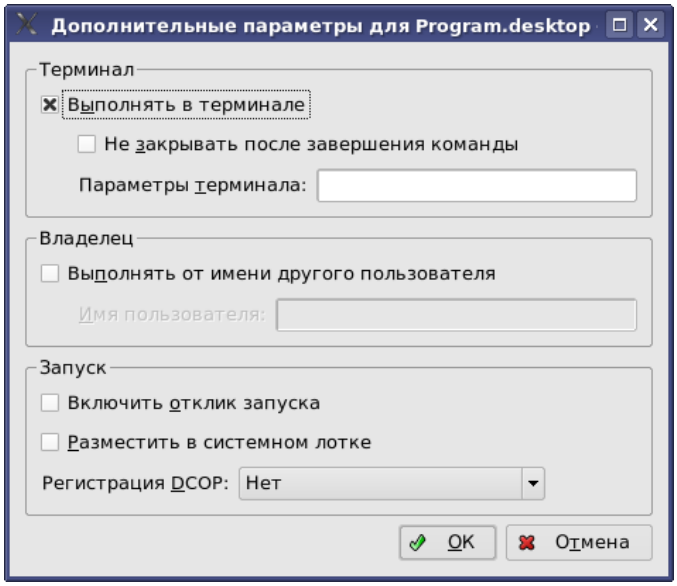


Рис. 2.25. Диалоговое окно создания кнопки запуска приложения в терминале

Теперь по возвращении в Gnome я имею кнопку запуска, которая делает все необходимое, то есть, запускает gpsim в терминале.

Для подходящего кода программы я воспользуюсь тем кодом, что некогда был написан в среде программирования MPLAB. Я скопирую текст программы, удалю все лишнее, и вставлю в новый проект, который создам в piklab. На этой садии работы я хочу, чтобы при получении команды, простой символ «1», отправляемый из графического интерфейса,

написанного в среде Gambas, моя программа микроконтроллера зажигала светодиод, который будет на макетной плате присоединен к одному из выводов порта. А при получении второй команды, тоже в виде простого символа «0», микроконтроллер гасил этот светодиод. Проверочный код программы выглядит следующим образом:

Файл заголовка counter_start.h:

```
#define bitset(var,bitno) ((var) |= 1 << (bitno)) // установка бит порта
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno))) // сброс бит порта
unsigned char getch(void); // прием символа
void     init_comms(); // инициализация коммуникации
```

Основной файл программы:

```
/* ------------------------------------------------------------------------- */
/* Template source file generated by piklab */
#include <pic16f628a.h>

/* ------------------------------------------------------------------------- */
/* Configuration bits: adapt to your setup and needs */
typedef unsigned int word;
word at 0x2007 CONFIG = _WDT_ON & _PWRTE_OFF & _RC_OSC_CLKOUT & _MCLRE_ON &
_BOREN_ON & _LVP_ON & _DATA_CP_OFF & _CP_OFF;

#include <stdio.h>
#include "counter_start.h"

unsigned char input;    // Для считывания приемного регистра
unsigned char COMMAND;  // символ команды
unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}

void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case '1': bitset (PORTA, 0);
            break;
            case '0': bitclr (PORTA, 0);
            break;
            }
}

void init_comms() // Инициализация модуля
{
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x0;
      TRISB = 0xFE;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      bitclr (PORTB, 0);      // Выключаем драйвер RS485 на передачу
}
```

```
void main() {
      init_comms(); // Инициализация модуля

start:      CREN =1; // Начинаем работать
      input = getch();
      cmd();
      goto start;
}
```

Разобраться в коде программы, которую когда писал ты сам, не проще, чем в чужой программе. Читать описание долго, а комментарии достаточно короткие, чтобы были понятны пока работаешь над программой (да и то не всегда), но не дающие понимания через несколько дней после завершения работы над кодом программы. Словом, я постарался выбросить все, что пока не относится к делу (каким это представляется в данный момент), включая часть шаблона для организации прерываний. Что из этого получится, не знаю. Пока я только привел код к виду, когда трансляция проходит без ошибок. А хотелось бы мне, чтобы получая от COM-порта команду «1» на вход RB1 (первый вывод порта B), микроконтроллер зажигал светодиод на RA0 (нулевой вывод порта A), а по команде «0», гасил его.

После трансляции программа piklab формирует нужный мне для отладки файл, поэтому я готов приступить к описанию работы с отладочной программой gpsim.

После запуска программы (столь удачно созданной кнопочкой запуска) мы получаем следующий вид программы:

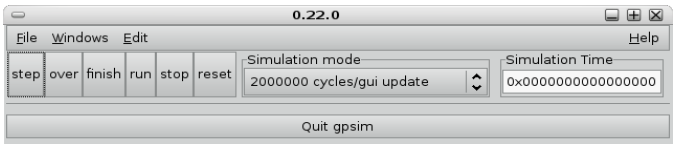


Рис. 2.26. Вид отладчика gpsim после первого запуска

Основное меню содержит три пункта, первый из которых **File** позволяет мне только открыть файл или выйти из программы. Я открываю файл counter_start.cod. И... ничего не происходит. Все-таки это не редактор текста. Но если обратиться к следующему пункту основного меню **Windows**, то открывается доступ к отображению различных рабочих окон отладчика.

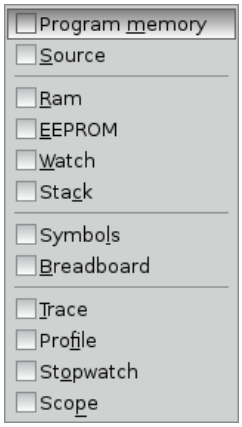


Рис. 2.27. Подменю раздела Windows основного меню отладчика

Для начала активизируем окно **Source**, что должно открыть исходный текст программы. И этот текст открывается. Отладочный текст имеет вид смешанного текста: исходного на языке Си, и оттранслированного на ассемблере. Комментарии видны, что позволит, надеюсь, мне сориентироваться в дремучих зарослях ошибок.

Если теперь запустить программу отладки с помощью клавиши **Run** на основной панели, то анимация быстро замирает. Только счетчик времени симуляции **Simulation Time** продолжает свой счет. Так и должно быть, поскольку вслед за инициализацией микроконтроллер переходит к ожиданию ввода команды через COM-порт компьютера.

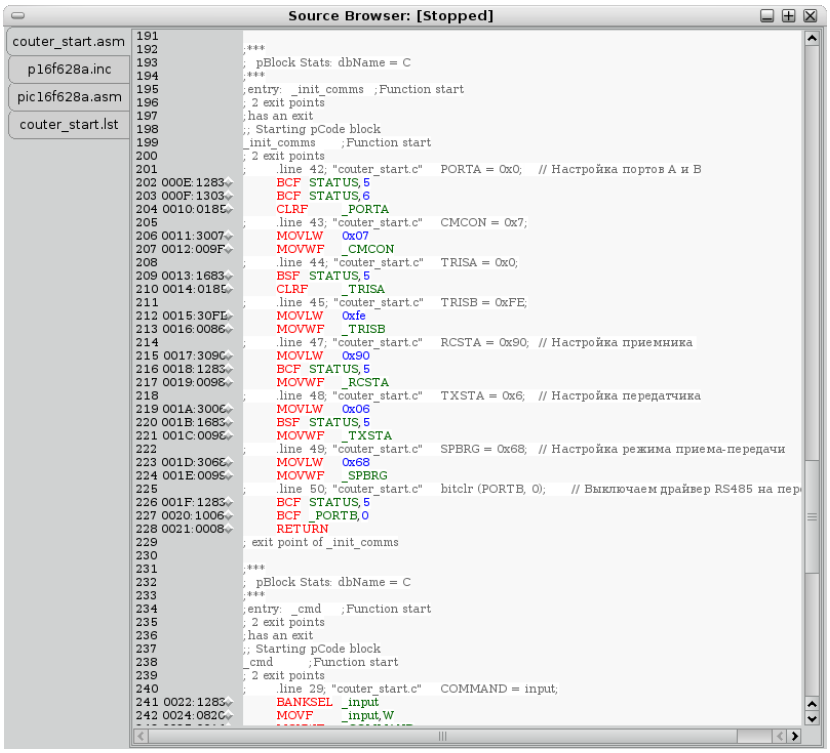


Рис. 2.28. Исходный текст программы, с которым предстоит работа

Мне хватает сообразительности (или я где-то это вычитал) открыть в подменю раздела Windows еще одно отладочное окно – **Breadboard** (макетная плата). На плате, как и положено, есть микроконтроллер. Есть и ряд клавиш, назначение которых мне пока не ясно, но мне хотелось бы поэкспериментировать с этими клавишами. Дело в том, что работая с программой MPLAB, я использовал инструмент отладки, который или назывался **usart**, или эта возможность задавалась в настройках программы. Сейчас я не помню, но хочу отыскать нечто похожее в gpsim. Правильное решение – чтение документации, и неправильное – щелкание по всем клавишам, в конце концов дают свои плоды. И здесь, видимо, уместо еще раз отметить, что при компиляции исходного кода в Ubuntu, я получил дополнительные файлы из исходного текста (или текстов) программы, а в Fedora Core потребовалось добавить пакет **gpsim-devel**. При первом открывании макетной платы она выглядит так:

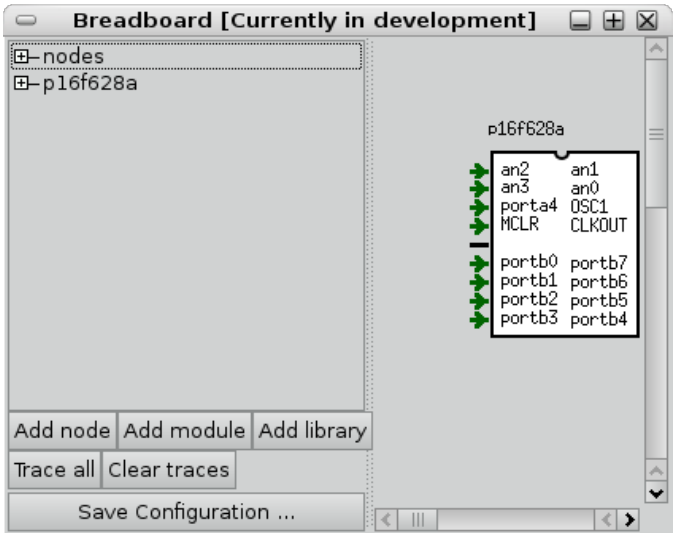


Рис. 2.29. Окно макетной платы отладчика gpsim

Теперь следует воспользоваться клавишей Add library, чтобы получить доступ к дополнительным модулям. И я не сразу понимаю, что именно следует ввести в окошке диалога, пока не повторяю то, что собственно и написано.

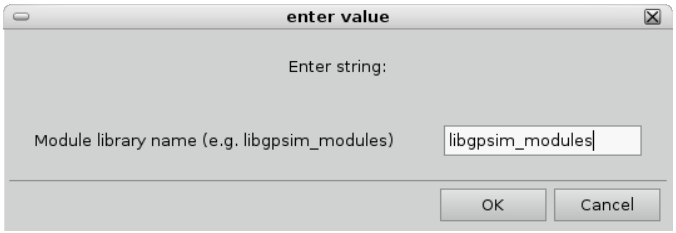


Рис. 2.30. Диалоговое окно добавления библиотеки

Теперь можно воспользоваться клавишей **Add module** для поиска модуля usart.

Рис. 2.31. Диалоговое окно добавления модулей

Двойной щелчок по этому модулю добавляет его на макетную плату и открывает еще одно (консольное) окно с надписью USART. Небольшой опыт работы с программами симуляции подсказывает мне, что новое приобретение в виде модуля usart необходимо подключить к микроконтроллеру. И что для этих целей можно воспользоваться узлами схемы (node). Для добавления узла на макетной плате есть клавиша **Add node**. Я добавлю два узла, которые назову rb1 и rb2. Диалоговое окно, появляющееся после нажатия клавиши добавления узла, похоже на окно диалога добавления библиотеки и требует введения имени узла. Можно проверить, что после нажатия на клавишу ОК, раздел узлов пополняется вновь созданными.

Теперь я хочу соединить узлы и вывода микроконтроллера. Для этого, открыв содержание микроконтроллера щелчком по квадратику со значком плюс слева от имени контроллера на макетной плате, я выбираю portb1, по которому тоже дважды щелкаю. Это действие вызывает появление новой клавиши в нижней части макетной платы с названием **Connect stimulus to node**. Щелчок по этой клавише открывает список всех созданных узлов, а я выбираю узел rb1, по которому дважды щелкаю с помощью клавиши мышки. После этого надпись выше клавиши, которая прежде гласила, что вывод не подключен, меняется на Connected to node rb1. Те же операции я повторяю с выводом portb2 микроконтрллера, который подключаю к узлу rb2.

После подключения контроллера я закрываю дерево его выводов щелчком по квадратику с его названием и значком минус внутри и открываю дерево выводов usart. Меня интересуют выводы usart.RXPIN и usart.TXPIN, которые я подключаю соответсветственно к узлам rb2 и rb1, таким же образом, как сделал это с выводами контроллера (заметьте, что RXPIN оказывает соединен с portb2, а TXPIN с portb1).

В этом месте я споткнулся (а кто не спотыкается). Открывающаяся при добавлении модуля usart консоль USART в моем представлении должна принимать ввод с клавиатуры и отображать то, что отправляет микроконтроллер. До отправки еще далеко, а ввести с клавиатуры я пытался. Если вернуться к коду моей проверочной программы и ее описанию, то можно заметить, что я отправляю символ «1» для зажигания светодиода на выводе RA0, и символ «0» для гашения светодиода. Это теперь так, а первоначально я собирался использовать символы «1» и «2». Почему я все переделал? Я сейчас долго и нудно постараюсь все объяснить. Например, так: НЕ ПОЛУЧИЛОСЬ.

Ввод в консоли команд не заработал. Тогда я решил использовать установку атрибутов usart. Чтобы это сделать достаточно на макетной плате в перечне элементов дважды щелкнуть по нужному элементу. Ниже открывается окно с перечнем атрибутов элементов. И теперь можно двойным щелчком клавиши мышки по нужному атрибуту вывести его в окошко пониже, где можно изменить значение атрибута.

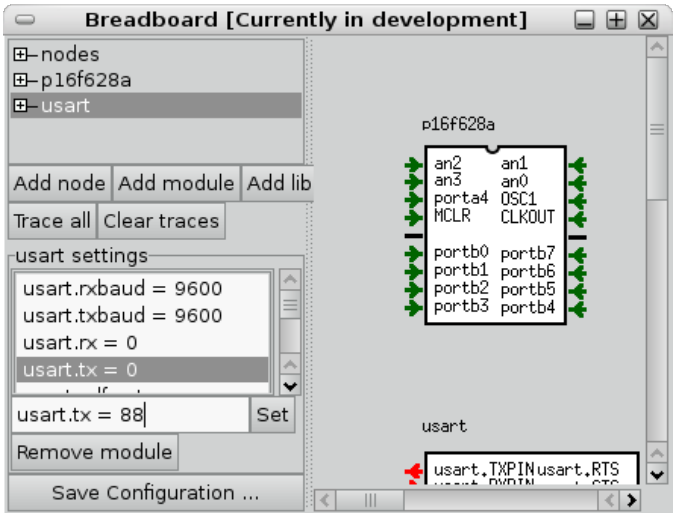


Рис. 2.32. Изменение значение атрибута элемента макетной платы

Я использовал атрибут usart.tx, которому задал значение «88». А наблюдать, как меняется вводимый символ можно с помощью окна наблюдения **Watch Viewer**. Для добавления в окно переменной, я наблюдаю за переменной COMMAND, которая получает значение вводимого символа, для ее добавления к наблюдению следует выделить эту переменную в тексте (окно Source), щелкнуть правой клавишей мышки по выделению, и выпадающем меню выбрать пункт **Add to watch**. Еще хорошо бы установить точку останова в том месте, где ей передается значение из ввода, просто щелкнуть в области значений строк левой клавишей мышки. Повторный щелчок снимает точку останова (breakpoint). Можно в тексте нажать правую клавишу мышки и из выпадающего меню выбрать **Breakpoint here**.

После запуска программы отладки, установив значение атрибута usart.tx, я стал вводить разные варианты для символа «1». И с двойными кавычками, и с одинарными, и без кавычек, и в виде шестнадцатеричного кода символа ASCII, и десятичного – все без положительных результатов, если не считать того, что вводимые десятичные значения порядка 100 меняли переменную COMMAND. Так я пришел к двум числам 88 и 89, которые давали значение символа «0» и символа «1», соответственно. Именно по этой причине я переделал код программы, что было совсем не сложно. Именно так я получил некоторую видимость работы программы, поскольку стал краснеть при вводе числа 89 нулевой вывод порта A, как и было задумано.

Я перебрал множество чисел, пытаясь выявить связь со значениями кодов ASCII, но понял, что это будет очень затяжное занятие и отказался от этого.

Затем, возвращаясь временами к USART консоли, и пытаясь ввести с клавиатуры символ через консоль, я случайно зацепил какую-то букву. Теперь программа отладки упорно стала «виснуть». Мне это не понравилось и я, что значит бессистемная работа, и я стал вновь пытаться ввести что-то полезное в окно консоли USART.

При этих попытках, то есть, активизировать окно и нажать на клавиатуре цифру 1 или 2, я не вижу результата, а нажав буквенную клавишу, я застреваю в функции getch(), выйти из которой мне никак не удается. Последнюю неприятность я пытаюсь исправить добавив в оператор переключения строку выхода, если ни один из случаев команды не совпадает. Теперь оператор выглядит так:

```
switch (COMMAND) {
      case '1': bitset (PORTA, 0);
```

```
      break;
      case '2': bitclr (PORTA, 0);
      break;
      default: break;
}
```

Мне кажется, что оснований где-либо застревать быть не должно. И действительно, это похоже, работает, поскольку при активизации окна консоли USART и вводе с клавиатуры любой буквы я останавливаюсь, но с помощью щелкания по клавишам **over** и **run** на основной панели отладчика мне удается оживить отладку и переместиться к ожиданию очередного ввода.

Однако цифры не работают. В настоящий момент (да и в будущем) для меня не важно, как я обозначу команду. Мне ничто не мешает заменить символ «0» на «a», а символ «1» на «b» в тексте программы, откомпилировать ее заново и попробовать еще раз запустить отладку. Если консоль не работает с цифрами, то с буквами она работает.

Заменим цифры на буквы, и ... И опять ничего. Но! Но теперь я проверяю, какой код я получаю в переменной COMMAND, когда нажимаю букву «a», и когда нажимаю букву «b» (латинские). Код выглядит для меня странно: a = 193, b = 198 десятичные. Именно их я в коде основной программы задаю в переключателе switch, как десятичные числа, эта часть программы принимает вид:

```
switch (COMMAND) {
      case 193: bitset (PORTA, 0);
      break;
      case 198: bitclr (PORTA, 0);
      break;
      default: break;
      }
```

Но зато теперь отладка проходит, а когда я нажимаю клавишу «a» на клавиатуре вывод RA0, долженствующий изображать светодиод, краснеет.

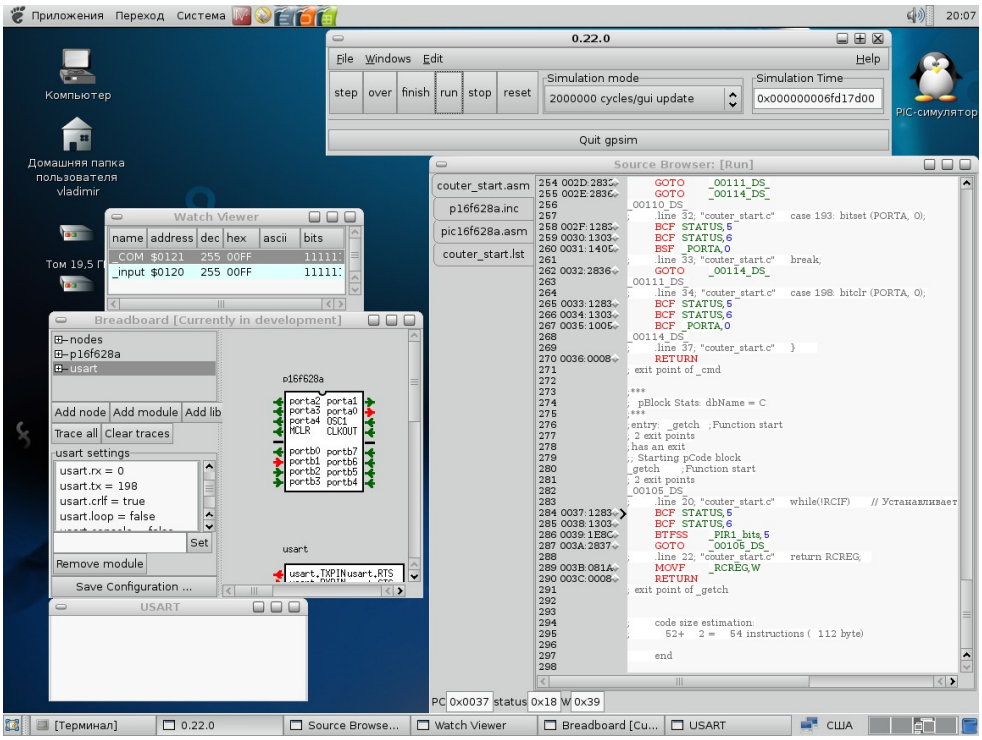


Рис. 2.33. Вид работающей отладочной программы gpsim

И все бы хорошо, когда бы не было плохо. Попытка запустить отладку вторично вновь сталкивает меня с проблемой зависания на функции getch(). Не вижу другого пути, как вернуться к чтению документации, тем более, что документация со времени моего первого знакомства с программой изменилась, а в исходном коде я нахожу папку с примерами. Приостановим-ка бессистемную работу, попробуем разобраться.

Глава 3. У камелька

Если у радиолюбителя есть компьютер, то он пробовал программировать. Аксиома.

Мне нравится словосочетание «у камелька». В нем есть что-то домашнее, уютное. Такое же ощущение должно появиться у вас при работе с Gambas, едва что-то начнет получаться. Но если вам не нравится название главы – зачеркните его, напишите такое, какое вам понравится. Любая книга, особенно техническая, даже если это роман или сказка «в технической прозе», как у меня, это не более, чем рабочий инструмент, мастерок каменщика или топор плотника. А инструмент должен быть удобным, это главное.

После опытов с «железом», что мне привычнее, я хотел бы вернуться к программной части рассказа. Более того, сделать эту часть только программной без питательных (для меня) добавок пайки и щелчков переключателей приборов.

Я не программист, который в течение ряда лет работал бы над проектами, писал коды программ, и которому есть что вспомнить о приключении уравнений или борьбе со злобными процедурами, об удачах в строительстве классов и провалах в фундаменте интерфейсов. Поэтому я хотел воспользоваться опытом более опытных программистов, взяв за основу главы что-нибудь любопытное из нескольких книг, которые я наметил купить. Увы. Я не нашел их в ближайших книжных магазинах, не нашел все требуемые книги в каком-либо интернет-магазине, а затевать путешествие по книжным магазинам города не показалось мне удачной идеей. Да и есть ли в этом смысл, если моя задача рассказать не столько о том, каким увлекательным занятием является программирование, что так и есть, сколько показать, насколько легко можно пользоваться современными средами программирования для своих нужд.

Вспоминая, что выше я упоминал о том, как когда-то, давно-давным, мы с сыном писали программу для самообучения печати «вслепую», я подумал что этот пример будет не хуже любого другого. Любая задача становится интересной только тогда, когда она ваша задача. Если кто-то другой предлагает вам задачу, то ее решение – это уже работа. Мне кажется, что все воспринимают это схожим образом. И если я даже придумаю нечто невероятно интересное (для меня), оно может показаться скучным для остальных.

Я уже много лет печатаю русский текст «вслепую», но умение делать это с латинской раскладкой утратил вскорости после того, как обрел это умение. И все эти годы, если возникает нужда что-то напечатать, например, на английском, я поправляю очки, зависаю над клавиатурой и лихо долблю по клавиатуре двумя пальцами. Мне очень интересно, смогу ли я, пока дописываю эту книгу, если когда-нибудь допишу ее, за две-три недели восстановить умение печать не только кириллицей, но и латиницей?

Начало проекта «Машинистка»

Прежде, чем открыть новый проект в Gambas, мне, так уж я устроен, мне необходимо как-то описать, а что, собственно, я намерен делать.

Я уже рассказывал о тех многочисленных возможностях, что существуют в Linux (практически в любом более или менее полном дистрибутиве) для такого рода работы. Но я хорошо знаю себя – если я начну работать с программой, помогающей мне составить план работы над проектом, я обязательно буду приводить эту работу начального этапа в картинках, описывающих эту работу. Не то, что я не могу описать это словами, такая задача мне кажется даже более привлекательной, чем прямая вставка изображения, но я почти уверен, как бы хорошо ни описал я рисунок, вы, читая описание, увидите совсем другое «кино». На этом,

видимо, строится вся литература. Но это может быть причиной ошибок в других видах деятельности, где увидеть, много полезнее, чем прочитать словесное описание. Это можно отнести к схемам, уравнениям или кодам программы, или к результатам работы программы, когда вид диалогового окна позволяет вам сразу понять, что нужно сделать, а описание этого действия, тщательное и занимательное, заставляет сделать множество ошибок из-за различного восприятия фразы разными людьми.

Но самым простым способом описать задачу остается лист бумаги и карандаш, или текстовый процессор и клавиатура. Им я и воспользуюсь.

Что представляет собой процесс печати? Не будучи специалистом в этом вопросе, я воспользуюсь теми знаниями, что некогда получил при чтении книги «Осваиваем микрокомпьютер». Возможно, существует множество методик, но я готов использовать только ту, что использовал сам. Итак.

Основой этой методики является принцип обслуживания определенных клавиш клавиатуры пальцами ваших двух рук. За каждым пальцем закреплено несколько клавиш. В начальный момент восемь пальцев двух рук находятся на среднем ряду клавиатуры так, чтобы указательный палец левой руки был на латинской букве F, а указательный палец правой руки на букве J. Клавиатура имеет выступы на этих буквах, которые позволяют на ощупь легко расположить пальцы в начальный момент.

Второй принцип метода требует, чтобы после нажатия на любую клавишу все восемь пальцев возвращались в начальное положение. То есть печать выглядит так, что вы из начального положения печатаете любую букву соответствующим пальцем, а затем сразу возвращаетесь в начальное положение.

И, наконец, это уже не принцип метода, а мой личный опыт. Не пытайтесь запомнить, где на клавиатуре находится та или иная буква. Даже если вы это делаете неосознанно, следите за этим, чтобы пресечь все попытки «заучить» клавиатуру. Пусть за вас это делают ваши пальцы – это их епархия, это их работа, это их «мозги» должны напрягаться. И это, похоже, важный момент. Когда я сам начинал учиться печатать, я понял, насколько это мешает мне быстро освоить печать.

Таким образом, теперь уже о программе, я предполагаю нарисовать клавиатуру, раскрасив разными цветами рабочее поле для каждого пальца. При печати эти раскрашенные клавиши будут менять цвет, когда нажата соответствующая клавиша.

Далее, над клавиатурой будет располагаться поле для вывода печатаемого текста, как лист бумаги, выступающий из пишущей машинки. После заполнения строки, это поле должно удлиняться, как если бы вы нажали перевод каретки (если это так называется у пишущей машинки).

Последнее, что мне понадобится для программы, это поле, где будут выводиться слова, которые следует напечатать. Дело в том, что методика предполагает поэтапное использование в начале обучения только основного ряда клавиш, на котором размещаются ваши руки в начальный момент, затем к этому ряду подключается использование ряда, расположенного над ним, затем под ним, и лишь позже вам будут предложены слова, использующие все три ряда клавиш. Пробел удобно нажимать большими пальцами рук, как удобнее в тот или иной момент. При необходимости печати больших букв можно использовать свободную руку для того, чтобы нажать клавишу **Shift**.

Пока я не готов составить полное описание программы, но мне ясно, с чего следует начать. А может, признаюсь, сказывается мой скверный характер – мне интереснее начать работать с программой, чем продумывать план работы. К плану, я уверен, придется

вернуться, но сейчас я запущу Gamabs и создам новый графический проект.

Из предлагаемых при запуске вариантов я выбираю графический Qt проект. Имя проекту я даю Typist (машинистка). После двойного щелчка по FMain в менеджере проекта (левое окно) появляется основная форма программы. Я использую версию, работа над которой еще не завершена, а в стабильной версии форма, если я не ошибаюсь, появляется сразу. Но это не важно. Для того, чтобы нарисовать клавиши клавиатуры, я использую объекты TextLabel. В поле свойств Text (меню свойств объекта) я буду вводить буквы для клавиш, использую свойство Alignment для задания Center, чтобы буква располагалась в центре клавиши, и использую свойство Border для создания контура (я выбираю вариант Raised). Теперь я могу задать исходный цвет клавиши с помощью свойства Background, что важно для меня, поскольку мне нужно разметить «сферу» деятельности для каждого из восьми пальцев. Выбрав темно-зеленый цвет (на закладке Free), я вижу, что черные буквы на этом фоне, практически, не видны. По этой причине я меняю цвет Foreground для каждого из трех объектов TextLabel на белый. Несколько минут ушло у меня на первые шаги по построению проекта. Если его откомпилировать и запустить, то выглядеть это будет так:

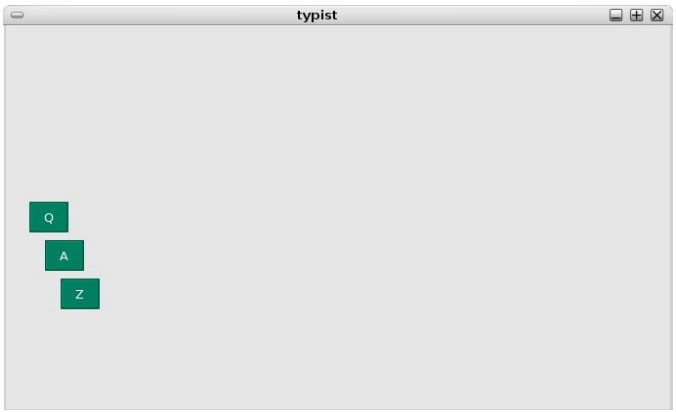


Рис. 3.1. Первые шаги в проекте «Машинистка»

Чтобы упростить работу я предполагаю скопировать это построение и повторить его еще девять раз. Должна получиться полная клавитура без клавиши пробела. Затем я изменю исходные цвета и добавлю клавишу пробела.

Для копирования я, удерживая левую клавишу мышки, провожу курсором прямоугольник, захватывающий все три клавиши, а после отпускания клавиши мышки они оказываются выделены. Насколько могу судить, операция совершенно одинаковая во многих приложениях и при разных операционных системах. Теперь, нажав правую клавишу мышки после размещения курсора на одном из выделенных объектов (и это довольно типично), из выпадающего меню я выбираю пункт **Copy**, а, щелкнув правой клавишей мышки на свободном месте формы, из выпадающего меню выбираю **Paste**. Мне остается подхватить мышкой полученные три клавиши и перенести их чуть правее уже готовых. Я пока не меняю ни букв внутри клавиш, ни их цвет. После того, как клавиатура готова, я меняю цвет и названия клавиш. Поскольку в работе принимает участие по четыре пальца каждой руки, я использую четыре цвета – темно-зеленый, синий, желтый и красный. Для второй половины клавиатуры эти цвета повторяются в обратном порядке. Цвет букв на желтом фоне приходится вернуть к черному, иначе их совсем не видно, и я разделяю клавиатуру для левой руки и правой больше, чем это имеет место в действительности, чтобы явно обозначить «сферы влияния». Пока у меня получается следующий вид программы.

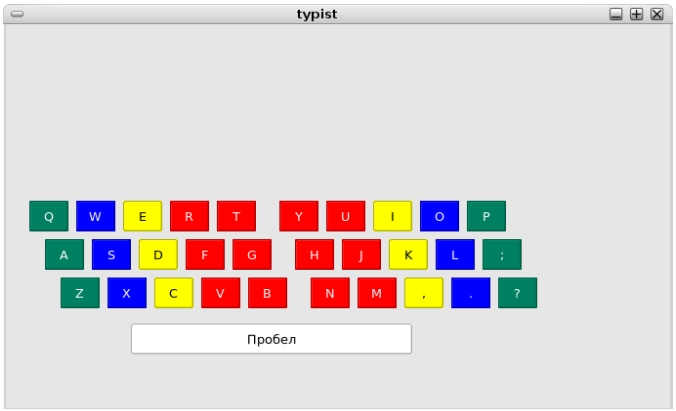


Рис. 3.2. Рисунок клавиатуры в проекте «Машинистка»

Как видно из рисунка, каждый палец обслуживает три клавиши, кроме указательных, у которых вдвое больше работы. Пробел, напомню, можно нажимать большими пальцами, каким в данный момент удобнее.

К рисунку я добавлю TextBox, объект изображающий лист бумаги, а чуть правее еще одну TextLabel, где должны появляться слова, которые следует напечатать. Рисунок приобретает почти окончательный вид, хотя прошло не более получаса с момента начала работы (с учетом перерывов в работе).

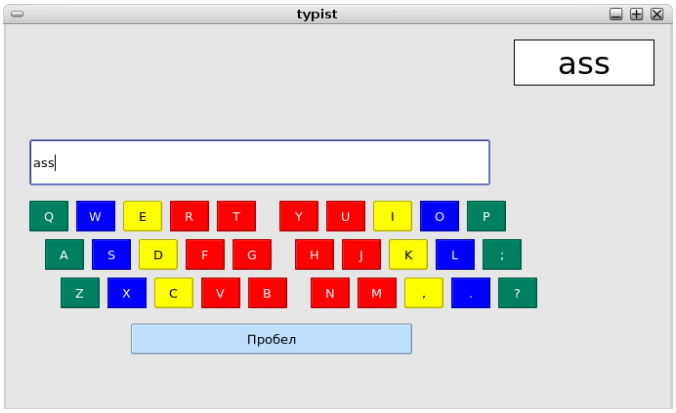


Рис. 3.3. Почти окончательный вид рисунка

Если программу запустить, то на листе бумаги уже можно начинать печатать.

Теперь мне хотелось бы, хотя это не обязательно, отображать нажатые клавиши изменением цвета. Задача для программиста, как мне кажется, совсем простая. Но не для меня. Попробую подойти к ее решению, сделав предположение, что среди свойств объектов, размещенных мной на форме есть те, что помогут мне выполнить задуманное. С этой целью я нажму правой клавишей мышки на объекте TextBox, выберу раздел **Event** выпадающего меню и выберу пункт **KeyPress**. Теперь в коде программы появляется шаблон подпрограммы отвечающий этому событию. Поскольку я не знаю, прав ли я в своем предположении, я проведу опыт, добавив в шаблон простую строку. На рисунке клавиатуры клавиша «Q» изображена с помощью **TextLabel**. Если я ввожу это имя в редакторе кода, то появляется список всех этикеток, и мне достаточно выбрать **TextLabel1** для моего эксперимента. После

того, как я ставлю точку после выбранного объекта открывается меню выбора его свойств, где я могу выбрать свойства фона. А значение этого свойства я просто скопирую с соответствующего у клавиши пробела. Теперь код программы выглядит следующим образом:

```
PUBLIC SUB TextBox1_KeyPress()

      TextLabel1.Background = &HBFDFFF&

END
```

В итоге после запуска программы при нажатии на любую клавишу цвет клавиши «Q» меняется. И остается таким, каким стал. А это не совсем то, что мне нужно. Я хотел бы изменить эту часть кода примерно так:

```
PUBLIC SUB TextBox1_KeyPress()

      IF Key.Code = 113 THEN TextLabel1.Background = &HBFDFFF&

END
```

Значение 113 – это десятичное значение кода ASCII символа «q». Проверив этот вариант, я убедился, что он не работает. Обращение к руководству (help) еще больше огорчило меня. В руководстве явно не советуют использовать такую форму работы с символами. Хорошо бы сейчас посоветоваться с опытным программистом, но такой возможности у меня нет. Придется попытаться обойти эту проблему. И сделать это я хочу с помощью следующего кода:

```
PUBLIC SUB TextBox1_KeyPress()

      TextLabel32.Text = Str$(Key.Code)

END
```

При нажатии любой клавиши в окошко, которое я предназначил для появления слов-заданий, у меня оно имеет имя TextLabel32, в это окошко будут отправляться в виде строки коды символов нажатой клавиши. Мне останется только проверить их работоспособность в предыдущем коде. Первый эксперимент с выводом кода дает результат:

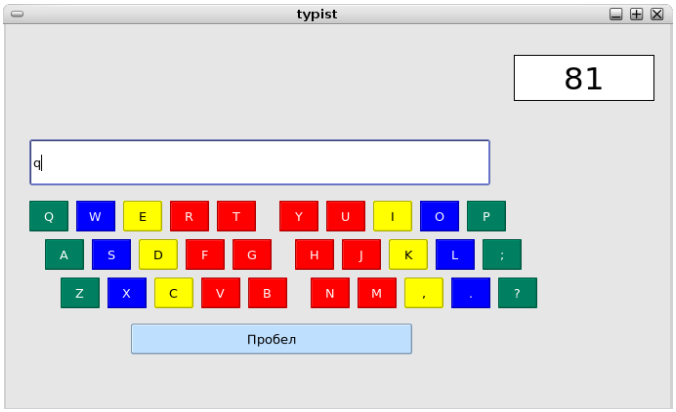


Рис. 3.4. Определение кода нажатой клавиши

Совет, насколько я понимаю, был правилен. Это похоже на десятичное значение заглавной

«Q», но что я получаю в действительности? Интересно было бы разобраться, но чуть позже. Сейчас я попробую проверить, что у меня получится, если я в код программы добавлю событием отпускание нажатой клавиши (с помощью раздела **Event** выпадающего меню и пункта **KeyRelease** при щелчке правой клавишей мышки на **TextBox**), куда добавлю немного текста:

```
PUBLIC SUB TextBox1_KeyPress()
      IF Key.Code = 81 THEN TextLabel1.Background = &HBFDFFF&
END

PUBLIC SUB TextBox1_KeyRelease()
      IF Key.Code = 81 THEN TextLabel1.Background = &H008060&
END
```

В таком виде клавиша действительно меняет цвет, если это клавиша «q», при нажатии, возвращаясь к первоначальному при отпускании клавиши. Цвет для возвращения я в коде программы добавил, просто копируя его значение из меню свойств клавиши.

Я не думаю, что это хороший стиль программирования, но я не стилист, не программист, по причине чего использую то, что нашел. Я могу проверить коды всех нужных мне клавиш, дописать аналогичный текст для всех клавиш графической клавиатуры, и проверить, все ли они правильно работают. Пока мне этого должно хватить. Тем более, что полученные коды совпадают с десятичными значениями заглавных букв кода ASCII, откуда их можно взять. Текст программы в этой части (а это пока все, что есть) приобретает вид:

```
' Gambas class file

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()

END

PUBLIC SUB TextBox1_KeyPress()
      IF Key.Code = 81 THEN TextLabel1.Background = &HBFDFFF& 'q
      IF Key.Code = 65 THEN TextLabel2.Background = &HBFDFFF& 'a
      IF Key.Code = 90 THEN TextLabel3.Background = &HBFDFFF& 'z
      IF Key.Code = 87 THEN TextLabel6.Background = &HBFDFFF& 'w
      IF Key.Code = 83 THEN TextLabel5.Background = &HBFDFFF& 's
      IF Key.Code = 88 THEN TextLabel4.Background = &HBFDFFF& 'x
      IF Key.Code = 69 THEN TextLabel9.Background = &HBFDFFF& 'e
      IF Key.Code = 68 THEN TextLabel8.Background = &HBFDFFF& 'd
      IF Key.Code = 67 THEN TextLabel7.Background = &HBFDFFF& 'c
      IF Key.Code = 82 THEN TextLabel12.Background = &HBFDFFF& 'r
      IF Key.Code = 70 THEN TextLabel11.Background = &HBFDFFF& 'f
      IF Key.Code = 86 THEN TextLabel10.Background = &HBFDFFF& 'v
      IF Key.Code = 84 THEN TextLabel15.Background = &HBFDFFF& 't
      IF Key.Code = 71 THEN TextLabel14.Background = &HBFDFFF& 'g
      IF Key.Code = 66 THEN TextLabel13.Background = &HBFDFFF& 'b
      IF Key.Code = 89 THEN TextLabel18.Background = &HBFDFFF& 'y
      IF Key.Code = 72 THEN TextLabel17.Background = &HBFDFFF& 'h
      IF Key.Code = 78 THEN TextLabel16.Background = &HBFDFFF& 'n
      IF Key.Code = 85 THEN TextLabel21.Background = &HBFDFFF& 'u
      IF Key.Code = 74 THEN TextLabel20.Background = &HBFDFFF& 'j
      IF Key.Code = 77 THEN TextLabel19.Background = &HBFDFFF& 'm
      IF Key.Code = 73 THEN TextLabel24.Background = &HBFDFFF& 'i
```

```
      IF Key.Code = 75 THEN TextLabel23.Background = &HBFDFFF& 'k
      IF Key.Code = 44 THEN TextLabel22.Background = &HBFDFFF& ',
      IF Key.Code = 79 THEN TextLabel27.Background = &HBFDFFF& 'o
      IF Key.Code = 76 THEN TextLabel26.Background = &HBFDFFF& 'l
      IF Key.Code = 46 THEN TextLabel25.Background = &HBFDFFF& '.
      IF Key.Code = 80 THEN TextLabel30.Background = &HBFDFFF& 'p
      IF Key.Code = 59 THEN TextLabel29.Background = &HBFDFFF& ';
      IF Key.Code = 47 THEN TextLabel28.Background = &HBFDFFF& '/
      IF Key.Code = 32 THEN TextLabel31.Background = &HFFFF00& 'Пробел
END

PUBLIC SUB TextBox1_KeyRelease()
      IF Key.Code = 81 THEN TextLabel1.Background = &H008060& 'q
      IF Key.Code = 65 THEN TextLabel2.Background = &H008060& 'a
      IF Key.Code = 90 THEN TextLabel3.Background = &H008060& 'z
      IF Key.Code = 87 THEN TextLabel6.Background = &H0000FF& 'w
      IF Key.Code = 83 THEN TextLabel5.Background = &H0000FF& 's
      IF Key.Code = 88 THEN TextLabel4.Background = &H0000FF& 'x
      IF Key.Code = 69 THEN TextLabel9.Background = &HFFFF00& 'e
      IF Key.Code = 68 THEN TextLabel8.Background = &HFFFF00& 'd
      IF Key.Code = 67 THEN TextLabel7.Background = &HFFFF00& 'c
      IF Key.Code = 82 THEN TextLabel12.Background = &HFF0000& 'r
      IF Key.Code = 70 THEN TextLabel11.Background = &HFF0000& 'f
      IF Key.Code = 86 THEN TextLabel10.Background = &HFF0000& 'v
      IF Key.Code = 84 THEN TextLabel15.Background = &HFF0000& 't
      IF Key.Code = 71 THEN TextLabel14.Background = &HFF0000& 'g
      IF Key.Code = 66 THEN TextLabel13.Background = &HFF0000& 'b
      IF Key.Code = 89 THEN TextLabel18.Background = &HFF0000& 'y
      IF Key.Code = 72 THEN TextLabel17.Background = &HFF0000& 'h
      IF Key.Code = 78 THEN TextLabel16.Background = &HFF0000& 'n
      IF Key.Code = 85 THEN TextLabel21.Background = &HFF0000& 'u
      IF Key.Code = 74 THEN TextLabel20.Background = &HFF0000& 'j
      IF Key.Code = 77 THEN TextLabel19.Background = &HFF0000& 'm
      IF Key.Code = 73 THEN TextLabel24.Background = &HFFFF00& 'i
      IF Key.Code = 75 THEN TextLabel23.Background = &HFFFF00& 'k
      IF Key.Code = 44 THEN TextLabel22.Background = &HFFFF00& ',
      IF Key.Code = 79 THEN TextLabel27.Background = &H0000FF& 'o
      IF Key.Code = 76 THEN TextLabel26.Background = &H0000FF& 'l
      IF Key.Code = 46 THEN TextLabel25.Background = &H0000FF& '.
      IF Key.Code = 80 THEN TextLabel30.Background = &H008060& 'p
      IF Key.Code = 59 THEN TextLabel29.Background = &H008060& ';
      IF Key.Code = 47 THEN TextLabel28.Background = &H008060& '/
      IF Key.Code = 32 THEN TextLabel31.Background = &HBFDFFF& 'Пробел
END
```

Добавляя часть кода я проверяю, работает ли она, поправляя «на ходу» ошибки, появляющиеся даже при копировании. Интересно было бы посмотреть, как эту часть программы могут кодировать программисты. Можно было бы поискать тексты в Интернете, но это тоже позже, поскольку есть и более интересные «неполадки». Но это все позже. А сейчас меня больше заботит другое – текст, который я ввожу, отображается строкой в **TextBox**. А мне хотелось бы, чтобы при нажатии ввода текст перемещался на следующую строку. Попробовав варианты с добавлением символа перевода строки и возврата каретки, я прихожу к выводу, что есть простое решение – заменить **TextBox** на **TextArea**. Исправив в тексте кода имя объекта, я получаю все, что мне хотелось бы. При нажатии на ввод текст смещается вверх, а курсор оказывается на новой строке. Это оказывается гораздо проще, чем мои попытки проделать это с **TextBox**.

Теперь мне хотелось бы решить следующую задачу. После нажатия на пробел слово-задание должно меняться. Но в первую очередь эти слова следует где-то разместить. Самый

простой вариант – создать строковый массив.

После нескольких неудачных попыток мне удается разместить объявление массива (**TestWords[]**) в самом начале текста кода и инициализировать массив первыми тремя словами. А с помощью конструкции IF...THEN заставить как-то изменяться слова в поле **TextLabel32** (где и положено появляться словам-заданиям). Для этого я добавляю к объявлению массива еще и индексную переменную «I». Нечто в роде нижеследующего подает признаки жизни.

```
' Gambas class file
PUBLIC TestWords AS String[] = ["ass", "add", "fad"]
PUBLIC I AS Integer = 0

PUBLIC SUB TextArea1_KeyRelease()
      ......................
      ......................
      IF Key.Code = 32 THEN
            TextLabel31.Background = &HFFFF00&
            TextLabel32.Text = TestWords[I]
            I = I + 1
      ENDIF 'Пробел
END
```

Развиваем успех

Но это, конечно, не все. Слов для работы на среднем ряду клавиатуры у меня 13. Хотелось бы, чтобы эти слова появлялись в окне задания непрерывно, то есть, после появления последнего слова вновь появлялось первое. И еще одно. Идея самообучения печати вслепую в том, чтобы на первом этапе обучения использовать только буквы среднего ряда клавиатуры, где размещаются пальцы в начальный момент, и куда они должны возвращаться после каждого нажатия любой другой клавиши. Во всяком случае, это касается той методики, которую я пытаюсь освоить. А это значит, что существует второй этап обучения, когда слова должны содержать буквы среднего и верхнего ряда, затем третий этап, и четвертый этап. То есть, мне понадобится несколько массивов со словами-заданиями.

В данный момент, прав я или нет, я предполагаю добавить нужное количество текстовых массивов со словами заданиями (назову их Less1[]...Less4[]). Я сохраню текстовый массив TestWords[], как рабочий массив. А для переключения между заданиями использую основное меню, которое можно создать, если щелкнуть правой клавишей мышки в свободном месте формы и выбрать пункт **Menu editor...** в выпадающем меню. Открывается диалог создания меню, где можно, нажав клавишу **Insert**, ввести текст пунктов меню (**Caption**), и с помощью стрелок управления разместить вводимые пункты меню, как подменю основного меню.

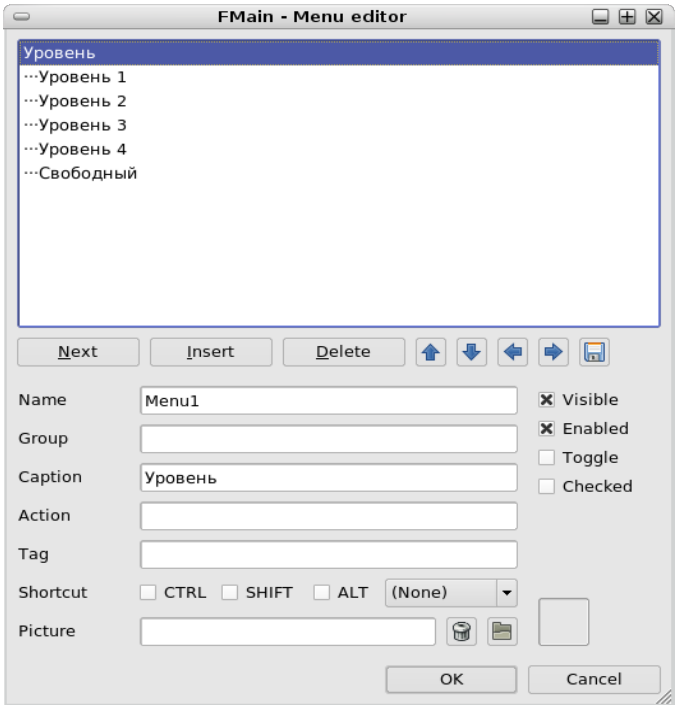


Рис. 3.5. Диалоговое окно создания основного меню в Gambas

Редактор позволяет сделать не только это, но это сейчас для меня самое нужное. Поскольку, закончив создание этого меню, я могу открыть его на форме, выбрать раздел подменю «Уровень 1» с помощью курсора мышки, и щелкнуть по нему, получив подпрограмму, в которой сделаю присваивание содержимого первого текстового массива моему рабочему массиву:

```
PUBLIC SUB Menu2_Click()
      TestWords = Less1
      TextLabel32.Text = TestWords[I]
END
```

Здесь же я вывожу первое слово-задание в поле **TextLabel32**. Работа с программой должна выглядеть следующим образом. После запуска программы следует выбрать нужный уровень, а после появления первого слова задания можно приступать к работе. Если для первого уровня я списал слова-задания из книги, добавив два-три своих, то для второго и последующих уровней слова придется придумать. Начало работы с программой выглядит так:

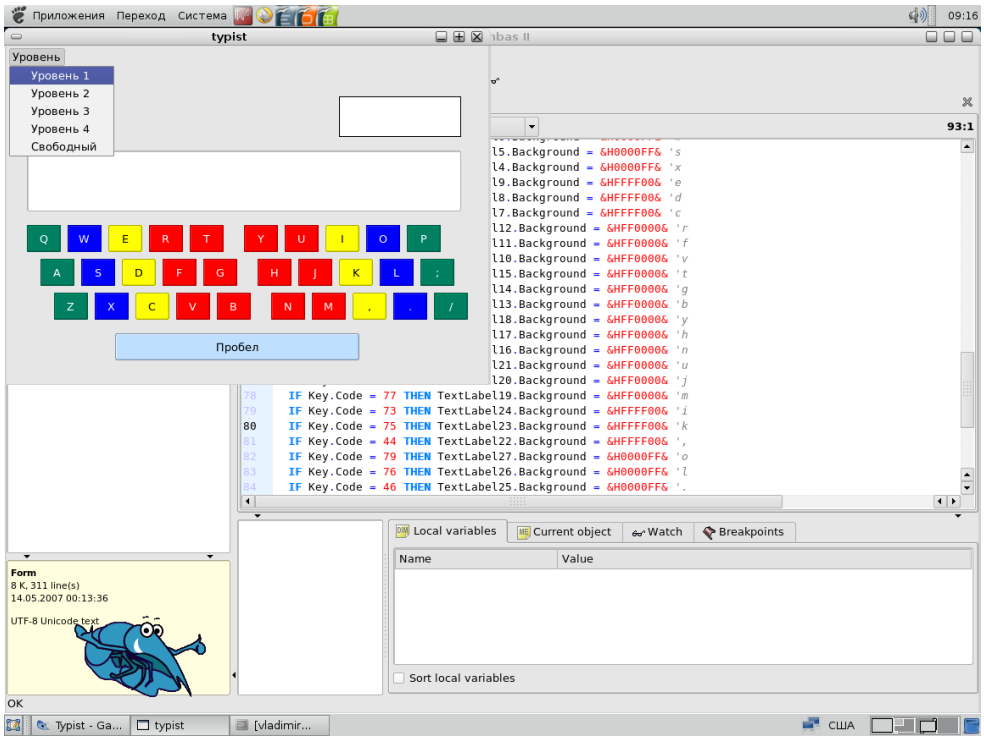


Рис. 3.6. Начало работы с программой

Я не знаю, есть ли какие-либо закономерности в словах-заданиях в той методике, которую пытаюсь реализовать, и нужно ли, чтобы все слова были осмысленными, или лучше, если они будут бессмысленными? Я попробую видоизменить то, что есть, в той мере, в какой моя фантазия не исчерпает моего терпения.

Слова первого уровня:

ass, add, fad, all, lass, falls, lads, dad, alas, ask, salad, hag, jak, jagh.
Из них создадим:
wass, qadd, fat, yell, kugg, talihg, lords, pad, yellow, krast, gatapul, jegos, tajer, troll.
Для третьего уровня:
azax, valan, lams, xaj, fan, sabac, malz, cagh, jana, balls, mama, xanas, vagaz, bags.
И для четвертого уровня я опять воспользуюсь книгой:
fox, box, soks, hnoks, ring, table, tan, whale, mate, rain, sent, tank, scary, plum.

После добавления кода путем копирования, в основном, программа приобретает вид:

```
'' Gambas class file

PUBLIC TestWords AS String[13]

PUBLIC Less1 AS String[] = ["ass", "add", "fad", "all", "lass", "falls", "lads",
"dad", "alas", "ask", "salad", "hag", "jak", "jagh"]

PUBLIC Less2 AS String[] = ["wass", "qadd", "fat", "yell", "kugg", "talihg",
"lords", "pad", "yellow", "krast", "gatapul", "jegos", "tajer", "troll"]

PUBLIC Less3 AS String[] = ["azax", "valan", "lams", "xaj", "fan", "sabac",
"malz", "cagh", "jana", "balls", "mama", "xanas", "vagaz", "bags"]
```

```
PUBLIC Less4 AS String[] = ["fox", "box", "soks", "hnoks", "ring", "table",
"tan", "whale", "mate", "rain", "sent", "tank", "scary", "plum"]

PUBLIC I AS Integer = 0

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()

END

PUBLIC SUB TextArea1_KeyPress()
      IF Key.Code = 81 THEN TextLabel1.Background = &HBFDFFF& 'q
      IF Key.Code = 65 THEN TextLabel2.Background = &HBFDFFF& 'a
      IF Key.Code = 90 THEN TextLabel3.Background = &HBFDFFF& 'z
      IF Key.Code = 87 THEN TextLabel6.Background = &HBFDFFF& 'w
      IF Key.Code = 83 THEN TextLabel5.Background = &HBFDFFF& 's
      IF Key.Code = 88 THEN TextLabel4.Background = &HBFDFFF& 'x
      IF Key.Code = 69 THEN TextLabel9.Background = &HBFDFFF& 'e
      IF Key.Code = 68 THEN TextLabel8.Background = &HBFDFFF& 'd
      IF Key.Code = 67 THEN TextLabel7.Background = &HBFDFFF& 'c
      IF Key.Code = 82 THEN TextLabel12.Background = &HBFDFFF& 'r
      IF Key.Code = 70 THEN TextLabel11.Background = &HBFDFFF& 'f
      IF Key.Code = 86 THEN TextLabel10.Background = &HBFDFFF& 'v
      IF Key.Code = 84 THEN TextLabel15.Background = &HBFDFFF& 't
      IF Key.Code = 71 THEN TextLabel14.Background = &HBFDFFF& 'g
      IF Key.Code = 66 THEN TextLabel13.Background = &HBFDFFF& 'b
      IF Key.Code = 89 THEN TextLabel18.Background = &HBFDFFF& 'y
      IF Key.Code = 72 THEN TextLabel17.Background = &HBFDFFF& 'h
      IF Key.Code = 78 THEN TextLabel16.Background = &HBFDFFF& 'n
      IF Key.Code = 85 THEN TextLabel21.Background = &HBFDFFF& 'u
      IF Key.Code = 74 THEN TextLabel20.Background = &HBFDFFF& 'j
      IF Key.Code = 77 THEN TextLabel19.Background = &HBFDFFF& 'm
      IF Key.Code = 73 THEN TextLabel24.Background = &HBFDFFF& 'i
      IF Key.Code = 75 THEN TextLabel23.Background = &HBFDFFF& 'k
      IF Key.Code = 44 THEN TextLabel22.Background = &HBFDFFF& ',
      IF Key.Code = 79 THEN TextLabel27.Background = &HBFDFFF& 'o
      IF Key.Code = 76 THEN TextLabel26.Background = &HBFDFFF& 'l
      IF Key.Code = 46 THEN TextLabel25.Background = &HBFDFFF& '.
      IF Key.Code = 80 THEN TextLabel30.Background = &HBFDFFF& 'p
      IF Key.Code = 59 THEN TextLabel29.Background = &HBFDFFF& ';
      IF Key.Code = 47 THEN TextLabel28.Background = &HBFDFFF& '/
      IF Key.Code = 32 AND IF I < 13 THEN
            I = I + 1
            TextLabel31.Background = &HFFFF00&
            TextLabel32.Text = TestWords[I]
      ELSE IF Key.Code = 32 AND IF I = 13 THEN
            I = 0
            TextLabel31.Background = &HFFFF00&
            TextLabel32.Text = TestWords[I]
      ENDIF 'Пробел
END

PUBLIC SUB TextArea1_KeyRelease()
      IF Key.Code = 81 THEN TextLabel1.Background = &H008060& 'q
      IF Key.Code = 65 THEN TextLabel2.Background = &H008060& 'a
      IF Key.Code = 90 THEN TextLabel3.Background = &H008060& 'z
      IF Key.Code = 87 THEN TextLabel6.Background = &H0000FF& 'w
      IF Key.Code = 83 THEN TextLabel5.Background = &H0000FF& 's
```

```
      IF Key.Code = 88 THEN TextLabel4.Background = &H0000FF& 'x
      IF Key.Code = 69 THEN TextLabel9.Background = &HFFFF00& 'e
      IF Key.Code = 68 THEN TextLabel8.Background = &HFFFF00& 'd
      IF Key.Code = 67 THEN TextLabel7.Background = &HFFFF00& 'c
      IF Key.Code = 82 THEN TextLabel12.Background = &HFF0000& 'r
      IF Key.Code = 70 THEN TextLabel11.Background = &HFF0000& 'f
      IF Key.Code = 86 THEN TextLabel10.Background = &HFF0000& 'v
      IF Key.Code = 84 THEN TextLabel15.Background = &HFF0000& 't
      IF Key.Code = 71 THEN TextLabel14.Background = &HFF0000& 'g
      IF Key.Code = 66 THEN TextLabel13.Background = &HFF0000& 'b
      IF Key.Code = 89 THEN TextLabel18.Background = &HFF0000& 'y
      IF Key.Code = 72 THEN TextLabel17.Background = &HFF0000& 'h
      IF Key.Code = 78 THEN TextLabel16.Background = &HFF0000& 'n
      IF Key.Code = 85 THEN TextLabel21.Background = &HFF0000& 'u
      IF Key.Code = 74 THEN TextLabel20.Background = &HFF0000& 'j
      IF Key.Code = 77 THEN TextLabel19.Background = &HFF0000& 'm
      IF Key.Code = 73 THEN TextLabel24.Background = &HFFFF00& 'i
      IF Key.Code = 75 THEN TextLabel23.Background = &HFFFF00& 'k
      IF Key.Code = 44 THEN TextLabel22.Background = &HFFFF00& ',
      IF Key.Code = 79 THEN TextLabel27.Background = &H0000FF& 'o
      IF Key.Code = 76 THEN TextLabel26.Background = &H0000FF& 'l
      IF Key.Code = 46 THEN TextLabel25.Background = &H0000FF& '.
      IF Key.Code = 80 THEN TextLabel30.Background = &H008060& 'p
      IF Key.Code = 59 THEN TextLabel29.Background = &H008060& ';
      IF Key.Code = 47 THEN TextLabel28.Background = &H008060& '/
      IF Key.Code = 32 THEN TextLabel31.Background = &HBFDFFF& 'Пробел
END


PUBLIC SUB Menu2_Click()
      TestWords = Less1
      TextLabel32.Text = TestWords[I]
END

PUBLIC SUB Menu3_Click()
      TestWords = Less2
      TextLabel32.Text = TestWords[I]
END

PUBLIC SUB Menu4_Click()
      TestWords = Less3
      TextLabel32.Text = TestWords[I]
END

PUBLIC SUB Menu5_Click()
      TestWords = Less4
      TextLabel32.Text = TestWords[I]
END
```

Программу можно запустить и начать в ней работать.

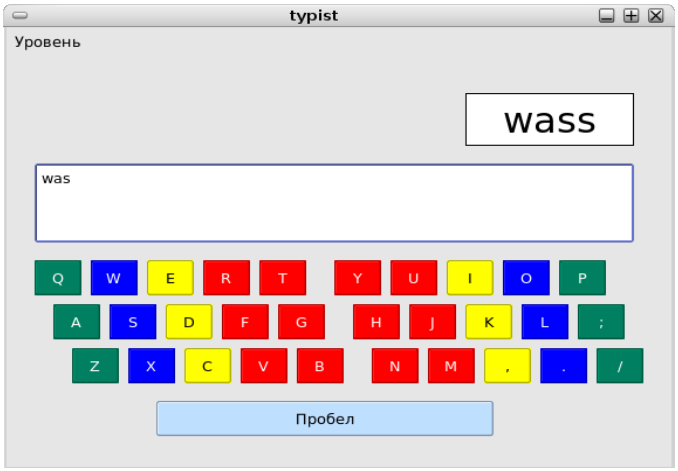


Рис. 3.7. Работающая завершенная программа проекта «Машинистка»

Последнее, что я сделаю, это создам исполняемый файл. Для этого в разделе основного меню **Project** я выберу пункт **Make executable...**, где можно указать папку для установки программы (я ее установлю в домашнюю папку) и можно выбрать опцию создания ярлычка на рабочем столе. Картинку к этому ярлычку (справа с именем typist) я добавил позже.

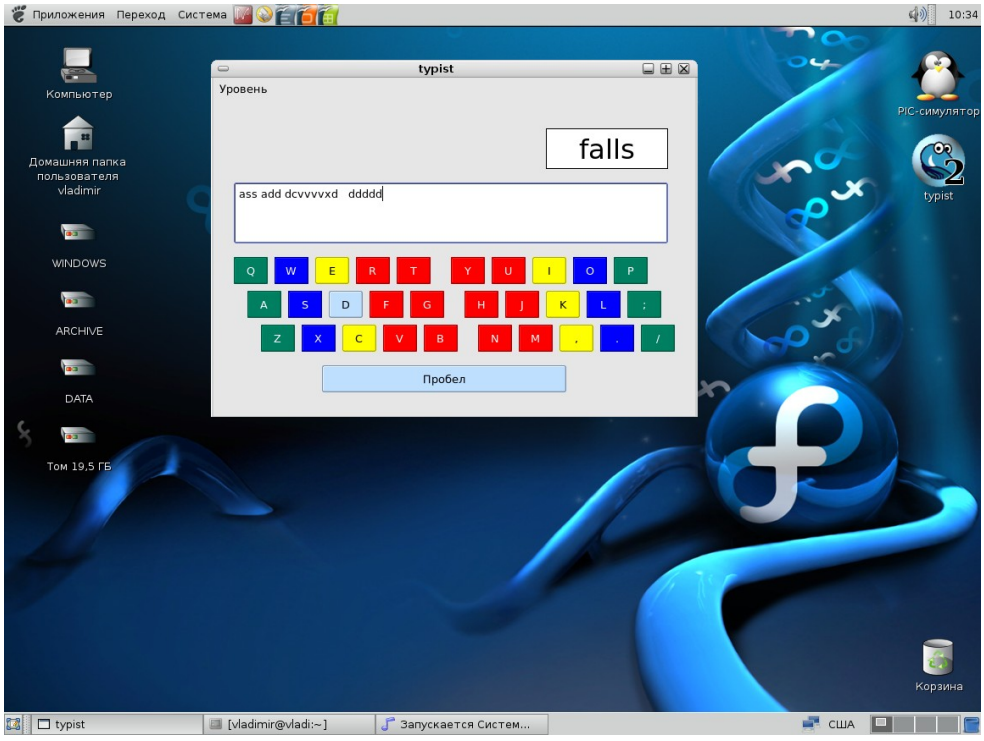


Рис. 3.8. Запуск программы как исполняемой

Она запускается и работает, как любая другая программа. Немного поразмыслив, я решил добавить в основное меню пункт «Выход», что добавило код:

```
PUBLIC SUB Menu7_Click()
      FMain.Close
END
```

И на этом завершить работу.

Заметки и пометки «на память»

Для начала я хочу сказать, что изменение цвета при нажатии клавиши процедура не обязательная, возможно, это даже будет мешать. Если эту процедуру убрать, то код программы уместится на «четвертушке» бумаги. Для опытного программиста – это некоторое количество минут работы. Для меня, положим, день. Много ли это, чтобы создать работающую полезную программу? Как вы думаете?

По мере создания кода, в основном копированием, я использовал только подсказку по языку, да некоторые опыты по выяснению того, что у меня происходит. Вместе с тем, система программирования Gambas оснащена отладчиком (которым я пока не умею пользоваться), как и положено любой системе программирования.

Проблемы с программой, насколько я понимаю, у меня остались. Например, хотя я и планировал создать программу, помогающую освоить печать вслепую для латинской раскладки клавиатуры, от программы было бы больше пользы, если дополнительно можно было выбрать кириллицу. Чтобы научиться печать на родном языке. Но.

Но тогда мне точно пришлось бы отказаться от «мигающих» клавиш, поскольку я не получал кодов, когда проверял их. Причина, как мне кажется, в использовании UTF-8 и системных настроек. Это можно было бы исследовать дополнительно, но любые исследования влекут появление новых интересных проблем, исследование которых... и т.д. Когда-то надо остановиться, и чем раньше, тем лучше.

Текст слов-заданий можно было бы разместить в произвольном текстовом файле, чтобы считывать его при необходимости. Но этот файл был бы еще одним файлом, без которого программа не смогла бы работать, а на моем компьютере и без этого трудно разобраться. Я решил ограничиться строковыми массивами.

Если вы захотите повторить программу и научиться печатать вслепую, я приведу несколько советов из книги, откуда взята методика:

«Что нужно помнить?

Представьте себе, что основные клавиши, как магниты притягивают ваши пальцы в основную позицию.

Чтобы вспомнить, где находится нужная клавиша и каким пальцем ее нужно нажать, смотрите на схему клавиатуры (на экране), а не на саму клавиатуру».

И добавлю от себя:

Не пытайтесь запомнить расположение клавиш на клавиатуре, это только повредит вам. То есть, не думайте о том, где нужная клавиша, но о том, каким пальцем до нее добраться. Думайте о пальцах, об остальном пусть думают они.

Не спешите, пытаясь за день освоить печать. Каждодневные упражнения по 15-30 минут, переход на другой уровень, а через день возвращение на прежний – все это нормально. Повторяйте предыдущие уровни, переходите на следующие, пока не почувствуете, что можете печатать. Тогда начинайте печатать свободный текст (в программе), затем в любом редакторе. И не бросайте это после того как освоили печать. Должно пройти какое-то время, чтобы привычка закрепилась.

До завершения работы над книгой (если я надумаю ее завершать), я постараюсь, следуя своим собственным советам, освоить печать латиницей. Если и когда это произойдет, я обязательно расскажу вам, что из этого получилось.

Зачем я вообще затеял рассказ о программе для обучения печати вслепую? Мне хотелось

показать, насколько это просто, если вы работаете в современной среде программирования. Порой появляются проблемы, решать которые удобнее, используя программирование. Если вы уверены, что с созданием программы вам не справиться, вы либо не избавитесь от своей проблемы, либо выберете путь, который займет больше времени и отнимет у вас больше сил, и, в конечном счете, может и не дать нужного решения.

Некогда, принимая решение, сделать ли устройство с использованием процессора или обойтись более дешевыми цифровыми микросхемами, я решил отказаться от процессора. Решение с цифровыми микросхемами, когда было найдено, казалось достаточно простым и дешевым, легким в реализации. Однако первые же испытания устройства потребовали введения множества корректив. А первые испытания готового образца выявили необходимость и в смене некоторых принципиальных подходов. В итоге переделки, которые занимали бы минуты при использовании процессора, выливались в дни.

Я не программист, для меня привычнее включить паяльник и приборы, чтобы понять, что происходит с устройством. Мне кажется утомительным, например, использовать программы симуляции работы схемы вместо того, чтобы на клочке бумаги быстро набросать участок схемы, измерить, посмотреть осциллографом и все понять. Перебирая такие клочки, я готов честно признаться в своей неправоте.

И еще одно. Программирование само по себе очень увлекательно. Вот я упомянул, что не получил кодов клавиш при переключении раскладки клавиатуры на русскую. Уверен, что поиск причины этого окажется увлекательнейшим приключением. А смысл этого приключения не только в том удовольствии, которое получишь когда доберешься до финиша, но и в том, что, возможно, будет получен ответ на вопрос, почему при симуляции работы микроконтроллера, о чем я говорил в предыдущей главе, у меня возникли некоторые проблемы с подачей команд. Может быть эти две проблемы не связаны, или связаны?

Я уже говорил, что тот способ, которым я заставляю «мигать» клавиши схемы клавиатуры на экране, не слишком хорош. Он не позволяет быстро переделать программу под обучение печати русских текстов. Поменяйте соответствующие строки на строки вида

```
IF Key.Code = 81 THEN TextLabel1.Background = &HBFDFFF& 'q
```

на

```
IF Key.Text = "й" THEN TextLabel1.Background = &HBFDFFF& 'й
```

и получите возможность работать с русским языком. Конечно, это следует поменять и в том месте кода, где обрабатывается отпускание клавиши, поменять слова-задания в массивах. Можно сделать программу двуязычной. Копирования при написании кода программы больше, но сама программа изменится мало.

Последнее, о чем следует сказать, относится к тем, кто всегда стремится к совершенству всех своих созданий. В Linux есть профессионально созданная программа аналогичного назначения, которая называется KTouch. Она, видимо, использует более совершенную методику обучения, соответственно иначе построена. Если вам захочется внести подобные усовершенствования в свой проект, то можно ориентироваться на эту программу.

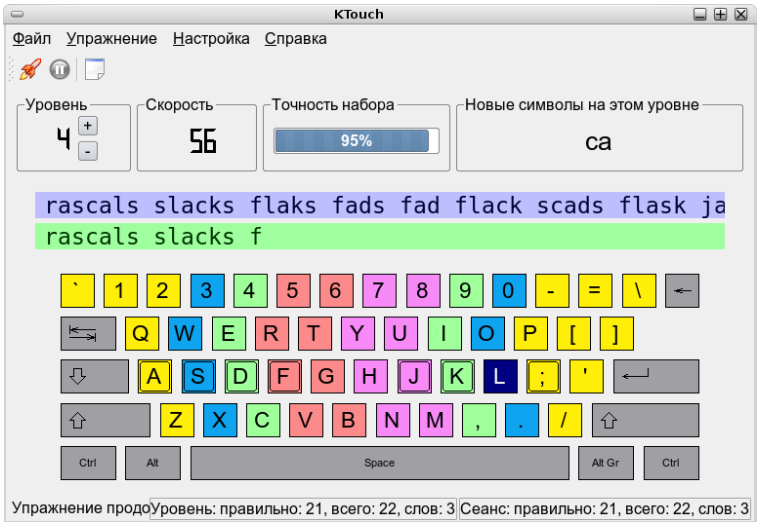


Рис. 3.9. Работа программы KTouch

Глава 4. Охота на кентавра

Пришла пора соединить интерфейс, созданный в Gambas, с микроконтроллером, запрограммированным в Piklab.

Засада в интерфейсе

Прежде, чем осуществить, и, конечно, проверить стыковку «на орбите», следует подправить обе части будущего единого целого. С этого и начнем.

Напомню, как выглядит интерфейсная часть проекта.

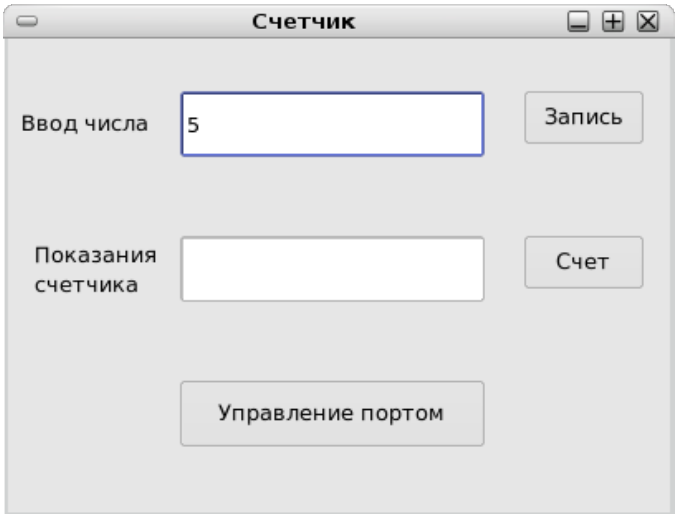


Рис. 1.4. Работа интерфейсной части проекта

Начинаем мы работу с нажатия на клавишу «Управление портом», чтобы открыть порт. Завершив работу повторно нажимаем эту клавишу для закрывания порта.

В окно ввода числа мы можем вписать число от 0 до 15. Не забыли, у нас 4 бита для записи в счетчик. После ввода числа нажимаем на клавишу «Запись», программа должна отправить введенное число в COM-порт. Дальше об этом числе должен позаботиться микроконтроллер, то есть, принять число, перенести его в свой порт, соединенный с выводами счетчика, предназначенными для задания значения предустановки, а затем после того, как мы нажмем на клавишу «Запись», интерфейс даст команду контроллеру на формирование сигнала предустановки. Контроллер, по окончании формирования импульса, должен прочитать состояние выходных бит счетчика и передать это в символьном виде программному интерфейсу. Интерфейс после получения числа должен вывести его в окно «Показания счетчика».

Из приведенного выше описания пожеланий следует сделать некий план работы. Не хочется называть его алгоритмом. План.

1. Ввод числа в окно ввода (обеспечено системой).
2. Нажатие (click) клавиши «Запись» отправляет число в символьном виде в COM-порт, предваренное символом команды «a», по которой микроконтроллер начнет обработку числа, формирование импульса записи и отправку в COM-порт символьное значение выходов счетчика.
3. Интерфейс ожидает появление ответа в COM-порте и выводит в окно «Показания счетчика» полученное значение или сообщение об отсутствии ответа от контроллера

после, например, 1 секунды ожидания.

Вот такой план. По мере его осуществления, я думаю, его придется подправить, но такова его участь. Запускаем среду программирования Gambas и начинаем с того места, где остановились в проекте «counter».

Первое, что я делаю, меняю план. Я подумал, что после запуска интерфейса, если когда-нибудь он будет готов, я могу забыть включить порт, и сразу попытаться ввести число и нажать на клавишу записи. Чтобы напомнить себе о необходимости открыть порт, я добавлю следующий код:

```
PUBLIC SUB Button1_Click()

      IF SerialPort1.Status THEN
            TextBox1.BackColor = Color.Green
      ELSE
            TextBox1.Text = "Вы забыли включить порт"
      ENDIF

END
```

Вы помните, что для получения шаблона подпрограммы обработки нажатия клавиши, достаточно двойного щелчка мышкой по клавише на форме. В данном случае клавиша «Запись» – это Button1. А фрагмент кода я вырезал из предыдущей заготовки проекта в том месте, где обрабатывалось нажатие на клавишу «Управление портом». Проверим, как работает эта вставка.

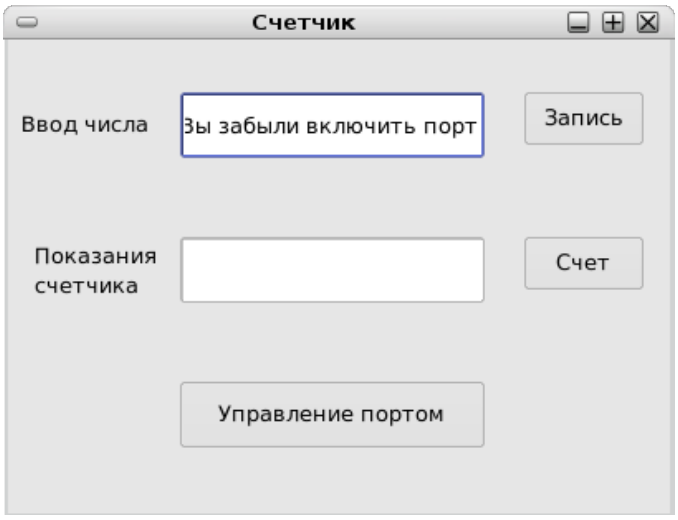


Рис. 4.2. Проверка изменений плана работы

Программный код в данный момент выглядит так:

```
' Gambas class file

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
```

```
END

PUBLIC SUB ToggleButton1_Click() 'клавиша «Управление портом»

      IF ToggleButton1.Value THEN
            SerialPort1.Open
            TextBox1.Text = SerialPort1.Status
      ELSE
            SerialPort1.Close
            TextBox1.Text = SerialPort1.Status
            TextBox1.BackColor = Color.Default
      ENDIF

END

PUBLIC SUB Button1_Click() ' клавиша «Запись»

      IF SerialPort1.Status THEN
            TextBox1.BackColor = Color.Green
      ELSE
            TextBox1.Text = "Вы забыли включить порт"
      ENDIF

END
```

Теперь по плану. Нажатием клавиши «Запись» отправим в порт символ команды «a» и символ числа, которое введем. Поскольку число 15 имеет два символа, подумаем, как их переправить в контроллер. Для упрощения задачи я предполагаю использовать два символа при записи всех чисел. То есть, 1 будет выглядеть как 01. Если этот вариант вам не нравится, то можно сейчас (или позже) добавить блок кода, который делал бы автоматическое преобразование, или при программировании контроллера учесть эту особенность ввода чисел.

Как отправить символ в COM-порт (порт последовательного ввода-вывода)? Вот, что я нахожу в разделе help:

> *SerialPort (gb.net)*
> *This class was designed to allow to communicate using a serial interface (usually RS-232 serial port). This class inherits from Stream class, so you can use standard streams methods to send and receive data, and to close the port.*
>
> *Последовательный порт (раздел gb.net)*
> *Этот класс был разработан для коммуникаций с использованием последовательного интерфейса (обычно, последовательный порт RS232). Класс наследует от класса Stream (поток), так что вы можете использовать стандартные потоковые метода для отправки и получения данных, и для закрывания порта.*

Насколько я понимаю, мне нужно отправиться к справке по работе с потоком:

> *Stream (gb)*
> *This class is the parent class of every Gambas object that are stream.*
> *These objects can be used with all the input/output Gambas functions: PRINT, INPUT, LINE INPUT, CLOSE, and so on.*
>
> *Поток (раздел gb)*
> *Этот класс родительский для каждого Gambas объекта, являющегося потоком.*
> *Эти объекты могут быть использованы со всеми функциями ввода-вывода Gmabas:*

PRINT, INPUT, LINE INPUT, CLOSE и т.д.

Таким образом, для отправки чего-либо в последовательный порт, я могу (или предполагаю, что могу) воспользоваться функцией PRINT. Хорошо. В help есть описание синтаксиса этой функции:

PRINT [#hStream ,] Expression [{;|;;| ,} Expression ...] [{ ; | ;; | , }]

Попробуем подставить вместо hStream наш порт SerialPort1, а что касается выражения (Expression), то я не знаю, что с ним делать. И после нескольких неудачных попыток останавливаюсь на таком варианте этого блока программы:

```
PUBLIC SUB Button1_Click() ' клавиша "Запись"
      IF SerialPort1.Status THEN
            TextBox1.BackColor = Color.Green
            PRINT #SerialPort1 "a"
            PRINT #SerialPort1 TextBox1.Text
      ELSE
            TextBox1.Text = "Вы забыли включить порт"
      ENDIF
END
```

При нажатии на клавишу «Запись», если последовательный порт открыт, то текстовое поле зеленеет (Color.Green), в порт отправляется символ «a», затем отправляется то, что введено в зеленеющее поле текстового ввода (TextBox1.Text). Если же я забыл открыть порт, то получаю в этом поле сообщение «Вы забыли включить порт» (надо было написать «открыть порт», но исправлять лень).

В таком виде при компиляции и после запуска программы ошибок не возникает. Не то, что в предыдущие попытки, когда я пытался понять, что мне делать с этими выражениями в функции печати. Я их превращал в значения, затем превращал в строки, пытался делать еще что-то не менее «умное», но ошибки возникали либо при компиляции, либо при работе.

Будем считать, что с этим блоком кода программы я справился.

До каких пор будем так считать? Ровно до тех, пока не станет ясно, что все это не так. Но при таком подходе хотелось бы ясности уже сейчас, пока кода мало, и легче разобраться с ошибками. Беда только в том, что я не знаю как проверить, действительно ли я отправляю что-то в порт, или это правильная с точки зрения языка конструкция, но не выполняющая ее предназначение.

Можно было бы подключить кабелем второй компьютер, на котором запустить программу чтения из последовательного порта (ее тоже быстренько написать?).

Я поступлю иначе, а вы решите для себя, что вам удобнее. Я использую то, что у меня осталось от предыдущих экспериментов, когда я макетировал модули для «умного дома». С макетной платой, на которой я в панельке размещал микроконтроллер, «общение» происходило именно по COM-порту через преобразователь интерфейса RS232 в RS485. Позже я приведу схемы.

Железное решение

Вот и вновь передо мною среда программирования микроконтроллеров Piklab, напомню, что контроллер PIC16F628A, а язык программирования Си.

Вот и вновь передо мною Piklab, где я подправлю программу, чтобы для начала получить команду «a», по которой зажгу светодиод, уже установленный на макетной плате. Вообще,

это удобно – использовать светодиоды на выводах порта контроллера. Можно многое проверить в разных экспериментах. Проект тоже называется «counter».

Я приведу только блок кода, в котором (по некоторому размышлению) изменил обработку полученных команд:

```
void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a': bitset (PORTA, 0); // «a», включить светодиод
            break;
            case 'b': bitclr (PORTA, 0); // «b», выключить
            break;
            default: break;
      }
}
```

Перед использованием программы на Gambas я приведу команды в соответствие.

Мне нужно только оттранслировать программу, выбрав в основном меню в разделе **Build** пункт **Build Project**. Программатор с установленной в панельку микросхемой контроллера у меня подключен давно. Остается подключить его к программе, раздел основного меню **Programmer**, а пункт **Connect**. И можно, проверив на странице Hex-кода, что слово конфигурации соответствует моим пожеланиям (2118, так оно выглядит в Piklab), можно записать программу в контроллер, что я и делаю. Для этого использую клавишу на основной инструментальной панели с изображением микросхемы, в которую направлена стрелка.

Программируется микросхема довольно быстро (программа маленькая), наученный горьким опытом, я проверяю правильность записи, щелкнув по клавише инструментальной панели с вопросительным знаком поверх микросхемы. Все в порядке. Остается самая для меня неприятная часть работы – найти, куда я «убрал» (запихнул, засунул и т.п.), куда я убрал макетную плату и блок питания. Пообедаю, и найду (?).

Скоро сказка сказывается, да не скоро дело делается. Но, с другой стороны, сколько веревочке ни виться, пора приниматься за работу.

В программе интерфейса, я вновь в Gambas, я использую вторую клавишу «Счет» для выключения светодиода путем отправки символа команды выключения «b». В блоке щелчка по клавише «Запись» я закомментирую лишнюю команду, а новый блок получается таким:

```
PUBLIC SUB Button2_Click() ' клавиша «Запись»
      PRINT #SerialPort1 "b"
END
```

Увы. То, что так убедительно и красиво на бумаге, в жизни бывает не менее убедительно, но не так красиво. Не работает макет, а причин может быть много. Первым делом нужно проверить, отправляется ли что-то в порт, когда я нажимаю клавиши передачи символов управления.

В моем распоряжении есть, по меньшей мере, два способа это проверить – включить осциллограф, чтобы попытаться увидеть сигнал на выходе COM-порта, или попытаться мультиметром увидеть изменение сигнала на том же выводе порта или в линии RS485. Если с линией мне все ясно, она хорошо выделена у меня на макетной плате, то с выводом порта сложнее. Для начала следует найти схему включения конвертера RS232-RS485.

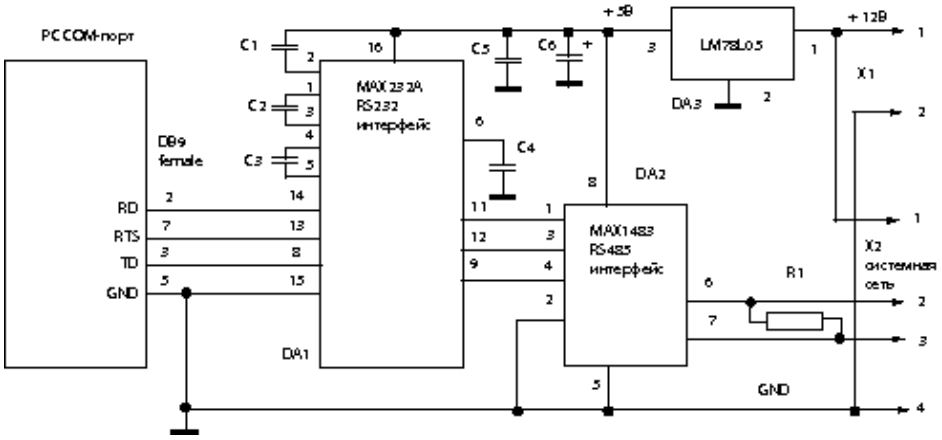


Рис. 4.3. Схема конвертера RS232-RS485

Проверка состояния вывода RTS (13 вывод микросхемы MAX232) с помощью мультиметра показывает, что при первом нажатии клавиши «Управление портом» напряжение на этом выводе меняется (на напряжение противоположного знака), а второе нажатие возвращает его к первоначальному виду. Похоже, обращение к порту есть. Но попытка отправить что-то функцией PRINT не дает видимых изменений на выводе 8 микросхемы MAX232. Хуже того, программа, как выяснилось, виснет, не позволяя закрыть порт повторным нажатием клавиши «Управление портом». Что это? Кого ждем?

Видимо, ждем закрытия порта, поскольку, если подождать подольше, то порт закрывается. Почему так долго?

И это не единственное почему. Подключив осциллограф к выводу 8 микросхемы MAX232A, где я ожидал бы увидеть передаваемые с компьютера данные, я их не вижу. Работая над книгой «Умный дом», я в начальный момент в схожей ситуации использовал программу тестирования COM-порта, позже использовал собственную программу, написанную в среде Visual Basic. Но эти возможности касались работы в Windows, тогда как сейчас я использую Linux. Думаю, программы для работы с COM-портом есть и для Linux, но быстро найти хотя бы одну из них не получается. И я вспоминаю, что был еще вариант программы управления модулями «умного дома», написанный в среде KDevelop из дистрибутива Linux.

Я запускаю KDevelop, и чтобы не «изобретать велосипед», просто пытаюсь проделать все то, что описал в соответствующей главе собственной книги. Делаю при этом несколько поспешных ошибок, делаю вывод о том, что описание не мешало бы сделать более подробным (сам виноват, тогда казалось, что этим обидишь читателя), поскольку далеко не всегда понимаю, что следует сделать, отыскиваю в Интернете нужный для этих целей пакет, дополняющий классы необходимыми для работы с последовательным портом, и... застреваю на компиляции после попытки добавить эти классы. Вот так. Упростил себе задачу! Мне казалось, что я быстро смогу повторить сделанную некогда программу, упростив ее до тестового варианта. Мне не много нужно – открыть и закрыть порт, да отправить в порт символ «а», правда в виде строки вида «аааааааааааааааааааааа». Одиночный символ наблюдать на экране осциллографа достаточно сложно, проще увидеть наличие «жизни» на выводах микросхемы при передаче нескольких символов. Но у меня до этого наблюдения дело не доходит. Я застреваю на компиляции файла из-за того, что добавленные классы работы с портом обращаются к библиотеке Qt, есть такая в Linux, и не могут с ней разобраться даже с моей помощью. Но я твердо помню, что все это проделывал года два

назад, все работало, была даже программа, прекрасно работавшая с той же макетной платой, что я использую сейчас. Если она была, ее надо найти.

Отыскать что-либо на моем компьютере по прошествии двух лет задача не для слабонервных. Каждый раз, сталкиваясь с этим, я даю себе слово навести порядок, но попытки навести порядок, боюсь, предвносят не меньше беспорядка, чем создается его в «плановом порядке». Все, после очередной переустановки операционной системы наведу полный порядок.

Нужная мне программа после получаса поисков находится на CD-диске. Ее компиляция не вызывает особых проблем. Остается выяснить, в чем разница?

Она только в версии добавленных мною файлов. Я скачал из Интернета последнюю версию, а компиляция проходит с более ранней. Самое простое, что я могу придумать – скопировать нужные мне файлы работы с последовательным портом из старого проекта в новый. Это заметно улучшает процесс «оживления» программы, приходится только закоментировать несколько строк, относящихся к проверке использования пакета для Linux или Windows. Теперь у меня проходит компиляция и нового проекта. Остается поправить несколько строк кода, чтобы вернуться к попытке увидеть данные на выходе COM-порта.

Попытка быстро проверить работу новой программы с помощью некогда работавшей казалась мне удачной находкой, правильным и быстрым решением. Я ошибался. Но теперь мне хочется довести это «лирическое отступление» до логического завершения, тем более я не вижу другого пути. Мне нужно убедиться в том, что COM-порт работает, что конвертер и макетная плата работают, поскольку за прошедшие годы всякое могло случиться. Пора проверить все.

Итак. Схему конвертера я привел выше. Разъем X2 на этой схеме передает напряжение питания 12 В (выводы 1 и 4) и сигнал в линию RS485 (выводы 2 и 3).

На макетной плате расположена микросхема интерфейса RS485 и панелька для микроконтроллера PIC16F628A.

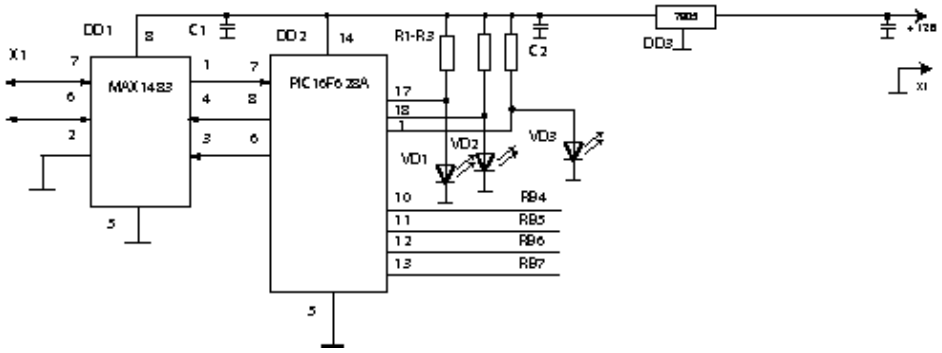


Рис. 4.4. Схема макетной платы

Используемые мною микросхемы MAX1483 интерфейса RS485 имеют штатное, практически, включение, совпадающее с включением аналогичной микросхемы MAX3082.

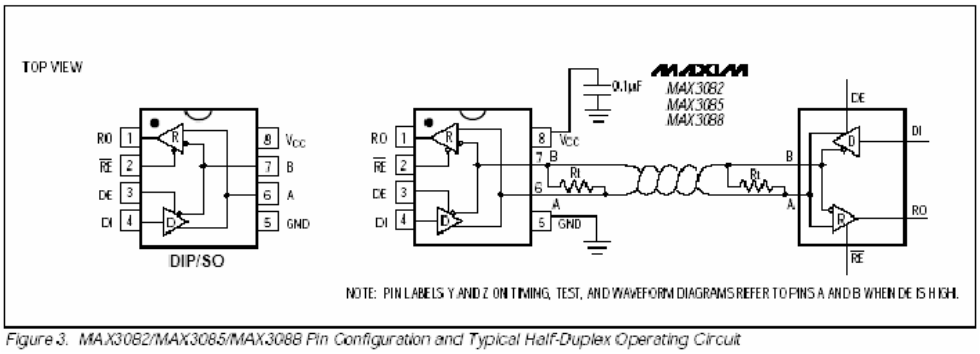


Figure 3. MAX3082/MAX3085/MAX3088 Pin Configuration and Typical Half-Duplex Operating Circuit

Рис. 4.5. Штатное включение микросхемы MAX1483

И я хочу проверить работу порта, наблюдая сигналы на его выходе, на выходе микросхемы MAX232A, в линии RS485, на выходе интерфейсной микросхемы макетной платы. Для этого я хочу из некогда работавшей программы проекта, написанного на C++ в среде KDevelop, сделать простую программу работы с COM-портом.

Лирическое отступление

Я не планировал обращаться к среде программирования KDevelop. Обычно не принято говорить с людьми о своих знакомых, которых твой собеседник не знает, и мне остается только познакомить (бегло и наспех) вас с моим «шапочным» знакомым. То что мое знакомство «шапочное» характеризует не с лучшей стороны меня, но никак не среду программирования KDevelop. Очень хорошая и удобная, с моей точки зрения, среда программирования на множестве языков.

Сразу после запуска программы, если вы не создали еще ни одного проекта, окно редактирования KDevelop пусто, хотя рабочее окно имеет помимо основного меню еще множество закладок. Выбрав в основном меню раздел **Проект**, с помощью подменю **Новый проект...** вы попадаете в диалог создания нового проекта. Если установить опцию (чуть ниже окна дерева выбора языка программирования слева) **Показать все шаблоны проекта**, то вы можете подивиться тому обилию языков программирования, которым можете воспользоваться для своих целей. Но меня интересует только быстрое повторение старого проекта, при удалении из него лишнего, который был сделан как простейшее приложение KDE на C++. Этот вариант я и выбираю.

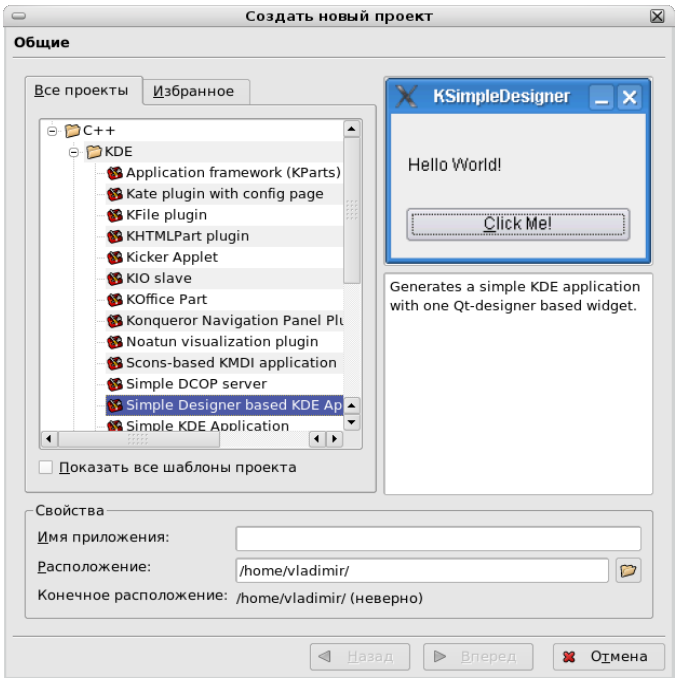


Рис. 4.6. Окно диалога выбора языкового шаблона в KDevelop

После ввода имени проекта (я назвал его counter_test) безучастная пока клавиша **Вперед** оживает, можно двигаться дальше. В этой версии необходимо ввести имя автора на следующей странице диалога, и можно дополнительно, если вы знаете как этим воспользоваться в дальнейшем, задать управление версиями и подправить шаблон файла заголовков проекта. После завершения всех подготовительных процедур и нажатия на клавишу **Готово** можно приступать к работе над проектом.

В качестве заготовок (шаблонов) у меня есть основной файл программы, файл заголовков, файл графического интерфейса, который можно исправить. Но прежде, чем править интерфейс, следует разобраться с добавлением нужного мне класса (или классов), требуемого для работы с последовательным портом. Для этого можно воспользоваться либо клавишей инструментального меню (крайней слева), подсказка которой предлагает **Генерировать новый класс**, либо в разделе основного меню **Проект** выбрать пункт **Новый класс**. Эту операцию я проделываю трижды, для каждого из трех файлов пакета: Posix_QextSerialPort, QextSerialBase и QextSerialPort.

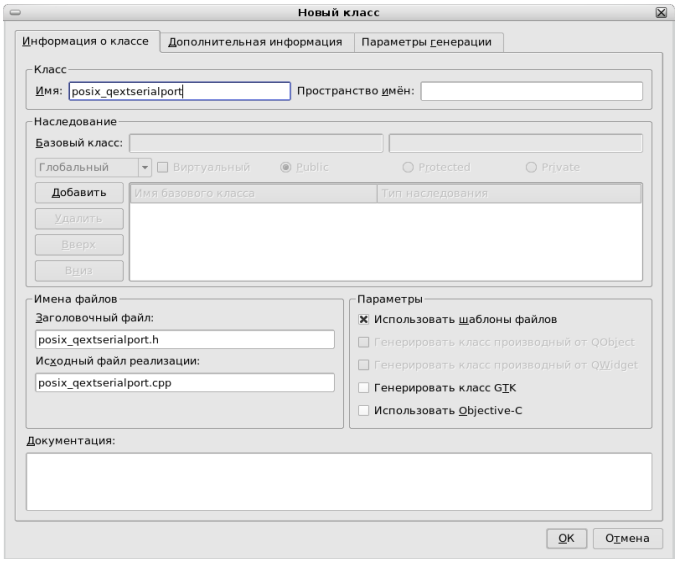


Рис. 4.7. Диалог создания нового класса в KDevelop

После нажатия на клавишу ОК появляется диалоговое окно, в котором можно выбрать, будет ли созданный класс добавлен к проекту. Я соглашаюсь добавить его к проекту. После трехкратного повторения этих операций мой проект пополняется тремя классами. Теперь я меняю созданные файлы (программы и заголовка) теми файлами, что есть в пакете (старой версии). Просто перезаписываю их. Кроме этого в параметрах, по старой памяти, в разделе **Поддержка C++** на странице **Авто-дополнение кода**, там где рядом с окном баз данных для авто-дополнения кода я нажимаю клавишу **Добавить**, чтобы добавить Qt.

Теперь в разделе Сборка нажимаем пункт **Запустить automake и др.**, когда процесс успешно завершен, запускам в том же разделе пункт **Запустить configure**, а следом **Собрать проект**.

После первой попытки компиляции я подправляю файл qextserialport.h:

```
/*POSIX CODE*/
//#ifdef _TTY_POSIX_
#include "posix_qextserialport.h"
#define QextBaseType Posix_QextSerialPort

/*MS WINDOWS CODE*/
//#else
//#include "win_qextserialport.h"
//#define QextBaseType Win_QextSerialPort
//#endif
```

Правка свелась к удалению нескольких строк (я их закомментировал). После этого компиляция проекта стала проходить без ошибок.

Теперь я намерен переделать шаблон графического интерфейса. Мне нужны три кнопки: открыть порт, закрыть порт, запись и еще одна, которую я называл счет, но к счету она не будет иметь отношения. Прежде, чем мне удалось внести эти исправления, пришлось использовать подменю, которое появляется после щелчка правой клавиши мышки по форме, где я использовал попеременно пункты **Разорвать расположение** и **Расположить по сетке**, сохраняя сделанные изменения.

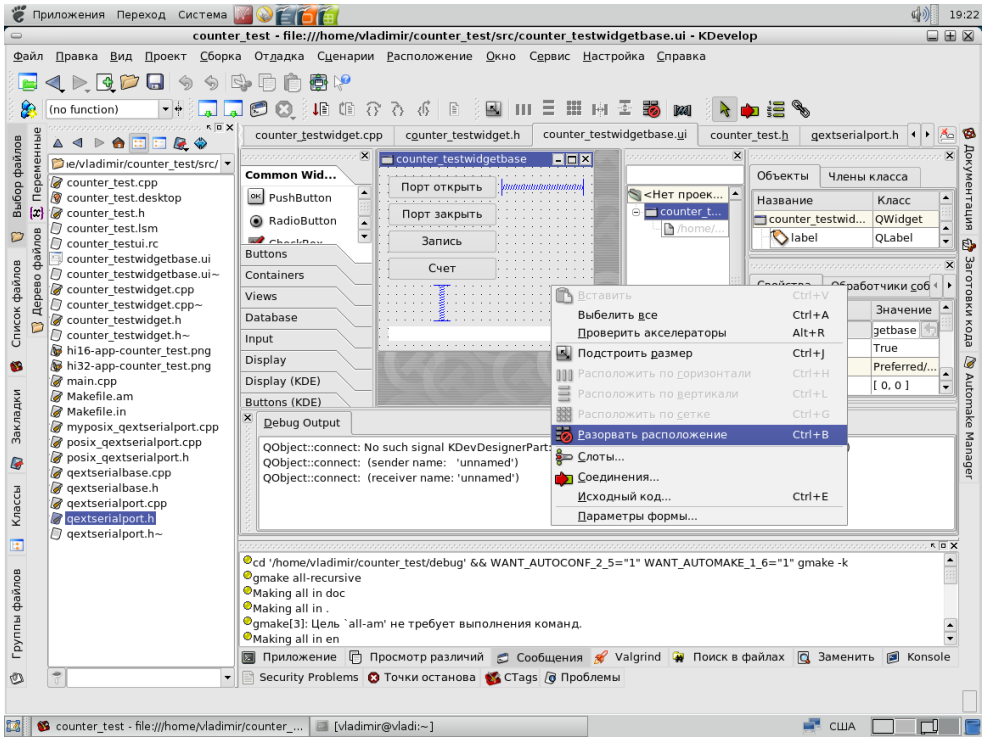


Рис. 4.8. Переделки графического интерфейса

После переделок интерфейса в разделе выпадающего меню Слоты... я добавляю слоты по аналогии с уже существующим, используя клавишу **Добавить функцию**, а в разделе Соединения... добавляю соединения с помощью клавиши **Новое**. Мне остается добавить в файл counter_testwidget.h:

```
public slots:
    /*$PUBLIC_SLOTS$*/
    virtual void button1_clicked();
    virtual void button2_clicked();
    virtual void button3_clicked();
    virtual void button4_clicked();
```

несколько строк (я повторяю имеющуюся запись virtual void button1_clicked(), где добавил только 1 после имени button). Аналогично я поступаю с файлом counter_testwidget.cpp:

```
void counter_testWidget::button1_clicked()
{
    if ( label->text().isEmpty() )
    {
        label->setText( "Hello World!" );
    }
    else
    {
        label->clear();
    }
}

void counter_testWidget::button2_clicked()
{
}

void counter_testWidget::button3_clicked()
```

```
{
}

void counter_testWidget::button4_clicked()
{
}
```

Где к развернутой записи о button1 (здесь я тоже добавил 1) я, скопировав, естественно, добавил пустые пока записи об остальных клавишах интерфейса. Именно в этот файл, опять путем копирования, я собираюсь перенести все необходимые мне события. В результате этих манипуляций файл counter_testwidget.cpp выглядит так:

```
#include <qlabel.h>

#include "counter_testwidget.h"
#include "posix_qextserialport.h"
#include "qextserialport.h"
#include "qextserialbase.h"

Posix_QextSerialPort comPort("/dev/ttyS0");

counter_testWidget::counter_testWidget(QWidget* parent, const char* name, WFlags
fl)
        : counter_testWidgetBase(parent,name,fl)
{}

counter_testWidget::~counter_testWidget()
{}

/*$SPECIALIZATION$*/
void counter_testWidget::button1_clicked()
{
      comPort.setName("/dev/ttyS0");
      comPort.setDataBits(DATA_8);
      comPort.setStopBits (STOP_1);
      comPort.setBaudRate (BAUD2400);
      comPort.open(0);
      comPort.setRts (FALSE);
      if (comPort.isOpen())
{
      label->setText( "port open" );
}
}

void counter_testWidget::button2_clicked()
{
      comPort.close();
      if (!comPort.isOpen())
l     abel->setText( "port close" );
}

void counter_testWidget::button3_clicked()
{
      comPort.writeBlock("a", 1);
}

void counter_testWidget::button4_clicked()
{
      comPort.writeBlock("b", 1);
}
```

```
#include "counter_testwidget.moc"
```

Я не знаю, все ли правильно, но при сборке проекта претензий ко мне нет. К этому моменту я увидел две ошибки, допущенные при настройке порта в Gambas: скорость я задавал 9600, и бит RTS не сбрасывал. Есть надежда, что это и есть источник проблем. Запускаем программу... и...

«Но программа работала!» – говорю я.

«Не умеешь, не берись!» – отвечает внутренний голос.

Сей веселый диалог единственный результат эксперимента. Осциллограф не показывает наличие изменений на выводе передачи порта, микроконтроллер никак не реагирует на команды. А я точно помню, что программа работала, сейчас же она ведет себя точно так же, как ее Gambas-аналог. Даже подвисание при закрывании порта имеет место. Может быть, прав внутренний голос? Ну, нет! Внутренний голос может принадлежать моей лени, но никак не моему упрямству.

Вспоминая, как я некогда пытался устроить диалог между программой и микроконтроллером, я готов повторить еще одну проверку. В прошлый раз я отыскал в Интернете программу, которая позволяла проверить работу COM-порта. Эта программа проработала дней 15, а затем перестала работать, не работала и после переустановки. Мне хватило этих дней для продолжения работы, более того, позже я нашел в своем архиве еще одну, которая была получена с оборудованием и предназначалась для его настройки. Эту программу я готов отыскать.

И отыскиваю, хотя со столь обильным ворчанием, что его хватило бы на месяц работы. Программа предназначена для работы в Windows, но охотно работает под эмулятором Wine, не требует установки, и полностью ставит меня в тупик. У меня такое ощущение, что я что-то упустил между запуском программы в Gambas и в KDevelop. Мелькнули какие-то соображения, которые я тут же забыл. У программы есть окошко, в которое можно ввести символ управления, напомню, для зажигания светодиода я использую символ «a», для гашения – «b». Выключив клавишу RTS в программе, есть у нее такая возможность, я наблюдаю, как зажигается и гаснет светодиод на макетной плате под управлением микроконтроллера, подчиняющегося отправляемым мною символам. Даже осциллограф включать не надо. Обидно.

С другой стороны, это лучше, чем испорченный COM-порт. Хоть это утешает.

Я еще раз просматриваю настройки порта в программе, не нахожу отличий: 2400, 8 бит данных, 1 стоповый бит, нет проверки на четность. Все, как я прописываю в тестовой программе в KDevelop, и те же настройки, что я задавал в программе на Gambas.

Расстроенный и разочарованный, сердитый на себя и на весь мир, я запускаю программу на Gambas, чтобы наблюдать еще одно явление: после открывания порта, отправки команды «зажечь светодиод (запись)», если я закрываю порт, то светодиод зажигается. Я могу погасить его манипулируя клавишей «Управление портом» и клавишей гашения (счет). Но время позднее, я выключаю компьютер до завтрашнего дня. Новый день пусть будет добрее ко мне.

Будет. Но не сейчас. Опять программа на Gambas не работает даже с уговорами через открывание и закрывание порта, программа в KDevelop с ней солидарна. Это еще один из аргументов к моим размышлениям, не пора ли прекратить всю эту писанину. Linux мало кого интересует, радиолюбители больше интересуются справочными данными и конкретными указаниями, какие настройки нужно сделать, чтобы схема заработала. А ты с дилетантскими

рассуждениями о программировании! Похоже, что пора. Еще раз попробую все с самого начала. Первой я запускал программу в Windows для работы с COM-портом. Затем запустил программу в Gambas.

К загадке, почему не работает ранее работавшая программа, добавляется загадка странного поведения недавно написанной программы, поскольку с помощью чудноватого манипулирования открыванием и закрыванием порта я вновь могу зажигать и гасить светодиод на макетной плате.

Вот. Я вспомнил, что хотел проверить. При работе модема нужно то ли включать, то ли отключать контроль потока, программный или аппаратный, уж и не помню. Переместившись в программу, написанную в среде KDevelop, я нахожу, как отключить этот контроль:

```
comPort.setFlowControl(FLOW_OFF);
```

Добавленная к настройкам порта при его открывании эта строка устраняет все странности. Все работает. Думаю, в прошлый раз мне просто повезло. Видимо, при включении компьютера порт получал настройки по умолчанию, включающие этот параметр. Мне пришлось его задавать специально.

Теперь я могу вернуться в Gambas, где в свойствах последовательного порта есть и этот параметр. Мне даже не приходится добавлять код в программу.

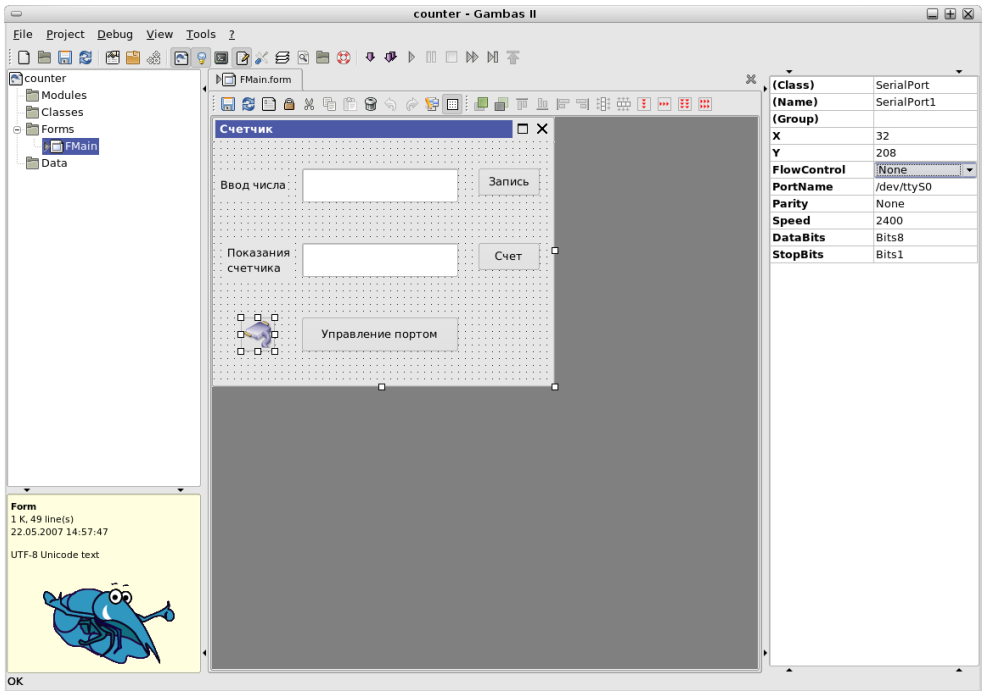


Рис. 4.9. Задание параметра контроля потока порта в Gambas

Включив порт клавишей «Управление портом», щелкая клавишами «Запись» и «Счет» я могу включать и выключать светодиод, как и планировалось изначально. И вывод – не следует полагаться на удачу, следует познакомиться с работой оборудования, которое ты предполагаешь использовать, и разобраться с его параметрами. И задать все параметры без исключения, даже если в прошлый раз повезло, и кто-то сделал это за тебя. Надейся на себя. Так будет вернее.

Грустное завершение рассказа о счетчике

Пришла пора расстаться с проектом проверки счетчика. Я не собирался его выполнять полностью и проверять на макетной плате. Однако, чтобы не испытывать угрызений совести, я использую существующую макетную плату, на которой есть три светодиода (см. рис.4.4), подключенных к выводам порта А RA0, RA1 и RA2. Выводы RA3, RA4 и RA5 имеют «подтягивающие» резисторы (их нет на рисунке, как и выводов). Под эти возможности готового макета я переделаю программы.

По команде «Запись» от графического интерфейса микроконтроллер должен переписать заданное число, три бита позволят мне использовать числа от 0 до 7, в три младших бита порта А. А по команде «Счет» прочитать состояние следующих трех выводов порта А, сделаем вид, что они соединены с выходами счетчика, и отправлять их состояние в символьном виде обратно. Таким образом микроконтроллер получает две команды, я оставлю их прежними, хотя легко можно заменить их любыми подходящими символами или набором символов. После первой команды контроллер получает число в символьном виде, которое должен обработать для отображения светодиодными индикаторами. Я могу соединить выводы RA0 и RA3, RA1 и RA4, и т.д. Но могу и просто подключать входные выводы к общему проводу, как мне заблагорассудится.

Начать писать код программы можно с интерфейсной части проекта в Gamabs, а можно вначале запрограммировать микроконтроллер в программе Piklab. Пусть будет Piklab.

Начну я с того, что подправлю файл заголовка:

```
#define bitset(var,bitno) ((var) |= 1 << (bitno)) // установка бит порта
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno))) // сброс бит порта
unsigned char getch(); // прием символа
void putch(unsigned char); // передача символа
void     init_comms(); // инициализация коммуникации
void cmd();
```

Правка касается пока только добавления функции передачи символа, с помощью которой можно будет отправлять с контроллера информацию о состоянии выводов порта А, которые якобы соединены с выходом счетчика.

В основном тексте программы сразу следует изменить строку в функции void init_comms(). Именно в этой функции задается конфигурация порта А, где прежде все выводы были направлены на выход. Теперь три из них следует перенаправить на вход.

```
TRISA = 0x38;
```

Не всегда я так поступаю, что неправильно, но на этот раз я проверю изменилось ли состояние выводов контроллера с помощью программы gpsim. После запуска симуляции я могу убедиться, что выводы порта А соответствуют желаемому.

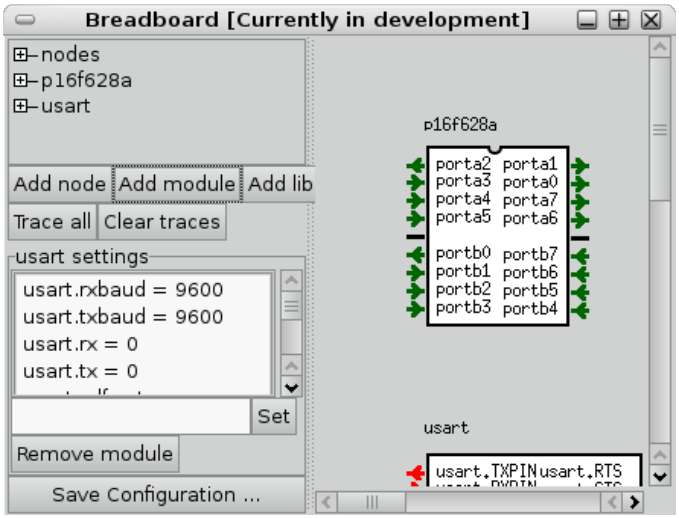


Рис. 4.10. Результат конфигурации выводов порта A в программе gpsim

На макетной плате проекта, где стрелки выводов портов показывают, как они будут работать, видно, что выводы porta3 – porta5 должны работать на ввод.

В файл заголовка я добавлю еще одну переменную и две функции, отвечающие командам интерфейсной части проекта «Запись» и «Счет»:

```
unsigned char NUMBER;
void cmd_write ();
void cmd_count ();
```

А сами функции определю в основном файле следующим образом, но последовательно проверяя их работу:

```
void cmd_write ()
{
      NUMBER = input;

      switch (NUMBER) {
            case '0':
            bitclr (PORTA, 0);
            bitclr (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            case '1':
            bitset (PORTA, 0);
            bitclr (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            case '2':
            bitclr (PORTA, 0);
            bitset (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            case '3':
            bitset (PORTA, 0);
            bitset (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            case '4':
```

```
            bitclr (PORTA, 0);
            bitclr (PORTA, 1);
            bitset (PORTA, 2);
            break;
            case '5':
            bitset (PORTA, 0);
            bitclr (PORTA, 1);
            bitset (PORTA, 2);
            break;
            case '6':
            bitclr (PORTA, 0);
            bitset (PORTA, 1);
            bitset (PORTA, 2);
            break;
            case '7':
            bitset (PORTA, 0);
            bitset (PORTA, 1);
            bitset (PORTA, 2);
            break;
            default: break;
            }
}
```

Сразу внесем изменение в функцию приема команды:

```
void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            input = getch();
            cmd_write ();
            break;
      }
}
```

И проверим, работает ли эта часть программы. О пользе проверки программы после даже небольших изменений я выше говорил, а сейчас готов подтвердить это личным примером. Первая же трансляция заставляет меня искать ошибки – опечатки, которую я умудрился добавить в код программы. И возникает вторая проблема, как проверить правильность кода. Логично это сделать в программе симуляции gpsim, но отношения с gpsim в части использования USART у меня как-то не сложились.

На всякий случай я еще раз перепроверяю эти отношения, но увы мне! Единственное, что остается, использовать макетную плату для непосредственной проверки. Запрограммировав микроконтроллер в программе Piklab, водрузив контроллер в панельку макетной платы, я подключаю ее к COM-порту и запускаю интерфейс в Gambas. Основные команды, в которых я использую отправку символов «a» и «b», продолжают работать, но попытка ввести в окно ввода числа с тем, чтобы светодиоды отобразили результат этого нововведения, оканчивается неудачей. Скорее всего, мое видение программы, написанной для микроконтроллера, и его видение этой программы отличаются слишком сильно. Это же может касаться и интерфейсной части.

Попробую разобраться. Вначале с интерфейсом в Gambas. Здесь у меня только одно сомнение, правильно ли я пытаюсь прочитать ввод в окне «Ввод числа» и передать его в COM-порт. Немного повозившись с разными вариантами проверки этого, я нахожу правильное решение – если команда с помощью отправки символа a:

```
PRINT #SerialPort1 "a"
```

отрабатывается исправно, то я могу, удалив эту строку из текста программы, использовать ввод не числа, а этого символа в окно ввода числа. Тогда следующая строка:

```
PRINT #SerialPort1 TextBox1.Text
```

отправит команду после нажатия на клавишу «Запись». Так ли это? Я проверяю, и убеждаюсь, что это так. Осталось восстановить текст интерфейсной части проекта и считать, что с этой частью в настоящий момент все в порядке.

Теперь стоит подумать, как проверить программу микроконтроллера? Здесь в первую очередь мне хотелось бы понять, попадаю ли я в функцию cmd_write (). Почти вся функция должна проверяться работой светодиодов, а то, что они не горят, может свидетельствовать как о том, что в функцию я не попадаю, так и о том, что, я минуя переключатель switch (NUMBER) в этой функции. Чтобы проверить это, я добавлю в конец функции проверку:

```
void cmd_write () {

      NUMBER = input;

      switch (NUMBER) {
            case '0':
            bitclr (PORTA, 0);
            bitclr (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            ..............................................
            default:
            bitset (PORTA, 2); // проверка
            break;
            }
}
```

То есть, если я не могу отработать значение числа, но в функцию обработки попадаю, светодиод на выводе RA2 будет загораться. Программируем контроллер. Переносим на макетную плату. Убеждаемся, что светодиод горит. В функцию записи я попадаю, но переменная NUMBER не отвечает ни одному из перечисленных случаев. Припоминая прошлый опыт работы с функцией получения символа getch(), я достаточно много провозился с ней, но, боюсь, не в полной мере понял ее работу. Можно сделать еще одну попытку понять работу этой функции, но я пытаюсь переместить ее с места на место, чтобы обойти проблему. И в одном из вариантов, размышляя о собственном бессилии, пытаясь придумать нечто полезное, я перебираю все числа, которые следует ввести для передачи, отправляя их через COM-порт в микроконтроллер. Неожиданно для себя я обнаруживаю реакцию контроллера на эти числа. Они не отрабатываются должным образом, но контроллер реагирует на них. По этой причине мое внимание привлекает конструкция кода вида:

```
void cmd_write () {

      NUMBER = input;

      switch (NUMBER) {
            ................
            case '1':
            bitset (PORTA, 0);
```

```
            bitclr (PORTA, 1);
            bitclr (PORTA, 2);
            break;
            ..................
            }
}
```

В моем понимании эта конструкция устанавливает вывод RA0 в единицу, а следующие два вывода порта в ноль. Но, насколько я помню, это может быть и не так. Программа может последовательно перебирать заданные команды, каждый раз при установке или сбросе бита очищая все остальные. То есть в показанном случае сброс вывода RA2 окажется последней командой, которая сбросит и RA1, и RA0. Были у меня и прежде «непонятки» с использованием функций побитовой установки и сброса.

Я меняю эту конструкцию на другую:

```
void cmd_write () {

      NUMBER = getch();

      switch (NUMBER) {
            ..............................
            case '1':
            PORTA = 0x1;
            break;
            ..............................
            }
}
```

и это срабатывает. Теперь основная программа для микроконтроллера выглядит следующим образом:

```
/* ------------------------------------------------------------------------ */
/* Template source file generated by piklab */
#include <pic16f628a.h>

/* ------------------------------------------------------------------------ */
/* Configuration bits: adapt to your setup and needs */

typedef unsigned int word;
word at 0x2007 CONFIG = _WDT_OFF & _PWRTE_OFF & _INTOSC_OSC_NOCLKOUT &
_MCLRE_OFF & _BOREN_OFF & _LVP_OFF & _DATA_CP_OFF & _CP_OFF;

#include <stdio.h>
#include "counter_start.h"

unsigned char input;    // Для считывания приемного регистра
unsigned char COMMAND;  // символ команды
unsigned char NUMBER;


unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}
```

```
void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            getch();
            cmd_write ();
            break;
            case 'b': bitset (PORTA, 1);
            break;
            default: break;
            }
}

void cmd_write () {

      NUMBER = getch();

      switch (NUMBER) {
            case '0':
            PORTA = 0x0;
            break;
            case '1':
            PORTA = 0x1;
            break;
            case '2':
            PORTA = 0x2;
            break;
            case '3':
            PORTA = 0x3;
            break;
            case '4':
            PORTA = 0x4;
            break;
            case '5':
            PORTA = 0x5;
            break;
            case '6':
            PORTA = 0x6;
            break;
            case '7':
            PORTA = 0x7;
            break;
            default:
            break;
            }
}

void cmd_count ()
{

}

void init_comms() // Инициализация модуля
{
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x38;
      TRISB = 0xFE;
```

```
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      bitclr (PORTB, 0);      // Выключаем драйвер RS485 на передачу
}

void main(void) {
      init_comms(); // Инициализация модуля

start:      CREN =1; // Начинаем работать
      //getch();
      input = getch();
      cmd();
      goto start;
}
```

Светодиоды зажигаются соответственно вводимому в окно «Ввод числа» числу.

Остается реализовать и отладить ту часть программы микроконтроллера, которая будет передавать состояние вводов от микроконтроллера к интерфейсу. А код программы интерфейса дополнить чтением из COM-порта того, что отправил микроконтроллер.

В интерфейсной части проекта в Gambas я добавляю одну строку:

```
PUBLIC SUB Button2_Click()
  PRINT #SerialPort1 "b"
  INPUT #SerialPort1 TextBox2.Text
END
```

До переделки кода программы микроконтроллера интерфейс выводит в окне приема то, что я отправляю в контроллер.

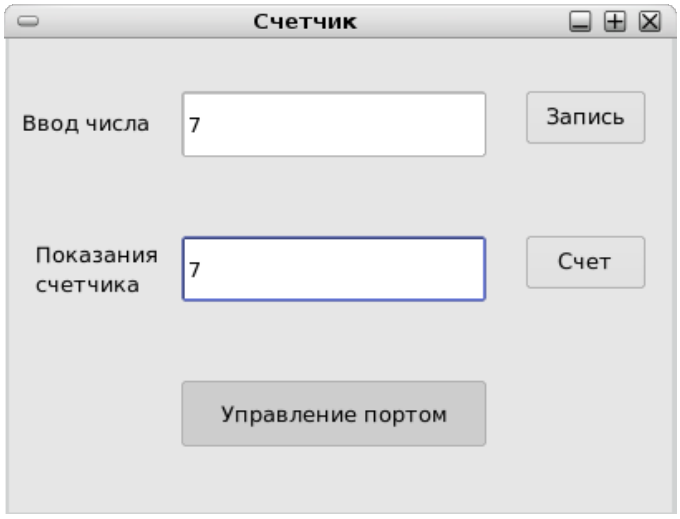


Рис. 4.11. Получение символов от COM-порта в интерфейсной части проекта

Посмотрим, что будет после переделки, с которой, как я подозреваю, тоже не все будет гладко.

В основной текст программы для микроконтроллера я добавляю функцию отправки символа:

```
void putch(unsigned char byte) // Отправка байта
{
      while(!TXIF)      // Устанавливается, когда регистр не пуст
            continue;
      TXREG = byte;
}
```

не забыв объявить ее в файле заголовка.

В функцию приема команд я добавляю отправку состояния порта, добавив переменную unsigned char COUNTER в начало программы:

```
void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            getch();
            cmd_write ();
            break;
            case 'b': cmd_count();
            break;
            default: break;
            }
}
```

И, конечно, заполняю функцию отправки значения вводов порта:

```
void cmd_count ()
{
      COUNTER = PORTA;
      bitclr (PORTA, 0); // Проверка
      putch(COUNTER);
}
```

Строку проверки, признаюсь, я добавил после первого неудачного опыта. Последовательность эксперимента такова – я вписываю число 7, нажимаю клавишу «Запись», затем нажимаю клавишу «Счет», ожидая получить нечто, соответствующее моим ожиданиям. Поскольку я записываю семерку, то все светодиоды должны загораться. Если я попадаю в функцию отправки микроконтроллером символа, то один из светодиодов должен погаснуть. Проверяю.

Гаснут все светодиоды. Что напоминает мне о двух вещах: не следует наступать на грабли, о которых только что говорил (использование конструкции bitclr (PORTA, 0)), и второе – не мешало бы разрешить передачу!

Я вношу исправления, копируя необходимые части программы из прежнего проекта, а в итоге перестает работать все! Приплыли.

Глава 5. Сказка о неудачливом радиолюбителе

Нет, нет, и нет. Не буду.

Это я уговариваю себя отказаться от полномасштабной проверки проекта «counter». То есть, впаять в макетную плату панельку для счетчика, запаять все микросхемы перехода с RS232 на RS485 и т.д. Иногда я скучаю по пайке. Тогда включаю паяльник и начинаю что-нибудь паять. Но сегодня, чтобы спаять что-нибудь нужное, мне надо поехать в «Чип и Дип» и купить панельки, микросхемы, да и резисторов не мешало бы прикупить, еще несколько микросхем операционных усилителей никак не соберусь купить, не забыть бы...

Нет, нет, и нет. Магазин возле Курского вокзала, в который мне удобно было добираться, закрылся, а ехать в другие края сегодня не хочется. Да и с проектом я как-то «завял» даже на уровне макетной проверки. Одно другому не мешает, можно спаять новую макетную плату, подготовив ее к окончательной проверке, затем вернуться к нерешенным проблемам. Но нет. Если не решить возникшие проблемы, если не разобраться в причине неудач, то покупка всего необходимого для новой макетной платы с последующей пайкой может оказаться пустой тратой денег.

Начнем все сначала.

Возвращение на круги своея

Я возвращаюсь в программу Piklab, где, пользуясь текстом программы приведенным выше, когда у меня еще что-то работало, возвращаю программу микроконтроллера к ее раннему состоянию. Сейчас она выглядит так:

Файл заголовка counter_start.h

```
#define bitset(var,bitno) ((var) |= 1 << (bitno)) // установка бит порта
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno))) // сброс бит порта
unsigned char getch(); // прием символа
void init_comms(); // инициализация коммуникации
void cmd();
void cmd_write ();
void cmd_count ();
```

Основной файл counter_start.c

```
/* ------------------------------------------------------------------------ */
/* Template source file generated by piklab */
#include <pic16f628a.h>

/* ------------------------------------------------------------------------ */
/* Configuration bits: adapt to your setup and needs */

typedef unsigned int word;
word at 0x2007 CONFIG = _WDT_OFF & _PWRTE_OFF & _INTOSC_OSC_NOCLKOUT &
_MCLRE_OFF & _BOREN_OFF & _LVP_OFF & _DATA_CP_OFF & _CP_OFF;

#include <stdio.h>
#include "counter_start.h"

unsigned char input;    // Для считывания приемного регистра
unsigned char COMMAND;  // Символ команды
unsigned char NUMBER;         //  Значение для записи
```

```
unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}

void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            cmd_write ();
            break;
            case 'b':
            PORTA = 0x0;
            break;
            default:
            break;
            }
}

void cmd_write () {

      NUMBER = getch();

      switch (NUMBER) {
            case '0':
            PORTA = 0x0;      // Все светодиоды погашены
            break;
            case '1':
            PORTA = 0x1;      // Горит первый светодиод
            break;
            case '2':
            PORTA = 0x2;      // Горит второй светодиод
            break;
            case '3':   // Горят первый и второй светодиод
            PORTA = 0x3;
            break;
            case '4':
            PORTA = 0x4;
            break;
            case '5':
            PORTA = 0x5;
            break;
            case '6':
            PORTA = 0x6;
            break;
            case '7':
            PORTA = 0x7;
            break;
            default:
            break;
            }
}

void cmd_count ()
{

}
```

```
void init_comms() // Инициализация модуля
{
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x38;
      TRISB = 0xFE;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      PORTB = 0xFE;     // Выключаем драйвер RS485 на передачу
}

void main(void) {
      init_comms(); // Инициализация модуля

start:      CREN =1; // Начинаем работать
      input = getch();
      cmd();
      goto start;
}
```

Компилируем этот проект, загружаем программу в микроконтроллер. Я вновь использую готовую программу для работы с COM-портом. Отправляю символ «a», он проходит, но на макетной плате нет видимых изменений, как и должно быть, впрочем. Я отправляю символ «7», все три светодиода зажигаются. Я отправляю символ «b», все светодиоды гаснут. На всякий случай я повторяю эти операции заменив 7 на 1. Все работает правильно. Правильно работает и отправка строк вида «a7», «a1» или «a0».

Возвращаюсь к интерфейсной части на Gambas, но добавляя и перемещая разные строки, закрывая их символами комментария, я добиваюсь только одного – мой проект превращается в мешанину строк, в которых я сам уже перестаю что-либо понимать.

Самое разумное в этот момент, насколько я понимаю, удалить все лишнее и упростить все до предела. В первую очередь я хочу удалить клавишу «Управление портом». Все, что нужно сделать для управления работой порта, можно сделать при нажатии двух оставшихся клавиш. Зачем усложнять себе жизнь? В итоге программа приобретает вид:

```
' Gambas class file

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()

END

PUBLIC SUB Button1_Click() ' Клавиша «Запись»
      SerialPort1.Open
      SerialPort1.RTS = FALSE
      PRINT #SerialPort1 TextBox1.Text
      SerialPort1.Close
END

PUBLIC SUB Button2_Click() ' Клавиша «Счет»
      PRINT #SerialPort1 "b"
END
```

Нужную мне строку вида «a7» я вписываю в окошко перед нажатием клавиши «Запись», и

теперь эта часть программы работает, зажигая на макетной плате соответствующее команде количество светодиодов. Можно вернуться к считыванию состояния тех выводов контроллера, которые (как будто) подключены к выходам счетчика.

Вновь вернемся в программу Piklab, где произведем должные изменения в коде. Но прежде, чем это сделать, проверим одну часть кода, которая отвечает за команды:

```
void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            cmd_write (); // Выполнить принятую команду
            break;
            case 'b':
            PORTA = 0x0; // Погасить светодиоды
            break;
            default:
            break;
            }
}
```

То есть, отправляя команду «a7», я зажигаю все светодиоды, а после команды «b», которую я отправляю клавишей «Счет» интерфейса, они должны погаснуть.

Небольшие изменения в коде программы интерфейса (клавишу управления портом я удалил):

```
PUBLIC SUB Button2_Click() ' Клавиша "Счет"
      SerialPort1.Open
      SerialPort1.RTS = FALSE
      PRINT #SerialPort1 "b"
      SerialPort1.Close
END
```

Эта часть программы работает – ввод команды и нажатие клавиши «Запись» зажигает все три светодиода, нажатие клавиши «Счет» их гасит.

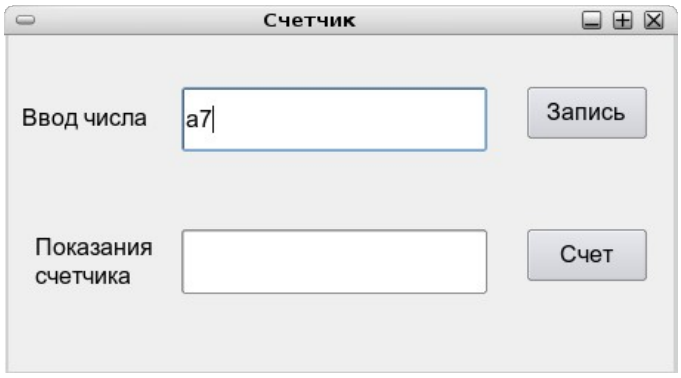


Рис. 5.1. Изменения в интерфейсной части проекта

Теперь можно попытать счастье с добавлением передачи состояния вводов.

Попытать можно, но будет ли оно? Пока ничего кроме «чудес в решете» я не наблюдаю.

Во-первых, при подключении одной программы к одному COM-порту, а программатора ко второму, одновременно, программатор дает сбои. Хорошо, подключаемся по очереди, получается лучше. Но, стоит мне добавить в текст программы для микроконтроллера описание функции, которая передавала бы состояние порта B, как контроллер перестает работать. Код, с моей точки зрения, не содержит ничего особенного:

```
void cmd_count ()
{
      CREN =0; // Выключаем прием
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      COUNT = '7'; //PORTB;
      putch(COUNT);
      PORTB = 0xFE;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      CREN =1; // Включаем прием
}
```

И самое главное, я пока не использую эту функцию в программе. А если я удаляю ее, то контроллер, по крайней мере, правильно реагирует на команды. Если бы не видел это своими глазами (или руками, как угодно), то не поверил бы. В чем же фокус? Приходит в голову мысль о неудачных именах, выбранных мною для функций. Я исправляю эти имена, но это не дает успешного продвижения вперед. Все тот же «кошмарный сон».

Раздраженный неудачей, сердитый на себя, и на весь мир заодно, я возвращаю текст программы к прежнему виду, но злополучную функцию переношу в самый конец кода, ниже функции main (), что в нормальном состоянии не сделал бы. Однако теперь контроллер выполняет команды, которые я от него ожидаю. Еще один вариант – размещение функции передачи состояния порта cmd_count() прямо над функцией main(). Опять программирование контроллера, и вновь удачно. Контроллер выполняет команды управления светодиодами.

Если вы что-то в этом понимаете, то я рад за вас, лично я в этом ничего не понимаю! Теперь пора бы заняться работой функции передачи состояния порта. На что и уходит достаточно много времени. Я уже не пытаюсь сразу передать от микроконтроллера состояние порта, я хочу получить ответ на команду в виде символа, например, «7». Выглядит эта часть программы микроконтроллера следующим образом:

```
void cmd() // Получение и выполнение команды
{
switch (input) {
      case 'a':
      PORTA = 0x0; // Погасить светодиоды
      break;
      case 'b':
      CREN = 0; // Выключаем прием
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch('7');
      PORTB = 0xFE;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      CREN = 1; // Включаем прием
      PORTA = 0x7; // Включаем все светодиоды
      break;
      default:
      break;
      }
}
```

Моя самоуверенность и ожидание быстрого успеха давно покинули меня, уступив место упрямству и раздражению, причина которого в том, что с этой программой микроконтроллера, в ее более сложном виде, я работал, и работал контроллер, и все работало. Но не в этот раз! Я даже не пытаюсь получить результат в интерфейсной части проекта, а использую программу для работы с COM-портом. Не очень удобный вариант, поскольку приходится запускать ее с помощью Wine – эмулятора Windows. Но ей я пока доверяю больше, чем своим творениям. Я ожидаю появления семерки в окне отображения вывода после отправки команды, привязанной к символу «b». Я упрощаю программу микроконтроллера до предела, удалив все, что не относится к приему и выполнению в виде отсылки символа «7» этой команды.

Я еще раз просматриваю текст старой, некогда работавшей программы, которую, очень сильно упростив я использовал. Замечаю, что в начальный момент в функции переключения при отправки ответа я использовал задержку перед выключением передатчика. В ходе экспериментов я убрал этот фрагмент. Восстанавливаю его.

```
switch (input) {
      ......
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch('7');
      for (i=0;i<1000;i++); // Пауза
      PORTB = 0xFE;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      ......
      }
}
```

Пытаюсь понять работу функции, работающей с передатчиком USART, putch(byte). Возвращаюсь к чтению документации на контроллер PIC16F628A в ее части работы с передатчиком. Пока читаешь, все понятно. Стоит задуматься о прочитанном, ничего не понятно. Например, как работает флаг прерывания, напомню, как выглядит функция передачи символа,

```
void putch(unsigned char byte)
{
      while(!TXIF) // Отправка байта
            continue;
      TXREG = byte;
}
```

которую я не придумал, а честно «срисовал» когда-то; флаг прерывания TXIF, который устанавливается и снимается, даже если прерывание не используется, должен реагировать, по описанию, на появление отправляемого символа в регистре TXREG. Но я записываю туда что-то после опроса состояния флага. Добавляет к моему замешательству и то, что я, насколько это понимаю, «кручусь» в цикле while до тех пор, пока флаг сброшен, но он появляется только после переписи символа из программно доступного мне регистра TXREG во внутренний регистр передатчика по фронту одного из импульсов, кажется стопового, предыдущего отправленного символа. Мне это все понятно в середине процесса, который, как некогда было модно говорить, «процесс пошел», существует, но мне не понятно, как он начинается, если отправляется первый символ.

Я совсем запутался. Я пытаюсь тупо переделать функцию putch(), удаляю и добавляю команды светодиодной индикации, чтобы понять, где застревает работа программы. Я

бросаю все. У меня нет больше сил воевать с «ветряными мельницами». Хотя два полезных проблеска в этой мгле имели место: я понимаю, что скрытые от меня процессы, как перепись во внутренние регистры, никаким осциллографом не «отловишь», то есть, основная идея использовать компьютер правильна; и второе – при работе с программой COM-порта я вспоминаю, что есть режим отображения в окне вывода в формате символов ASCII, что я использую, а можно отображать шестнадцатеричные значения, что я делаю. К моему удивлению после каждого цикла приема-передачи символа следует значение «00». А интерфейсная программа в моих попытках принять что-то от контроллера донимала меня «концом файла». Москит, который появляется при запуске Gambas, изображал ужас, затем крайнее отчаяние и показывал сообщение «Конец файла!». Я пытался этот конец файла куда-то пристраивать, но безнадежно, а теперь понимаю, что и пристраивать его не следовало, поскольку это «00» и есть, видимо, конец файла.

И еще одну полезную вещь я нахожу, когда, возвращаясь от безнадежности к интерфейсу на Gambas, в его многочисленных примерах есть пример работы с последовательным портом, который оказывается хорошей альтернативой программе для Windows. По этому случаю я решаю не бросать все, а еще немного помучиться.

Видимо, чтобы придать моим мучениям некоторую остроту, начинает давать сбои программатор. Он исправно работал все это время, а теперь дает ошибки записи в контроллер. Но у меня есть от этого средство. Не зря же я покупал программатор. Им пользоваться не так удобно, как самодельным, но у него автономное питание, есть некоторые дополнительные улучшения, на крайний случай есть программа, устойчиво работающая с ним. Меняю программатор, попутно проверяя очень меня удививший факт – появление ошибок при одновременной работе программы программатора и программы работы с COM-портом (на разных портах!). Ошибка остается. Но если выключить вторую программу, то программирование проходит успешно, при этом я пока использую программирование в программе Piklab, а не в отдельной программе. И на том «Спасибо!».

И спасибо моему упрямству. Я нахожу, как заставить микроконтроллер отправить нужную мне семерку. Оказалось, что достаточно увеличить паузу между вызовом функции отправки символа putch() и выключением передатчика.

```
switch (input) {
      ......
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch('7');
      for (i=0;i<5000;i++); // Пауза
      PORTB = 0xFE;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      ......
      }
}
```

Долгожданный ответ я вижу в окне вывода.

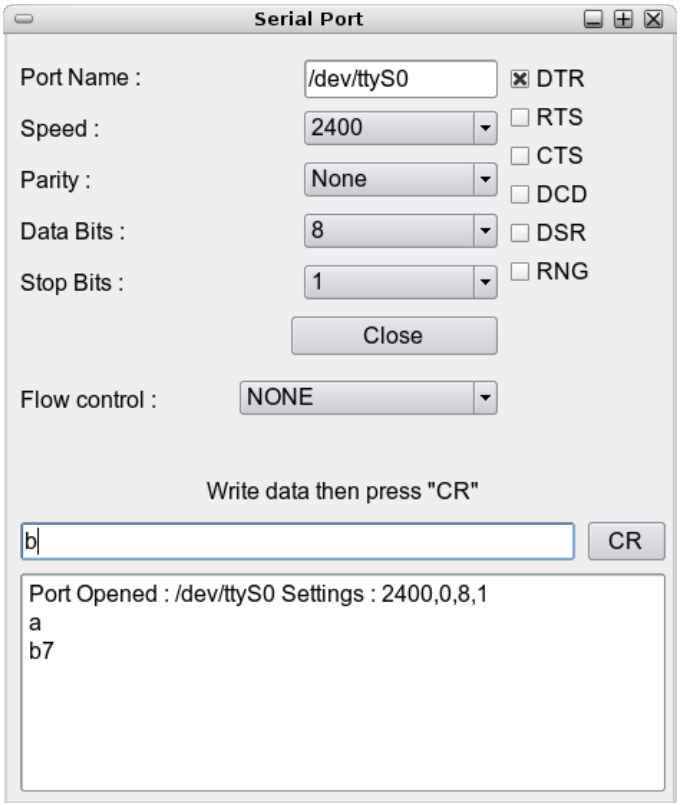


Рис. 5.2. Проверка с помощью примера из пакета системы Gambas

Долгожданный ответ скрыт в символах «b7».

Расширение кругов (на воде?)

Я не собирался доводить развитие проекта, о котором шла речь выше, даже до этой стадии. Все казалось достаточно очевидным, я не ждал каких-то подвохов, программа для микроконтроллера была взята из уже проверенного варианта, но теперь, когда все оказалось вопреки ожидаемому столь запутано, есть смысл довести работу с проектом до некоего логического завершения.

Начну я интерфейсной части проекта. Поскольку в терминале ответ от микроконтроллера виден, он должен быть виден и в соответствующем окне интерфейса. Часть кода, написанного мною в Gambas, которая отвечает за получение ответа от контроллера, выглядит так:

```
PUBLIC SUB Button2_Click() ' Клавиша "Счет"
      SerialPort1.Open
      SerialPort1.RTS = FALSE
      PRINT #SerialPort1, "b"
      INPUT #SerialPort1, TextBox2.Text, Eof(SerialPort1)
END
```

Результат безуспешных проб, этот код мне кажется правильным, но работать он не желает. Проведя ряд экспериментов по добавлению настроек порта непосредственно в текст программы, по добавлению разных переменных, я наконец нахожу разумным (и почему не сразу?) заглянуть в работающий с последовательным портом пример, который я использовал для предыдущей проверки, и в котором получил долгожданный ответ, как

последовательность символов «b7».

По виду коды в своей сути совпадают, хотя программы разные, и коды, соответственно, тоже. Проходит несколько часов заполненных фантазиями и разочарованием, прежде чем я замечаю, что получение ответа от COM-порта выполнено в работающем коде иначе, чем у меня. Я исправляю это место, копируя с последующим исправление нужную часть кода, и теперь работает и интерфейс.

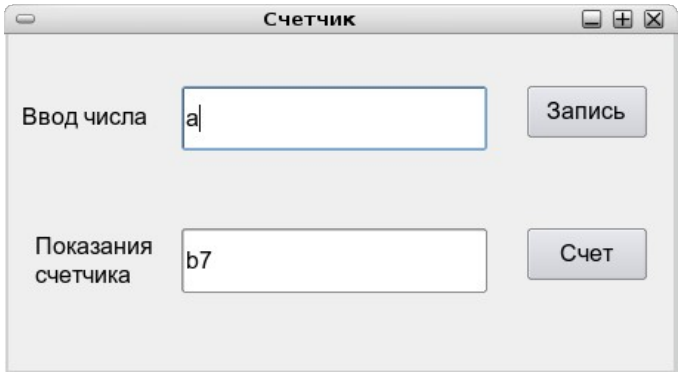


Рис. 5.3. Отклик контроллера в интерфейсной части программы

Часть кода, ответственная за это выглядит так:

```
PUBLIC SUB Button2_Click() ' Клавиша "Счет"
      SerialPort1.Open
      SerialPort1.RTS = FALSE
      PRINT #SerialPort1, "b"
END

PUBLIC SUB SerialPort1_Read()
      DIM answ AS String
      READ #SerialPort1, answ, Lof(SerialPort1)
      TextBox2.Text = TextBox2.Text & answ
      SerialPort1.Close
END
```

Здесь получение отклика от микроконтроллера интерфейсом вынесено как событие. Открывание и закрывание порта я поделил между двумя процедурами, что неправильно, но это легко поправить, хотя бы возвращаясь к первоначальному решению – добавить кнопку открывания и закрывания порта. Мне же следует решить задачу опроса порта ввода контроллера и передачу не просто символа, а этого состояния порта в виде символа. То есть, пора возвращаться к программе Piklab и программатору. Я сейчас не доверяю самодельному программатору и буду использовать купленный ExtraPic, не дорогой, но хорошо работающий (пока). И проверку я буду проводить, видимо, используя не интерфейс, а программу работы с COM-портом из примеров системы Gambas.

Не буду спешить. Для начала переделаю часть программы, отвечающую на команду «b»:

```
void cmd() // Получение и выполнение команды
{
      switch (input) {
            case 'a':
            PORTA = 0x0;
            break;
```

```
            case 'b':
      COUNT = PORTA & 0x38;
      if (COUNT == 0x38) PORTA = 0x7; // Индикация тестовая
            break;
            default:
            break;
      }
}
```

COUNT – это переменная типа unsigned char, добавленная мною в программу давно, но до сих пор не используемая. Теперь я хочу использовать ее для того, чтобы понять, что я получаю при чтении в эту переменную именно состояния выводов порта А, которые буду читать впоследствии?

Светодиоды загораются. Попробуем отправить эту переменную через COM-порт. Шестнадцатеричное число 38 соответствует символу «8» кода ASCII. В первом эксперименте это число я и отправлю.

```
switch (input) {
      ......
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch(0x38);
      for (i=0;i<5000;i++); // Пауза
      PORTB = 0xFE;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      ......
      }
}
```

Эта посылка доходит до адресата. Теперь пойдем дальше. Шаг я делаю совсем маленький – добавляю присвоение переменной состояния порта с маской:

```
COUNT = PORTA & 0x38;
```

Все. Нет реакции ни на команду «a», которая ничего не должна делать, только погасить все светодиоды (должна бы), ни на команду «b», которая должна прислать посылку и зажечь все светодиоды. Ни одна из команд не работает. И этого я вновь не понимаю. И не понимаю долго.

Чтобы отвлечься от грустных размышлений о том, как плохо не знать языков, я делаю попытку понять, что не так с самодельным программатором, который не так давно исправно работал. Программа Piklab, как мне кажется, стала вести себя несколько иначе после обновления. Проверить мне это легко, достаточно перезагрузить компьютер и войти в Ubuntu, где программа не обновлялась. И впрямь, когда я запускаю программу в Ubuntu, компилирую текст, а затем записываю его в контроллер, используя самодельный программатор, то ошибок не возникает. Мало того, запись идет гораздо быстрее, а в окне вывода мало сообщений, тогда как в Fedora 7 эти сообщения при записи кода мелькают непрерывно. Я не исключаю, впрочем, случайностей – когда что-то работает неустойчиво, несколько попыток не дают верного результата. А поскольку я перезагрузил компьютер, я возвращаюсь к проблеме не работающей программы. Для продвижения вперед я удаляю всю программу, оставляя только строки вида:

```
COUNT = PORTA;
COUNT = COUNT << 2; // Сдвиг на два бита влево
```

```
COUNT = COUNT >> 5; // Сдвиг на пять бит вправо
```

Такими операциями я хочу выделить вводы RA3-RA5 с целью их использования, когда и если мне удастся решить проблемы. С помощью симулятора gpsim я отслеживаю значения порта и переменной COUNT, и вижу, что операции побитового сдвига не выполняются. А это приводит меня к мысли, что дело-то может быть не в моих плохих знаниях, а в том, что используемый мной компилятор SDCC, программирование PIC-контроллеров у которого находится в стадии разработки, не всегда ведет себя корректно. Чтобы проверить это, я загружаю с сайта HI-TECH демо-версию их компилятора PICC. Слегка споткнувшись на то, что полученному мною файлу picc-demo.run в свойствах следует включить опцию «Позволять выполнение файла как программы» на вкладке «Права», я устанавливаю компилятор, перенастраиваю программатор на работу с этим компилятором, поправляю прежний текст (вместо #include <pic16f628a.h> следует записать #include <pic16f62xa.h>, а еще лучше этого не писать а добавить #include <htc.h>), удалив строку настройки конфигурации по адресу 2007h. Код программы компилируется, а программа работает.

С элегантной записью слова конфигурации по адресу 2007h я пока не знаю что сделать, по причине чего использую «обходной маневр» – записываю программу в контроллер из Piklab, прочитываю ее, а затем поправляю слово конфигурации в окошке обозначенном адресом «2007», куда записываю 2118. Результирующую программу я записываю еще раз. Долго, но работает. Кстати, и самодельный программатор опять работает без ошибок, что может быть случайностью.

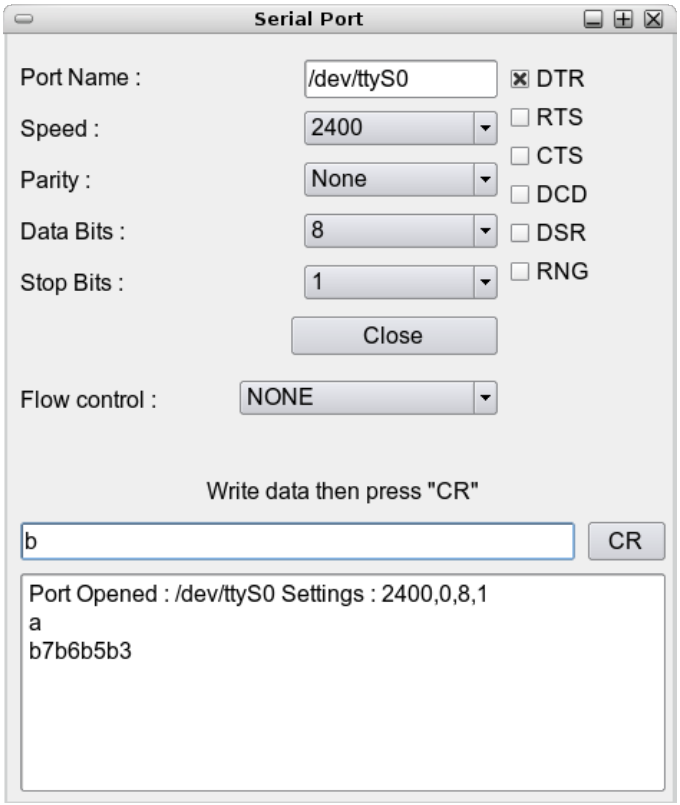


Рис. 5.4. Работающая программа после смены компилятора

Я даже могу себе позволить замыкая выводы на землю получить несколько значений – b6, b5 и b3. В этом есть своя прелесть, не так уж я неправ, но есть и недостаток, связанный с тем, что компилятор HI-TECH не генерирует файл .cod, который нужен gpsim для полноценной симуляции. Кроме того, компилятор в полной версии проработает 45 дней, потом его

придется сменить на бесплатную lite версию, в которой нет поддержки PIC16F628A, в которой я работаю с микроконтроллером как PIC16F627A, но это меньшее из зол. Важнее было выявить причину столь странного поведения программы, когда операция присваивания переменной, пусть даже значения порта, полностью выводила программу из строя. Мне кажется, что к тому моменту, когда я закончу писать эту книгу (если закончу), компилятор SDCC будет в том виде, когда у него не будет таких чудачеств.

Моя попытка удалить Piklab, установив его версию для Fedora Core 6, не приносит успеха – обе версии теперь работают одинаково, отлично от той, что есть в Ubuntu. Остается проверить, будет ли и как работать программа с интерфейсной частью. Еще раз для верности запускаю программу Piklab, «кланяюсь» ей, как и положено трижды (программирую микроконтроллер), чтобы запустив интерфейсную часть, получить результат:

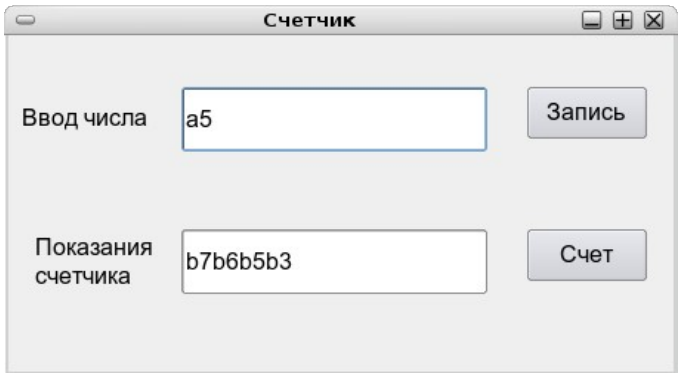


Рис. 5.5. Результат работы интерфейсной части по считыванию порта

Как и прежде я после первого получения кода «b7» последовательно замыкаю выводы RA3-RA5 на землю, чтобы получить другие значения состояния порта.

Подача управляющих сигналов к настоящему моменту мне кажется чисто технической задачей при написании кода. Чуть подольше, возможно, придется повозиться с организацией временных интервалов, но можно использовать встроенные таймеры, считывание результата работы счетчика тоже не представляется трудным. Будет ли это работать? Хороший вопрос, ответ на который может дать только прямой эксперимент – довести кодирование программ до конца, и поработать с полученной программой и «живым» счетчиком. Мне же не дает покоя то, как работает симулятор gpsim. Попробую потратить немного (если будет так) времени, чтобы разобраться с ним. Попутно хорошо бы понять, как, работая с компилятором HI-TECH, добиться симуляции схемы с помощью gpsim.

gpsim как зерцало грешника

Жаль, что не сложились отношения с компилятором SDCC, но что есть, то и есть, из этого исходить и приходится.

Самый простой вариант перехода к симуляции, но очень долгий, это из ассемблерной программы, даваемой компилятором, генерировать полный набор с помощью gpasm, который есть в программе Piklab. Но чистить и подгонять файл это долго. Здесь есть над чем задуматься.

Вторая проблема – проблема симуляции. Что-то не так получается с симуляцией. Этим, пожалуй, можно заняться сейчас. Программа gpsim приходит с несколькими примерами программ, из которых меня в первую очередь интересует программа usart_gui. Есть

ассемблерный файл, есть файл .stc, файл задания стимулов. Насколько я понимаю без файла usart_gui.cod симуляция не должна работать. А для получения этого файла я создаю папку на рабочем столе, копирую папку examples из набора gpsim, в которой создаю папку с громким названием test. В эту папку копирую оба файла из папки примеров: usart_gui.asm usart и_gui.stc. Содержимое папки в данный момент выглядит так:

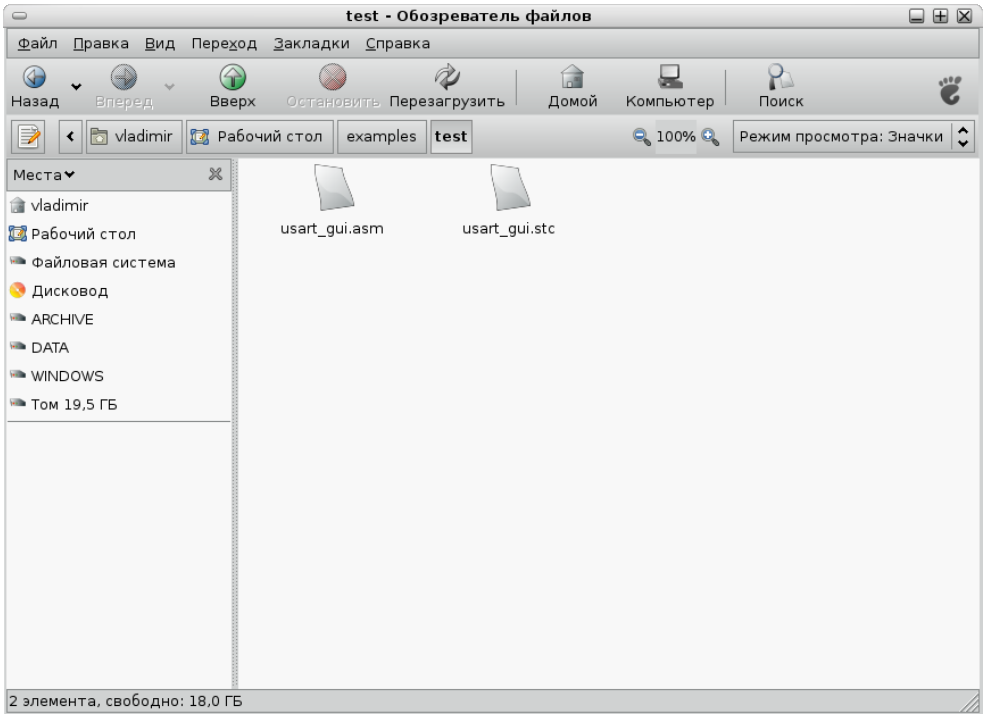


Рис. 5.6. Папка с исходными файлами из примеров gpsim

Теперь можно запустить программу Piklab, создать новый проект, который я назову также usart_gui. Для работы с ассемблерным файлом в разделе Toolchain следует выбрать GPUtils. Далее выбрать опцию добавления в качестве исходного файла уже существующий, в качестве которого на следующем шаге создания проекта выбрать usart_gui.asm, что выполняется нажатием на клавишу **Add**, с последующим указанием места расположения файла:

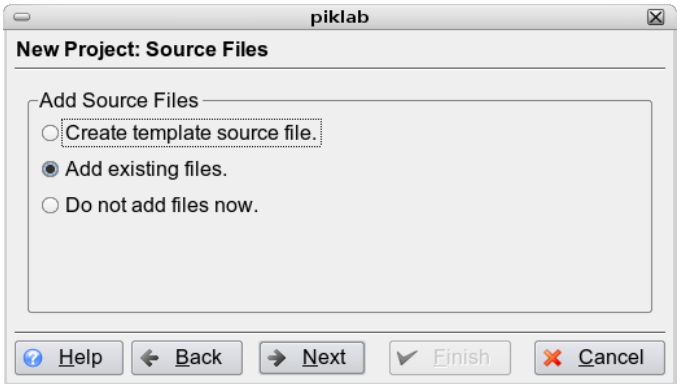


Рис. 5.7. Создание тестового проекта в Piklab

Файл готов к работе с ним, можно запустить компиляцию с помощью пункта **Build Project** раздела **Build** основного меню программы Piklab. Компиляция проходит быстро, а папка test пополняется рядом полезных файлов:

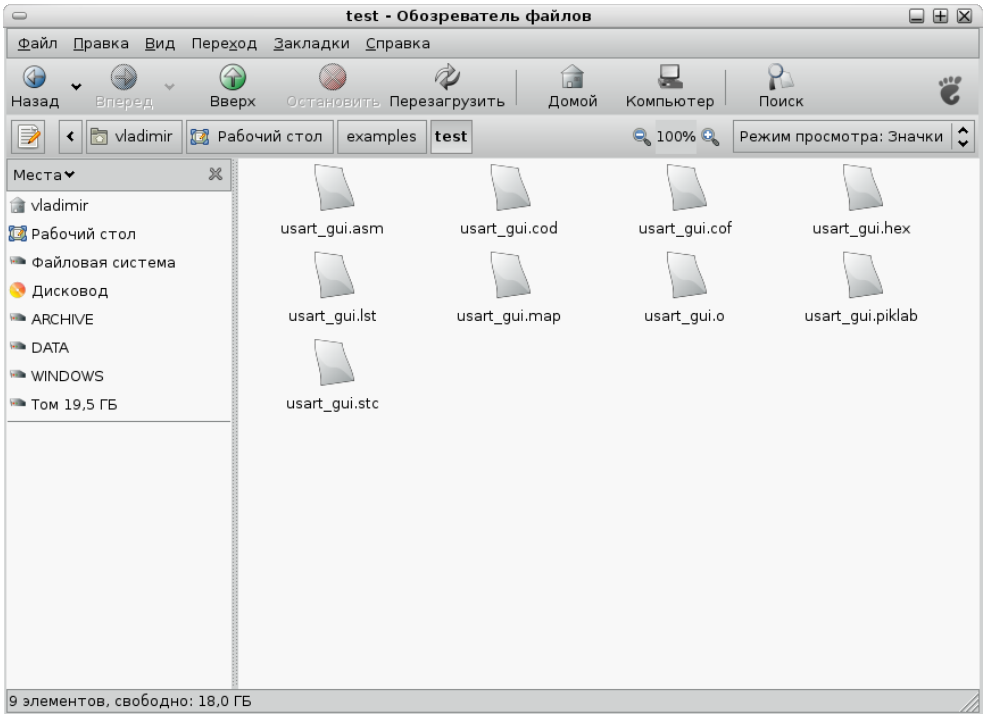


Рис. 5.8. Вид папки проекта после компиляции исходного текста

То, что я так подробно описываю эти рутинные операции, свидетельствует только о мере моей растерянности и озабоченности. Которые вызваны не столько тем, что что-то не получается, редко когда бывает иначе, сколько тем, что я могу неправильно истолковать результаты своих опытов, приписав инструментальному средству качества, которыми отличаются мои неумелые руки. Например, после получения всех необходимых файлов я загружаю в gpsim файл usart_gui.cod, полученный в результате компиляции, следом файл usart_gui.stc. И... и ничего хорошего. Оказывается достаточно загрузить файл usart_gui.stc, чтобы можно было запустить симуляцию. Все необходимое, включая настройку узлов, получается как по мановению волшебной палочки. Запуск симуляции с помощью клавиши Run сразу выводит сообщение в окно USART, а когда я ввожу символы с клавиатуры, я вижу их вывод в том же окне.

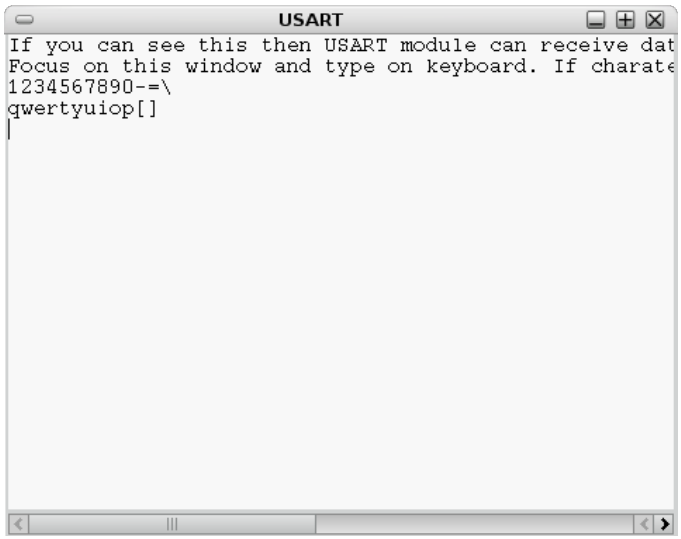


Рис. 5.9. Работа примера симуляции с модулем usart

У меня вывод символов с клавиатуры, практически, не получался. И только одно утешает меня – если остановить симуляцию, нажать на клавишу reset программы gpsim, а затем вновь запустить симуляцию, я получаю тот же результат, что получал и раньше, то есть, опять ничего не работает. Но это не мешает мне несколько раз повторить удачную находку с загрузкой файлов .stc из предыдущих проектов. Чтобы получить этот файл, я использую клавишу Save Cofiguration... на макетной плате Breadboard. Клавиша срабатывает не сразу, но когда срабатывает, то предлагает сохранить файл с тем именем, который выбираешь. В руководстве к программе gpsim упоминалось, что стимулы лучше получать из файла, но как это сделать я не понял. Теперь есть возможность избежать бесконечных повторов с вводом библиотеки, с заданием и привязкой узлов, и т.д.

Однако радость преждевременна. Файл, который я получаю с помощью клавиши сохранения конфигурации, если не ввести новое имя, называется netlist.stc, и загружает только часть необходимого. Используя в качестве подсказки работающий файл из примера и используя метод проб и ошибок, я нахожу, что следует изменить строку в начале файла netlist.stc:

```
# load s mycod.cod
```

Убрав комментарий, заменить шаблон mycod на имя реального файла, который у меня назван:

```
load s counter_start.cod
```

Кроме этого, двигаясь вниз по файлу (после некоторого потраченного времени), я поднимаю текст загрузки модуля usart над параметрами контроллера:

```
# Module libraries:

module library libgpsim_modules.so

# Modules:
module load usart U1

U1.rxbaud=9600
U1.txbaud=9600
U1.rx=0
U1.tx=0
U1.crlf=true
U1.loop=false
U1.console=false

U1.xpos=72,00000000000000
U1.ypos=276,0000000000000

p16f628a.CONFIG=$ff
p16f628a.WarnMode=true
p16f628a.SafeMode=true
p16f628a.UnknownMode=true
p16f628a.BreakOnReset=true
p16f628a.BreakOnInvalidRegisterRead=true
p16f628a.BreakOnInvalidRegisterWrite=true
p16f628a.frequency=20000000,00000000
p16f628a.xpos=72,00000000000000
p16f628a.ypos=72,00000000000000
```

После этой операции все начинает загружаться правильно, включая то, что оба узла, node

rb1 и node rb2, которые я создал для соединения usart с контроллером, оба узла определяются и появляются правильным образом. Но чуть позже выясняется, что формат чисел, определяемый системой, не вполне согласуется с программным. Я имею ввиду запятую в десятичных числах. Ее следует поменять на точку. Например, в последней строке:

```
p16f628a.ypos=72.00000000000000
```

Несколько программ, приведенных в примерах, показывают как работает симулятор, и сколько интересных возможностей скрывают дополнительные модули.

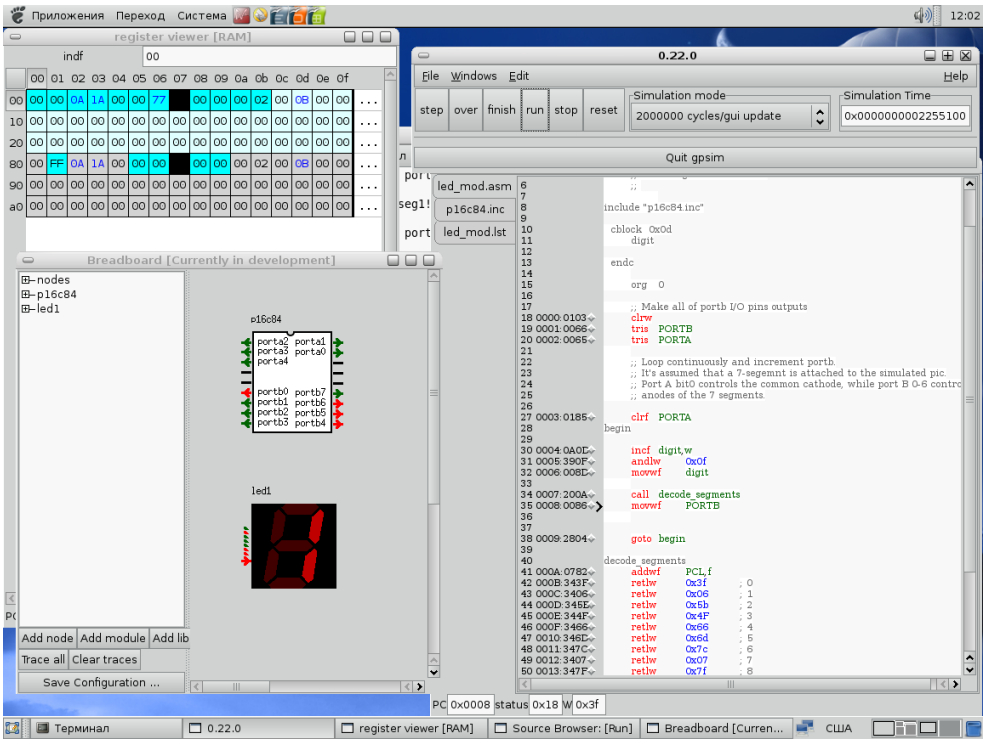


Рис. 5.10. Работа симулятора gpsim с семисегментным индикатором

Что можно сказать по предварительным результатам:

1. Не возникает проблем при работе в связке Piklab-gpsim, если использовать ассемблер.
2. Есть некоторые проблемы при использовании компилятора «С» SDCC, связанные, скорее всего, с самим компилятором.
3. Компилятор HI-TECH, полная и свободная версия, не дает при компиляции нужного файла формата .cod.

Что касается ассемблера, с профессиональной точки зрения это наилучший вариант, позволяющий полностью контролировать код программы, но вариант наиболее трудоемкий и долгий. Написание программного кода на ассемблере – предмет отдельного, я бы сказал, увлечения. Это увлечение вполне доступно любому, а по мере накопления опыта и код станет более изящным, и количество готовых программных модулей увеличится, хотя бы за счет разбиения программ из примеров на отдельные работающие модули. Словом, есть над чем подумать, поскольку в этом виде все работает, и работает хорошо.

С языком «С» при использовании компилятора SDCC, похоже, следует разобраться, почитав документацию к этому компилятору. Использование же компилятора HI-TECH

позволяет обойти проблему отсутствующего формата следующим образом (мне не очень понятным). В документации к компилятору говорится о том, что есть опция –OUTPUT=cod, которая должна выводить требуемый файл. Для использования этой опции перед компиляцией проекта в Piklab следует войти в пункт **Ptoject Options...** раздела **Project** основного меню, и выбрать режим работы не автоматический, а пользовательский.

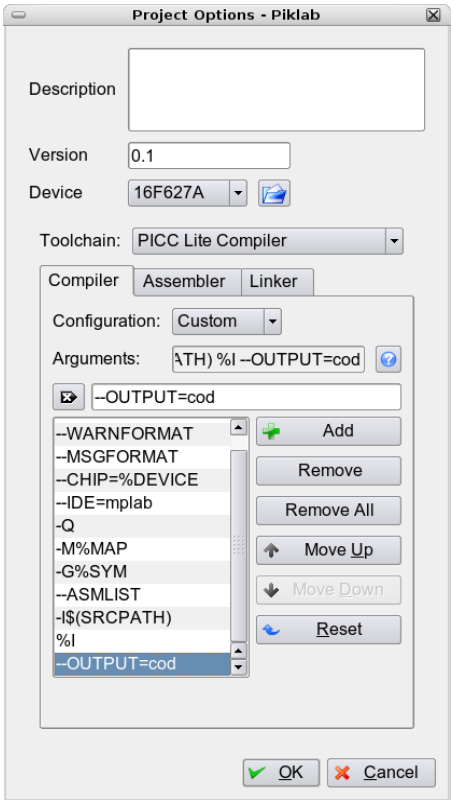


Рис. 5.11. Выбор пользовательского режима трансляции программы

Если в нижнем окне ничего не выделять, то в окне Arguments: можно ввести нужную опцию, а нажав на клавишу **Add**, добавить ее к пользовательским опциям. В этом месте я немного не понимаю происходящее. Если теперь компилировать проект, то нужного файла в проекте не обнаруживается. Но! Если убрать опцию -S, которая имеет место «по умолчанию», то после трансляции проекта файл с расширением .cod появляется. Я пытался смещать эту опцию, добавлять ее на вкладке Assembler, лучшего результата, чем получилось, я не достиг. В итоге последовательность действий такова (если вы не найдете лучшего решения):

1. Установить пользовательские опции трансляции проекта.
2. Оттранслировать файл, используя **Build-Compile File**.
3. Отладить файл в отладчике gpsim.
4. Оттранслировать проект, используя автоматический режим.

При этом возникает еще одна особенность. Использование PICC Lite работает довольно устойчиво, а полная версия (демо-версия), иногда отладчиком воспринимается, но чаще после начала загрузки дает сбой и исчезает с сообщением, что все у меня плохо. При работающем варианте, что мне кажется удобнее, gpsim показывает код текста программы, написанный на «С».

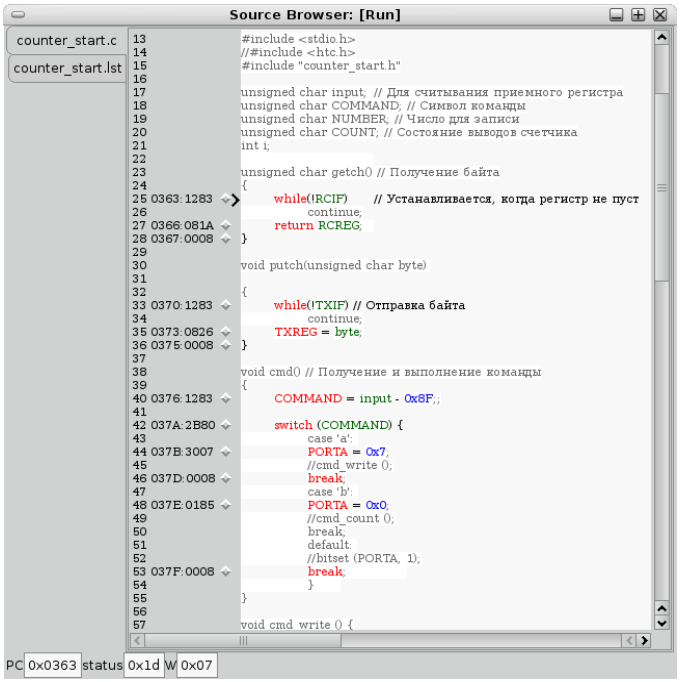


Рис. 5.12. Работа gpsim при симуляции cod-файла компилятора HI-TECH

После отладки программы можно выбрать автоматический режим компиляции в опциях проекта, еще раз оттранслировать его, выбрав **Build-Build Project**, и загрузить полученный hex-файл в контроллер.

В настоящий момент для меня остается не решенной проблема ввода символов с клавиатуры. Если бы не корректная работа программы из набора примеров gpsim, я готов бы был отмахнуться от этой проблемы. Но пример работает корректно. Правда, после рестарта отображение в окне USART напоминает то, что получаю я, но это после рестарта.

Из предположений, которые я отнес бы к разумным, есть два. Первое касается локали, используемой системой, но оно слабое, поскольку пример из набора пакета gpsim работает правильно. Второе, более правдоподобное, как мне кажется, может быть связано с настройками контроллера и модуля usart. И то, и другое должно быть настроено правильно. Я взял настройки старой программы, что некогда работала, но все ли в этом правильно, ведь прежде я работал в Windows и с программой MPLAB. Чтобы проверить настройки, в какой-то момент я пытался сделать при симуляции разные установки частоты контроллера, но это не привело к успеху. Однако здесь могли быть другие причины.

Попробуем иначе проверить этот момент. С целью проверки я изменю тестовую программу микроконтроллера. Я не буду принимать ввод с клавиатуры, но лишь отправлю на экран символ «7». Посмотрим, что обнаружится в регистре TXREG модуля usart, и что придет на экран USART?

Файл заголовка.

```
void putch(unsigned char); // Передача символа
void init_comms(); // Инициализация коммуникации
void cmd(); // Отправка символа «7»
```

Основной файл программы

```
/* ------------------------------------------------------------------------ */
/* Template source file generated by piklab */
```

```
#include <pic16f62xa.h>
#include <stdio.h>
#include "test1.h"
int i;

/*unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
} */

void putch(unsigned char byte)  // Отправка байта

{
      while(!TXIF) // Устанавливается, когда регистр пуст
            continue;
      TXREG = byte;
}

void cmd() // Получение и выполнение команды
{
      CREN = 0; // Выключаем прием
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch('7'); // Отправка символа «7»
      for (i=0;i<5000;i++); // Пауза
      PORTB = 0xFA;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      CREN = 1; // Включаем прием
      PORTA = 0x7;      // Тестовая индикация
}

void init_comms() // Инициализация модуля
{
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x38;
      TRISB = 0xFA;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      PORTB = 0xFE;     // Выключаем драйвер RS485 на передачу
}


void main(void) {
      init_comms(); // Инициализация модуля

start:      cmd();
      for (i=0;i<30000;i++); // Пауза
      PORTA = 0;  // Тестовая индикация
      for (i=0;i<30000;i++); // Пауза
      goto start;
}
```

В качестве компилятора я использую PICC Lite. В свойствах проекта настройки компилятора, как и выше, пользовательские – без опции -S, которую я удаляю с помощью клавиши **Remove**, и с добавленной опцией –OUTPUT=cod. Компиляция только основного файла (**Build-Compile File**). Файл test1.stc для работы с симулятором имеет вид:

```
# This file was written by gpsim.
# You can use this file for example like this:
#     gpsim -s mycode.cod -c netlist.stc

# If you want to add commands, you can create another .stc file
# and load this file from it. Something like this:
# ----------- myproject.stc ---------------

load s test1.cod

#frequency 20000000

# load c netlist.stc

# ----------------------------------------
# You can then just load this new file:
#     gpsim -c myproject.stc
# and use netlist.stc whenever you save from the breadboard.

# Processor position:

# Module libraries:

module library libgpsim_modules.so

# Modules:

module load usart U1

U1.rxbaud=2400
U1.txbaud=2400
U1.rx=0
U1.tx=0
U1.crlf=true
U1.loop=false
U1.console=true
U1.xpos=72.00000000000000
U1.ypos=276.0000000000000

p16f628a.CONFIG=$ff
p16f628a.WarnMode=true
p16f628a.SafeMode=true
p16f628a.UnknownMode=true
p16f628a.BreakOnReset=true
p16f628a.BreakOnInvalidRegisterRead=true
p16f628a.BreakOnInvalidRegisterWrite=true
p16f628a.frequency=1000000.00000000
p16f628a.xpos=72.00000000000000
p16f628a.ypos=72.00000000000000

# Connections:

node rb1
attach rb1 portb1 U1.TXPIN

node rb2
attach rb2 portb2 U1.RXPIN

# End.
```

Запуск симуляции не отображает в окне USART ничего. И, хотя в файле test1.stc частоту

контроллера я определил в 1МГц, мне приходит в голову заглянуть в его свойства на макетной плате симулятора. Не напрасно!

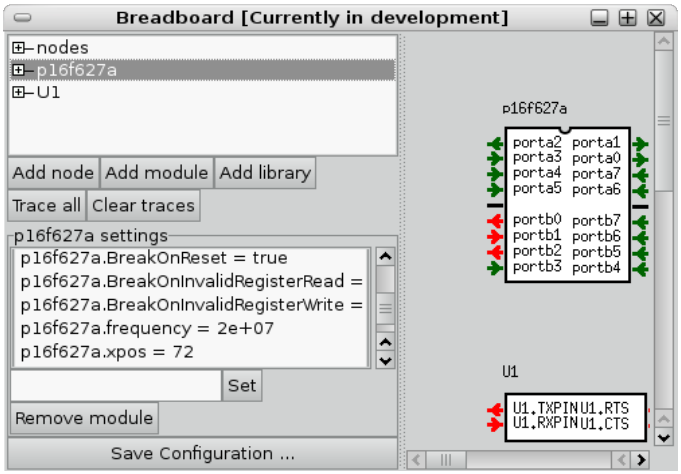


Рис. 5.13. Отображение настроек контроллера в gpsim

Вопреки моей уверенности в частоте контроллера она нисколько не изменилась против настроек «по умолчанию». Рассерженный этим обстоятельством, я удаляю весь блок настроек из stc-файла, пытаюсь «на ходу» поменять частоту, но без особого успеха, а затем, вспомнив, что в начале файла есть упоминание о частоте:

```
#frequency 20000000
```

Меняю эту строку, придав ей следующий вид:

```
frequency 1000000
```

Теперь, загрузив stc-файл в симулятор gpsim, я вижу, что частота приняла требуемое значение, что сказывается и на том, что я получаю в окне USART.

Рис. 5.14. Вид символа «7» в окне USART симулятора gpsim

Не победа, но шажок вперед. Сейчас вид окна такой, как я получал, запуская пример работы с usart, но после рестарта программы.

При этом в регистре TXREG, доступном для записи из программы регистре, который можно посмотреть в окне register viewer [RAM] симулятора по шестнадцатеричному адресу 19h , отображается правильное значение 37h.

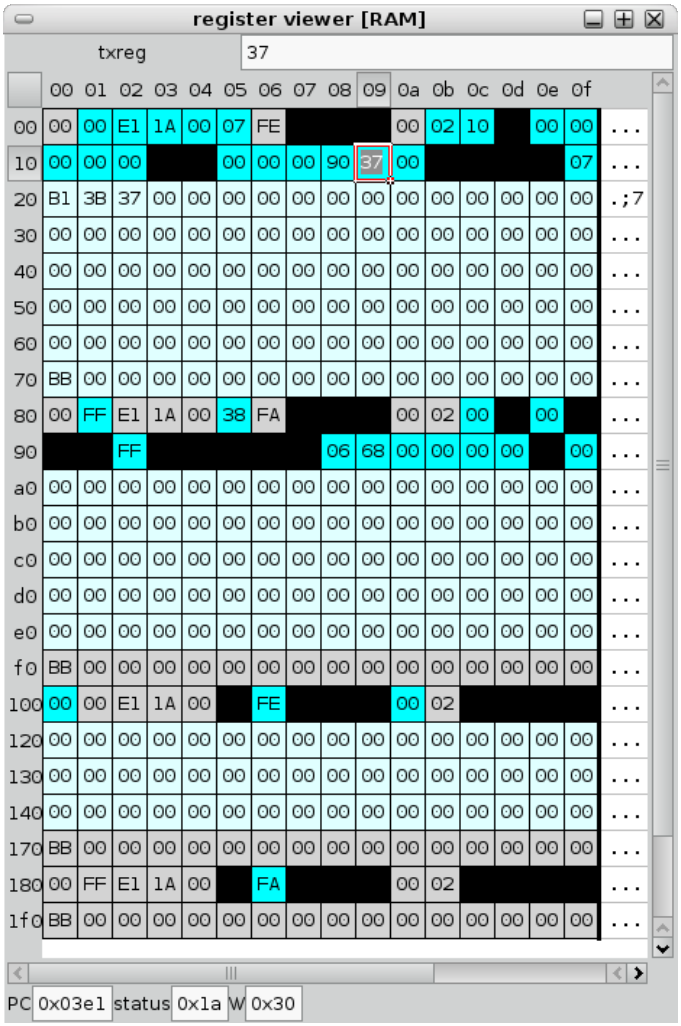


Рис. 5.15. Отображение регистров в симуляторе gpsim

Интересно, подумалось мне в этой связи, а почему я задаю частоту микроконтроллера в 1 МГц? Ответа я не знаю, тратить время на поиски старых записей не хочется, поэтому я исправляю в stc-файле частоту на 4МГц.

```
frequency 4000000
```

И столь долгожданное чудо свершилось. После запуска симулятора я наконец вижу в окне USART:

Рис. 5.16. Правильное отображение символа в окне USART симулятора

Еще не победа, но хороший шаг вперед. Пока удача на моей стороне, любопытно проверить, как оттранслирует эту программу компилятор SDCC. Удалим из папки проекта в Piklab все файлы кроме основных и stc-файла, последний не зависит от компилятора. И,

сменив в свойствах проекта (**Project Options...**) компилятор PICC Lite на Small Device C Compiller (SDCC), а контроллер на PIC16F628A, слегка подправив текст, что также сводится к замене контроллера, я запускаю компиляцию, а следом симуляцию. Как и следовало ожидать, результат не отличается от предыдущего. И даже индикаторы на breadboard, вернее выводы порта, которые я предназначил к индикации в программе, весело меняют цвет. В этом случае SDCC компилятор работает прекрасно.

А как обстоят дела с отображением ввода с клавиатуры? Для этой цели у меня предназначена программа test2. Она просто получает символ с клавиатуры и отправляет его обратно. Попробую я этот тест сразу с компилятором SDCC.

Файл заголовка

```
unsigned char getch(); // прием символа
void putch(unsigned char); // передача символа
void init_comms(); // инициализация коммуникации
void cmd();
```

Основной файл

```
/* ------------------------------------------------------------------------ */

/* Template source file generated by piklab */

#include <pic16f628.h>
#include <stdio.h>
#include "test2.h"

unsigned char input;
int i;

unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}

void putch(unsigned char byte)
{
      while(!TXIF) // Отправка байта
            continue;
      TXREG = byte;
}

void cmd() // Получение и выполнение команды
{
      CREN = 0; // Выключаем прием
      PORTB = 0xFF;  // Включаем передатчик
      TXEN = 1; // Включаем передачу
      putch(input);
      for (i=0;i<5000;i++); // Пауза
      PORTB = 0xFA;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      CREN = 1; // Включаем прием
}

void init_comms() // Инициализация модуля
{
```

```
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x38;
      TRISB = 0xFA;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      PORTB = 0xFE;     // Выключаем драйвер RS485 на передачу
      PORTA = 0x7;      // Тестовая индикация
}

void main(void) {

      init_comms(); // Инициализация модуля

start:      CREN =1; // Начинаем работать
      input = getch();
      cmd();
      goto start;
}
```

Результат радует глаз.

Рис. 5.17. Правильное отображение ввода с клавиатуры в gpsim

Наконец-то! И в этом случае компилятор SDCC работает верно.

Остается проверить программу полностью. С компилятором SDCC та же проблема, что и раньше – не отрабатывается присвоение переменной значения состояния порта, а свободная версия HI-TECH вполне, с моей точки зрения, адекватно отрабатывает программу.

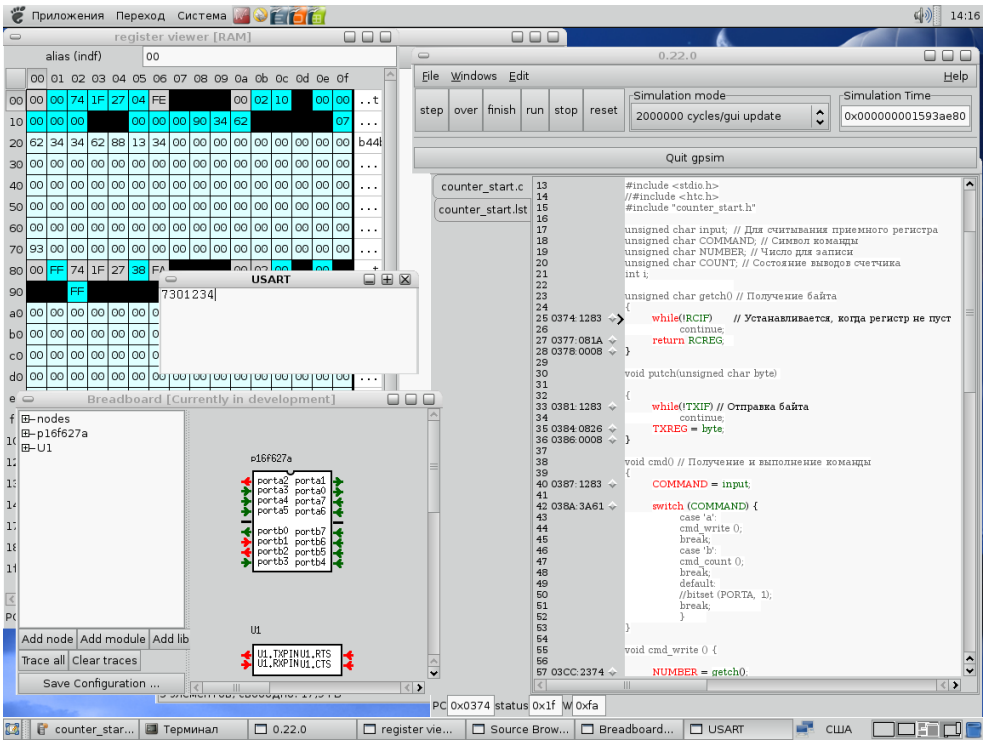


Рис. 5.18. Окончательная проверка программы в gpsim

Программа, которую я тестирую с помощью симулятора, имеет вид:

Файл заголовка

```
unsigned char getch(); // Прием символа
void putch(unsigned char); // Передача символа
void init_comms(); // Инициализация коммуникации
void cmd(); // Прием общих команд
void cmd_write (); // Прием команды «a»
void cmd_count (); // Прием команды «b»
```

Основной файл

```
/* ------------------------------------------------------------------------ */
/* Template source file generated by piklab */
#include <pic16f62xa.h>
#include <stdio.h>
#include "counter_start.h"

unsigned char input;    // Для считывания приемного регистра
unsigned char COMMAND;  // Символ команды
unsigned char NUMBER; // Число для записи
unsigned char COUNT; // Состояние выводов счетчика
int i;

unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}

void putch(unsigned char byte)
```

```
{
      while(!TXIF) // Отправка байта
            continue;
      TXREG = byte;
}

void cmd() // Получение и выполнение команды
{
      COMMAND = input;

      switch (COMMAND) {
            case 'a':
            cmd_write ();
            break;
            case 'b':
            cmd_count ();
            break;
            default:
            break;
            }
}

void cmd_write () {

      NUMBER = getch();

      switch (NUMBER) {
            case '0':
            PORTA = 0x0;
            break;
            case '1':
            PORTA = 0x1;
            break;
            case '2':
            PORTA = 0x2;
            break;
            case '3':
            PORTA = 0x3;
            break;
            case '4':
            PORTA = 0x4;
            break;
            case '5':
            PORTA = 0x5;
            break;
            case '6':
            PORTA = 0x6;
            break;
            case '7':
            PORTA = 0x7;
            break;
            default:
            break;
            }
}

void init_comms() // Инициализация модуля
{
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
```

```
	TRISA = 0x38;
	TRISB = 0xFA;
	RCSTA = 0x90;		// Настройка приемника
	TXSTA = 0x6;		// Настройка передатчика
	SPBRG = 0x68;		// Настройка режима приема-передачи
	PORTB = 0xFE;		// Выключаем драйвер RS485 на передачу
	PORTA = 0x1;		// Тестовая индикация
}

void cmd_count ()
{
	COUNT = PORTA;
	CREN = 0; // Выключаем прием
	PORTB = 0xFF;  // Включаем передатчик
	TXEN = 1; // Включаем передачу
	COUNT = COUNT + 0x30;
	putch(COUNT);
	for (i=0;i<5000;i++); // Пауза
	PORTB = 0xFA;  // Выключаем передатчик
	TXEN = 0; // Выключаем передачу
	CREN = 1; // Включаем прием
}


void main(void) {
	init_comms(); // Инициализация модуля

start:	CREN =1; // Начинаем работать
	input = getch();
	cmd();
	goto start;
}
```

stc-файл, конечно, почти совпадает по содержанию с тем, что было приведено выше, исключая строку

```
load s counter_start.cod
```

где меняется имя cod-файла на относящееся к текущей программе. В остальном все, практически, сохраняется. Включая, что после сброса компьютера, клавишей **Reset** симулятора, его частота устанавливается «по умолчанию», что мешает правильно понимать ввод-вывод.

Вялая попытка оправдаться

При последней проверке я немного слукавил. Та программа контроллера, которую я использовал совместно с интерфейсом и макетной платой (физической) отличается от тестовой версии наличием двух строк и функция отправки состояния порта А выглядит так:

```
void cmd_count ()
{
	COUNT = PORTA;
	CREN = 0; // Выключаем прием
	PORTB = 0xFF;  // Включаем передатчик
	TXEN = 1; // Включаем передачу
	COUNT = COUNT << 2;
	COUNT = COUNT >> 5;
	COUNT = COUNT + 0x30;
```

```
      putch(COUNT);
      for (i=0;i<5000;i++); // Пауза
      PORTB = 0xFA;  // Выключаем передатчик
      TXEN = 0; // Выключаем передачу
      CREN = 1; // Включаем прием
}
```

На «живой» макетной плате выводы порта соединяются с подтягивающими резисторами. При симуляции в gpsim есть возможность добавить такую «подтяжку», но мне не хотелось морочить голову ни себе, ни вам. Если использовать две удаленные строки, то порта А всегда будет обнулен, что и я и получаю при симуляции, а мне гораздо больше понравилось использовать команду «a» с символом включения индикаторов – числом от 0 до 7, которые затем и получать при чтении состояния порта. Слукавил.

Не вполне мне понятно, почему компилятор SDCC, прекрасно работающий во многих случаях, при чтении состояния порта

```
COUNT = PORTA;
```

не желает работать. Это еще одна интересная загадка. К которой добавляется тот факт, что собственно присвоение переменной состояния порта имеет место, как это показывает симуляция в gpsim. Интересно. Пока у меня только одно объяснение, функции чтения и записи в usart взяты мною из примеров для компилятора HI-TECH. Может быть здесь какой-то подводный камень?

Нет. Не здесь.

Все получилось проще (во всяком случае, сейчас). Версия, которой я пользовался – это версия 2.6. Появилась новая версия, 2.7.0. Замена версии компилятора устранила проблему. Теперь симуляция проходит точно так же, как и компилятором HI-TECH. Осталось проверить работу с «живой» макетной платой и интерфейсом, да проверить всю связку в другом дистрибутиве Linux (Ubuntu).

Все работает и с другим дистрибутивом, где я заменил версию SDCC на последнюю. Работает и с интерфейсной частью проекта. Лучше не бывает.

Хотя нет, бывает. Очень меня интересовало, почему в Ubuntu программирование из Piklab идет быстрее, а в окне программирования гораздо меньше сообщений, чем я получаю в Fedora 7, основной у меня ОС сегодня? И все оказалось очень просто – сам, как я полагаю, зацепил настройки Piklab, установив на вкладке General вывод всех сообщений об ошибках. Стоило поменять эту опцию на другую, как показано на рисунке ниже, и все стало быстро, как и в Ubuntu.

Сам виноват.

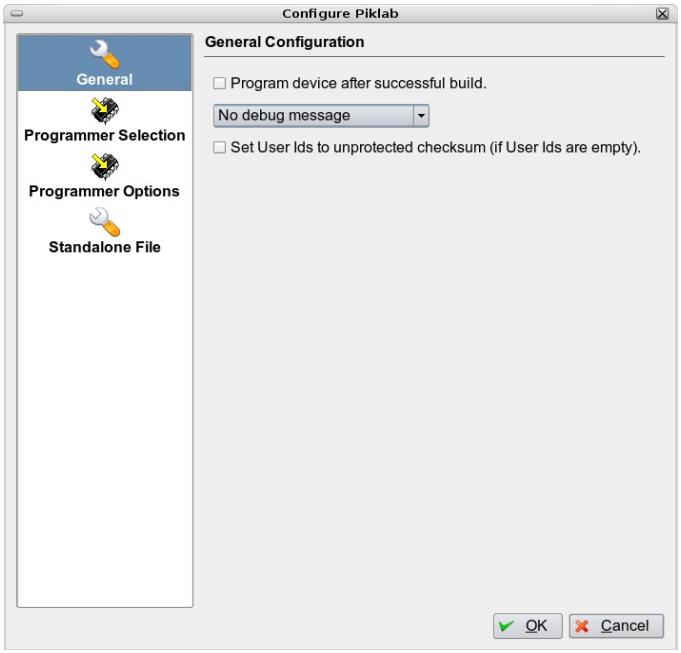


Рис. 5.19. Установка опции вывода сообщений в Piklab

И оргвыводы

Где можно применить такой подход? Я имею ввиду не борьбу «с ветряными мельницами», а использование микроконтроллера и интерфейса написанного в Gamabas. В первую очередь там, где требуется для наладки или проверки большое количество импульсных сигналов, следующих в определенной последовательности. Конечно, можно, как это делается традиционно, использовать для их получения единственный «физический» генератор импульсов, как тактовый, а затем формировать все остальное, используя схемы задержки, укорачивания и удлинения импульсов и т.д., но мне кажется, что веселый дует «компьютер и микроконтроллер» справятся с подобной задачей лучше, при небольших навыках быстрее, обойдется это дешевле, но, главное, реализация будет много интереснее.

Получив некоторый опыт в работе со всеми необходимыми программами, можно было бы написать программу, которая назвалась бы «Тестер для микросхем». Если на макетную плату рядом с панелькой для микроконтроллера поставить панельку для микросхем, если в микроконтроллер добавить несколько программ обслуживания этих микросхем, а в программу интерфейса добавить команды перехода к этим программам обслуживания микросхем, то что-то из всего этого, видимо, и получится.

Больше пользы эти шаги по созданию проверочного стенда могут принести при проверке цифровых модулей, требующих достаточно большого количества управляющих сигналов. Хотя можно найти и другие применения этому дуэту – программный интерфейс и микроконтроллер, если подумать. Или когда жизнь подскажет.

Итак. Среда программирования Gambas позволяет достаточно легко и быстро создавать интерфейсную часть, которая хорошо справляется с общением по COM-порту между вашими желаниями и возможностями микроконтроллера. Их, этих возможностей у микроконтроллера, великое множество.

Программа Piklab великолепно и без каких-либо проблем работает с программами, написанными на ассемблере (я веду речь только о том контроллере, с которым работал). Но писать программы на ассемблере есть смысл тогда, когда есть интерес к ассемблеру. В противном случае лучше воспользоваться либо компилятором PICC Lite производства HI-

TECH, или купить его полную версию, либо, что удобнее, использовать компилятор SDCC.

Программа Piklab достаточно уверенно работает с простейшим самодельным программатором JDM Classic, но еще успешнее с покупным и недорогим EXTRA-PIC.

При правильной настройке симулятор gpsim, который работает в Linux, оказывается не менее удобным отладочным средством, чем встроенный симулятор MPLAB.

Чтобы это доказать себе, а только себе и можно что-либо доказать вне рамок формальных наук, чтобы доказать это себе, я опробую еще одно средство, которое есть в gpsim, для чего быстро (как и в прошлый раз) напишу код программы test3 для контроллера PIC16F628A, код даже без файла заголовков.

```
/* ------------------------------------------------------------------------ */
/* Template source file generated by piklab */
#include <pic16f628a.h>

/* ------------------------------------------------------------------------ */
/* Configuration bits: adapt to your setup and needs */
typedef unsigned int word;
word at 0x2007 CONFIG = _WDT_ON & _PWRTE_OFF & _RC_OSC_CLKOUT & _MCLRE_ON &
_BOREN_ON & _LVP_ON & _DATA_CP_OFF & _CP_OFF;

int i;

void init_comms() // Инициализация модуля
{
      CMCON = 0x7;      // Настройка портов A и B
      TRISA = 0x38;
      TRISB = 0xFA;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      PORTB = 0xFE;     // Выключаем драйвер RS485 на передачу
      PORTA = 0x0;
}

void main() {
    /* << insert code >> */

start:      init_comms();
      PORTA = 0x0;
      for (i=0;i<100;i++);
      PORTA = 0x1;
      for (i=0;i<100;i++);
      PORTA = 0x1;
      goto start;
}
```

Если код откомпилировать, запустить gpsim, в котором открыть cod-файл, а затем открыть осциллограф (**Windows-Scope**), и щелкнуть в его правой части, где появляется окошко для ввода, куда вписать, например, pora0, а после запуска симуляции нажать Enter на клавиатуре (при этом окошко исчезает), то можно наблюдать изменение сигнала на нулевом выводе порта A:

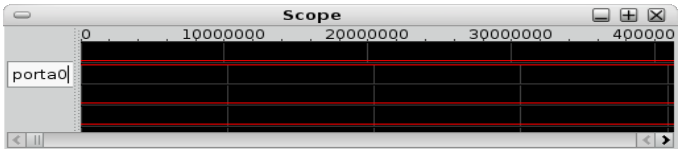


Рис. 5.20. Осциллограф в отладчике gpsim

По словам автора проекта gpsim это средство общения с микроконтроллерами нуждается в дополнительной работе, и пока только отображает логическое состояние портовых вводов-выводов, но оно уже есть. И, когда будет доработано, возможно будет не хуже того, что в MPLAB. В Linux многое меняется быстро.

Глава 6. Сказка о ловком программисте

Или о том, как с помощью программ можно решать житейские вопросы радиолюбителя.

Очень мне хотелось в этой главе позлословить о тех специалистах, которые пугают ужасными последствиями перехода с Windows на Linux, и тех, кто кормится от Windows, рассказывая, какие они белые и пушистые, что будет особенно заметно в новой версии. Но, написав половину главы, я увидел, как это все скучно и не интересно. И никакого отношения не имеет ни к электронике, ни к программированию, ни, особенно, к радиолюбительству.

Я вычеркнул все написанное, оставив только название главы себе в назидание, а тем, кто не прочь позлословить, в качестве «белого дракона на белом полотне».

За время написания этой книги я многократно обращался к книгам по программированию, написанным хорошими профессионалами. Листая книги, я находил много интересного и полезного, но для программистов. Это касается и работы с графикой, и расчетных задач, и всего того, что составляет предмет интереса и забот программиста. Но какое это имеет отношение к электронике или радиолюбительству?

С другой стороны, мне знакомо это чувство огорчения и неудовлетворенности, которое испытываешь, когда, найдя неисправность в полезном и, порой, весьма дорогостоящем электронном изделии, понимаешь, что следует заменить контроллер, но найти файл для загрузки в этот контроллер будет невозможно. Порой вся схема прибора или устройства – это микроконтроллер. В профессиональной практике ремонт подобной электроники сводится не к замене контроллера, а замене платы на новую, даже если это единственная плата. И самое ценное на этой плате – программа, «зашитая» в контроллер.

Любитель, получая подобное неработающее электронное чудо, которое его владельцы, возможно, давно и безуспешно «прогуливали» по мастерским и техническим центрам, находится в более выгодном положении, чем профессионал. Есть что-то, что безнадежно испорчено, есть интерес вернуть это к жизни, есть время, чтобы осуществить это. Нужно не так много – умение программировать. Понятно, что повторить программу, заложенную в промышленное изделие едва ли удастся, но ничто не мешает создать другую. Даже, если изделие не будет выполнять все функции, заложенные производителем, но будет выполнять только базовые, обычно этого достаточно. Если программисту-профессионалу важно изучить теоретические аспекты предмета, то как быстрее и проще научиться программировать любителю?

Программируя.

Предварительное рассмотрение проекта «Генератор»

Не хочу уверять, что ценность предлагаемого решения значительно больше вложенного в него труда, что практическая реализация проекта даст несомненные преимущества, но в определенных условиях это решение может оказаться не самым плохим.

Как и в предыдущем случае проект рассчитан на связку компьютер-микроконтроллер. Самым очевидным и достаточно удобным решение было бы построение генератора импульсов с задаваемой частотой и скважностью импульсов. Такое решение во-первых легко реализовать, во-вторых далеко не каждый любитель имеет в своем распоряжении генератор импульсов с регулируемой скважностью. Думаю, вы прекрасно справитесь с решением этой задачи без моих подсказок. А я хочу начать с построения генератора пилообразного напряжения.

Схемных решений, как мне кажется, может быть, по меньшей мере, несколько. Я остановлюсь на самом очевидном – микроконтроллер, как задающий генератор, цифро-аналоговый преобразователь (ЦАП), как формирователь сигнала. Предварительный макет решения выглядит несколько иначе:

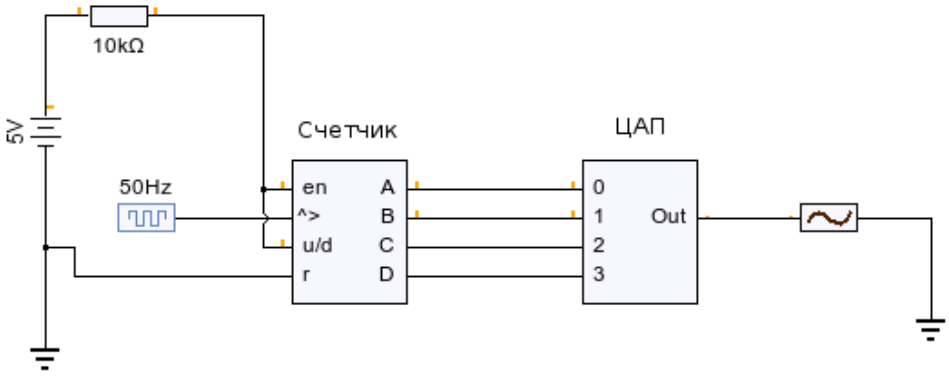


Рис. 6.1. Предварительная схема проекта

Схема, изображенная выше, в конечном виде должна приобрести вид, при котором функции тактового генератора (50 Гц на рисунке) и счетчика возьмет на себя контроллер, а цифро-аналоговый преобразователь станет вторым базовым элементом. Схема «нарисована» или, если угодно, «собрана» мною в весьма полезной программе по имени KTechlab, о которой я рассказывал в предыдущей книге «Наглядная электроника». Работа этой схемы в программе KTechlab выглядит следующим образом:

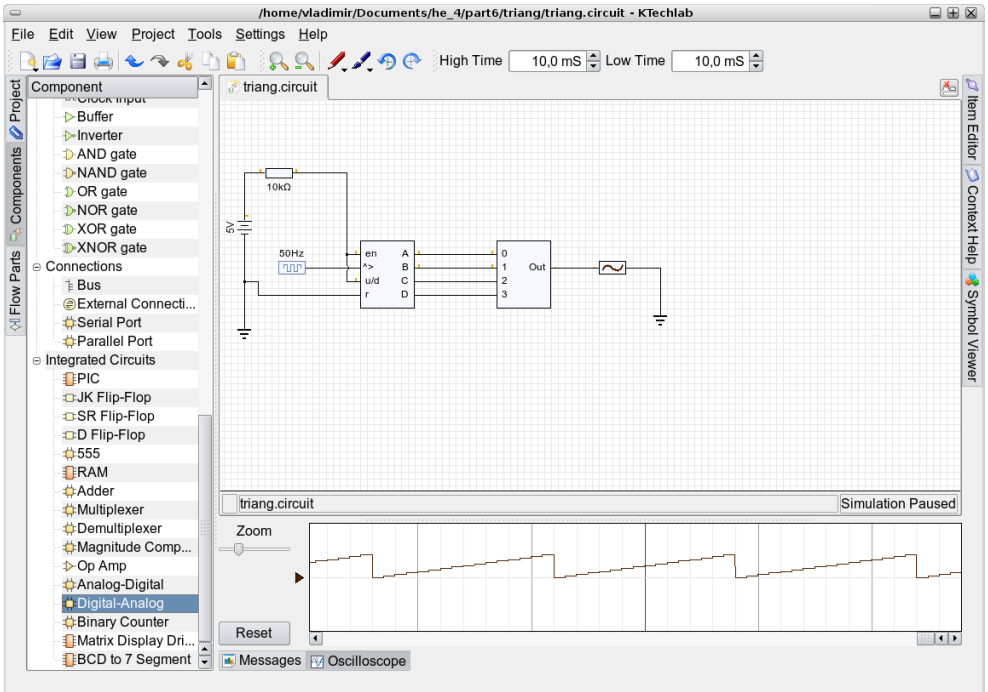


Рис. 6.2. Генератор пилообразного напряжения в программе KTechlab

Здесь вид сигнала при «прямой» работе счетчика, он считает в прямом направлении. Можете проверить, что присоединив вывод счетчика, обозначенный в программе, как «u/d» к

общему проводу, вы получите «обратную пилу»:

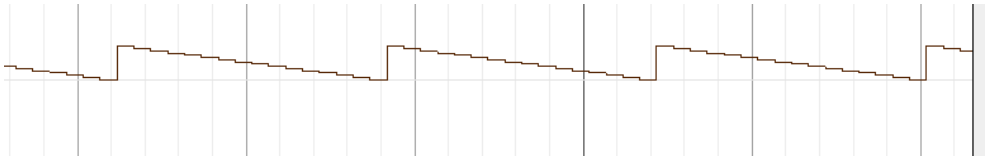


Рис. 6.3. «Обратное» пилообразное напряжение

Из этого можно сделать (поспешное?) предположение, что смена направления счета в середине периода должна дать «треугольное» напряжение на выходе генератора.

Предположение, возможно, и не поспешное, а реализация, которую я пытаюсь осуществить, явно поспешна – добавить четырех-входовый элемент И с инвертором на одном из входов, выход которого присоединить ко входу выбора направления счета. Счетчик у меня «застревает» в середине отсчета, «качаясь» между «8» и «7».

Попытаюсь, что-то изменить. Что-то изменить...

И в итоге получается следующее:

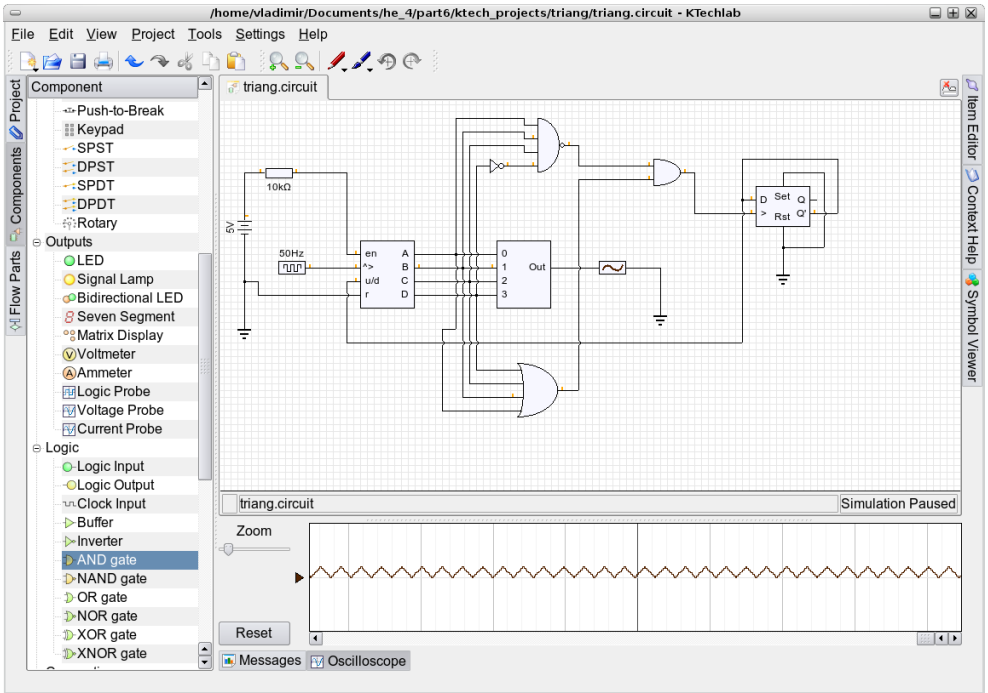


Рис. 6.4. Реализация треугольного напряжения на выходе генератора

Будем надеяться, что с идеей все в порядке, и можно подвести некоторые предварительные итоги.

- С помощью двоичного счетчика и цифро-аналогового преобразователя можно получить на выходе генератора «пилообразное» напряжение.
- Изменение направления счета меняет характер «пилообразного» напряжения.
- Если менять направление счета в середине периода, то можно получить «треугольное»

напряжение на выходе генератора.

- Реализация генератора с такими параметрами на базе цифровых микросхем возможна, но не столь проста, как хотелось бы.
    - Дополнительные проблемы возникнут при разработке конструкции.

В последнем случае я имею ввиду необходимость добавления элементов перестройки частоты, необходимость создания шкал и т.п., если, напомню, создавать схему на элементах, подобных тем, что изображены на рис. 6.1. Иное дело волшебная пара – микроконтроллер и программа на Gambas.

Продолжение работы над проектом «Генератор»

Прошу прощения, но если есть работа, которую можно «работать», ее следует «работать». Две недели ушли на эту работу, пришлось оставить книгу, но время не было потрачено зря. Я еще больше укрепился в мысли, что любителям сегодня следует учиться программировать. А с другой стороны, посмотрев сайты радиолюбителей, которые жалуются, что нынче, совсем не то, что раньше. Что не пойдешь в ближайший магазин, чтобы купить нужные детали, в лучшем случае закажешь через Интернет, и ждешь, когда все это получишь. Что элементы стали настолько маленькими, что с ними трудно работать без специального оборудования, что без дорогостоящих приборов нет смысла работать с современными компонентами. Что по этой причине, да и потому, что молодежь, которая через радиолюбительство приходила в профессиональную деятельность, сегодня понимает, чем «горбиться» годы, изучая электронику и все с ней связанное, лучше заработать деньги там, где и знать меньше нужно, да и работать проще, и, главное, денег много больше. И если вы подумали, что это все я нашел на отечественных сайтах, то не думайте так, речь идет о сетованиях американских любителей электроники (electronic hobbyists). Вот так.

Кроме того, я наткнулся на статью, где автор, много лет посвятивший популяризации Linux, принимавший участие в превращении этой операционной системы в то, чем она стала сегодня, задает себе тот же вопрос, который я задаю себе на протяжении всей книги: «А не напрасный ли это труд?».

Мне несколько проще ответить на этот вопрос. Я не стремлюсь к обязательному изданию книги, входя в положение издательств, задача которых не столько издать, сколько продать изданное. Если между пользователями компьютеров так мало пользователей Linux, то и рисковать с изданием книг о Linux не следует. Но мне довольно того, что мой сайт, где я помещаю свои книги, кто-то посещает, что дает надежду на полезность размещенного там хотя бы для малой части посетителей. Если хотя бы несколько человек из тех десятков тысяч, что заходили на сайт за десять лет его существования, нашли для себя что-то полезное, если хотя бы несколько человек увлеклись электроникой, поверив мне, что это интереснейшее занятие, то можно смело утверждать, что книги написаны не зря.

Но есть одно, мне кажется существенное, но. Если количество тех, кто увлечен электроникой или радиолюбительством, сокращается, если из тех, кто мог стать радиолюбителем, все больше молодежи остается за компьютером, все меньше тех, кто готов взять паяльник в руки и спаять простейшую схему, то больше ли вреда я принесу им, чем пользы? Не знаю.

Когда-то многие любители сетовали на вытеснение электронных ламп из обихода. Им было удобно, с чем я согласен, работать с лампами. Потом микросхемы заменили транзисторы. И работать с микросхемами менее комфортно, чем с транзисторами. Так и есть. Сегодня микросхемы стали либо слишком маленькими (поверхностный монтаж), либо очень сложными (процессоры и иже с ними). Мне кажется, что понять многие проблемы работы с

этими компонентами проще за компьютером. Но это перестает быть похоже на радиолюбительство.

Словом, дописываю эту главу книги, как получится, и книгу на этом завершу. А потом подумаю, а следует ли начинать новую?

В плане завершения главы мне хочется рассмотреть небольшие проблемы, если отнести эти мелочи к проблемам – как устроить так, чтобы микроконтроллер менял частоту генератора по заданным в программе интерфейса значениям частоты или периода. Самый простой способ задания частоты в программе контроллера – использование цикла при счете, где наращивания числа на выходе происходит с паузой. Что-нибудь вроде вложенного цикла:

```
start: for (i=0; i<=255; i++) // Наращивание значения на выходе
       {     PORTA = i;
             for (k=0; k<period; k++); // Пауза
       }
       goto start;
```

Я не уверен, что можно передать значение period, что значение не должно быть константой. И, конечно, этот период лучше было бы организовать с помощью таймера или ассемблерной вставки, используя дополнительную программу PiKLoops. Второй вопрос, который я задаю себе: «Как остановить этот бесконечный цикл, чтобы изменить частоту?»

Последнее я попробую сделать с помощью прерывания при изменении состояния usart микроконтроллера. Мне не хочется включать осциллограф, устанавливать ЦАП и наблюдать реальный сигнал на выходе. Речь идет только об эксперименте, который определил бы возможность применения решения. Поэтому я воспользуюсь тем, что имею на макетной плате – тремя светодиодами, подключенными к выводам контроллера. После передачи по COM-порту чисел от 1 до 9 скорость их переключения должна измениться заметным образом, что и будет свидетельствовать о правильной работе схемы. Начальную же скорость переключения я постараюсь подобрать опытным путем удобной для наблюдения. Вот такой у меня план.

Итак, первое, что можно сделать, это подобрать подходящее для наблюдения время в цикле задержки, который определит период конечного сигнала. Программа будет выглядеть следующим образом:

```
int per;
int i;
int l;

void main() {

       PORTA = 0x0;      // Настройка портов A и B
       TRISA = 0x38;
       TRISB = 0xFA;
       per = 1000; // Цикл задержки для формирования периода
start:       for (i=0; i<8; i++) {
             PORTA = i; // Переключение трех выводов порта A.
             for (l=0; l <= per; l++); // Формирование периода сигнала.
       }
       goto start;
}
```

С помощью переменной per я постараюсь подобрать удобное для наблюдения время. Я использую компилятор SDCC, а при создании нового проекта воспользуюсь предлагаемым

шаблоном, который позволяет задать нужную конфигурацию, записываемую по адресу 2007h. Напомню, что я использую внутренний тактовый генератор, а слово конфигурации в программе Piklab выглядит как 2118. В данный момент я не буду пытаться использовать симулятор, а сразу запишу полученный hex-код в микросхему. И, если не забыть настроить хотя бы порт A на вывод, как это сделал я, то изменение значения переменной per с 1000 на 10000 дает удобный видимый эффект. Попутно я разрешил свои сомнения по поводу условия работы цикла for, похоже, максимальное значение можно задать с помощью переменной. Формирование цикла задержки я намереваюсь с помощью коэффициента, так что переменная будет выглядеть как per = 1000*k. Можно проверить это до выполнения следующего шага. Хотя это и выглядит нелепо, но займет немного времени.

Теперь мне хотелось бы проверить, будет ли работать идея при использовании прерывания. Шаблон дает ту часть, которая отведена для кода подпрограммы прерывания. В подпрограмме я намереваюсь получить значение коэффициента, назовем его «k», на который позже умножим переменную per. Меняя значение k от 1 до 9 я попытаюсь добиться такого же эффекта, что и ранее.

В описании микросхемы PIC16F628A есть упоминания, касающиеся прерываний. В первую очередь в части, касающейся приема данных по USART:

Рекомендованные действия при приеме данных в асинхронном режиме:

- Установить требуемую скорость передачи с помощью регистра SPBRG и бита BRGH (см. раздел 12.1).
- Выбрать асинхронный режим сбросом бита SYNC в ‘0’ и установкой бита SPEN в ‘1’.
- *Если необходимо, разрешить прерывания установкой бита RCIE (бит 5) в ‘1’ (PIE1=8Ch).*
- Если прием 9-битный, установить бит RX9 в ‘1’.
- Разрешить прием установкой бита CREN в ‘1’.
- Ожидать установку бита RCIF, или прерывание, если оно разрешено битом RCIE.
- Считать 9-й бит данных (если разрешен 9-разрядный прием) из регистра RCSTA и проверить возникновение ошибки.
- Считать 8 бит данных из регистра RCREG.
- При возникновении ошибки переполнения сбросить бит CREN в ‘0’.

Здесь упоминается регистр PIE1. Обратимся к описанию этого регистра:

Регистр PIE1 (адрес 8Ch), доступен для чтения и записи, содержит биты разрешения периферийных прерываний.

Примечание. Для разрешения периферийных прерываний необходимо установить в '1' бит PEIE (INTCON<6>).

EEIE CMIE RCIE TXIE нет CCP1IE TMR2IE TMR1IE

бит 7 (EEIE – Разрешение прерывания по окончанию записи в EEPROM данных):

1 = прерывание разрешено
0 = прерывание запрещено

бит 6 (CMIE – Разрешение прерывания от компараторов):

1 = прерывание разрешено
0 = прерывание запрещено

бит 5 (RCIE – Разрешение прерывания от приемника USART):

1 = прерывание разрешено
0 = прерывание запрещено

бит 4 (TXIE – Разрешение прерывания от передатчика USART):

1 = прерывание разрешено
0 = прерывание запрещено

бит 3 (Не используется, читается как '0').
бит 2 (CCP1IE – Разрешение прерывания от модуля CCP1):

1 = прерывание разрешено
0 = прерывание запрещено

бит 1 (TMR2IE – Разрешение прерывания по переполнению TMR2):

1 = прерывание разрешено
0 = прерывание запрещено

бит 0 (TMR1IE – Разрешение прерывания по переполнению TMR1):

1 = прерывание разрешено
0 = прерывание запрещено

И еще один регистр, упомянутый выше.

Регистр INTCON (адрес 0Bh, 8Bh, 10Bh или 18Bh), доступен для чтения и записи, содержит биты разрешений и флагов некоторых источников прерываний.

Примечание. Флаги прерываний устанавливаются при возникновении условий прерываний вне зависимости от соответствующих битов разрешения и бита общего разрешения прерываний GIE (INTCON<7>).

GIE PEIE T0IE INTE RBIE T0IF INTF RBIF

бит 7 (GIE – Глобальное разрешение прерываний):

1 = разрешены все немаскированные прерывания
0 = все прерывания запрещены

бит 6 (PEIE – Разрешение прерываний от периферийных модулей):

1 = разрешены все немаскированные прерывания периферийных модулей

0 = прерывания от периферийных модулей запрещены

бит 5 (T0IE – Разрешение прерывания по переполнению TMR0):

1 = прерывание разрешено
0 = прерывание запрещено

бит 4 (INTE – Разрешение внешнего прерывания INT):

1 = прерывание разрешено
0 = прерывание запрещено

бит 3 (RBIE – Разрешение прерывания по изменению сигнала на входах RB7:RB4 PORTB):

1 = прерывание разрешено
0 = прерывание запрещено

бит 2 (T0IF – Флаг прерывания по переполнению TMR0):

1 = произошло переполнение TMR0 (сбрасывается программно)
0 = переполнения TMR0 не было

бит 1 (INTF – Флаг внешнего прерывания INT):

1 = выполнено условие внешнего прерывания на выводе RB0/INT (сбрасывается программно)
0 = внешнего прерывания не было

бит 0 (RBIF – Флаг прерывания по изменению уровня сигнала на входах RB4:RB7 PORTB):

1 = зафиксировано изменение уровня сигнала на одном из входов RB7:RB4 (сбрасывается программно)
0 = не было изменения уровня сигнала ни на одном из входов RB7:RB4

Я выделил то, что по моему мнению относится к задаче. Как мне кажется программа может выглядеть следующим образом:

```
/* ------------------------------------------------------------------ */
/* Template source file generated by piklab */
#include <pic16f628a.h>

/* ------------------------------------------------------------------ */
/* Configuration bits: adapt to your setup and needs */

typedef unsigned int word;

word at 0x2007 CONFIG = _WDT_OFF & _PWRTE_OFF & _INTOSC_OSC_NOCLKOUT &
_MCLRE_OFF & _BOREN_OFF & _LVP_OFF & _DATA_CP_OFF & _CP_OFF;

int i;
int k;
int l;
int per; // Переменная, определяющая с коэффициентом k период.
```

```
unsigned char getch() // Получение байта
{
      while(!RCIF)      // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}

void isr() interrupt 0 {
/* interrupt service routine */
    /* << insert interrupt code >> */
      k = getch() - 0x30;
      return;
}

void main() {
    /* << insert code >> */
      PORTA = 0x0;      // Настройка портов A и B
      CMCON = 0x7;
      TRISA = 0x38;
      TRISB = 0xFA;
      RCSTA = 0x90;     // Настройка приемника
      TXSTA = 0x6;      // Настройка передатчика
      SPBRG = 0x68;     // Настройка режима приема-передачи
      PORTB = 0xFE;     // Выключаем драйвер RS485 на передачу
      PIE1 = 0x20;      // Разрешение прерывания от приемника USART.
      INTCON = 0xC0; // Разрешение перефирийных прерываний.
      TXEN = 0; // Выключаем передачу
      CREN = 1; // Включаем прием.
      k = 1;

start:      per = 1000*k;
      for (i=0; i<8; i++) {
            PORTA = i;
            for (l=0; l <= per; l++);
      }
      goto start;
}
```

Получив после трансляции cod-файл для запуска симуляции, я хочу проверить работу программы в gpsim. Выделив переменные k и per, я добавляю их в окно наблюдения. Первый ввод с клавиатуры в окне USART дает мне:

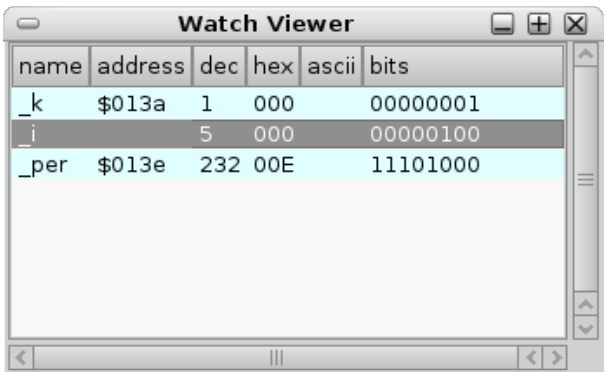


Рис. 6.5. Наблюдение переменных при задании коэффициента 1

На макетной плате выводы, к которым я присоединяю светодиоды, меняются с некоторой

частотой, которая меняется, если я ввожу с клавиатуры коэффициент 9 при активизации окна USART, а переменные меняются:

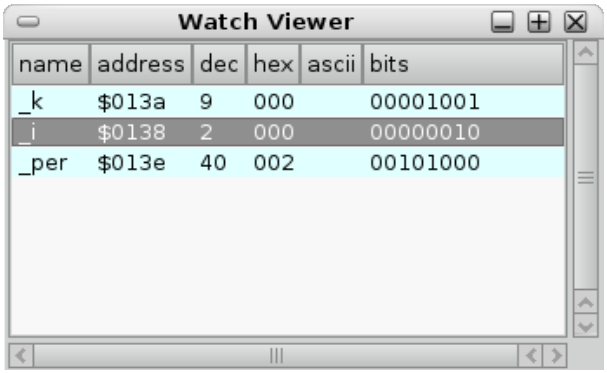


Рис. 6.6. Наблюдение переменных при вводе коэффициента 9

Все, казалось бы, хорошо. Но после программирования микросхемы и переноса ее на реальную макетную плату я не вижу заметных изменений в поведении светодиодов. Я пытаюсь работать с графическим интерфейсом. Пытаюсь проверить, запуская пример работы с последовательным портом. Результат одинаков. То есть, я его не вижу.

Теперь следует определиться, где возникла проблема? В программе, написанной для микроконтроллера, или в программе графического интерфейса?

Самый простой способ отделить «зерна от плевел» – использовать готовую программу для работы с COM-портом. Что я и делаю. И она позволяет мне увидеть изменение в поведении светодиодов. Из чего следует, что проблемы возникают при передаче значения от моего графического интерфейса к контроллеру.

Попробуем разобраться с этим.

Для отправки значения в COM-порт я использовал функцию (или метод) PRINT. Есть другая возможность – WRITE. Заменив в программе интерфейса обработку нажатия клавиши «Запись», с помощью которой я отправляю нужное мне значение в последовательный порт, следующим образом:

```
PUBLIC SUB Button1_Click() ' Клавиша "Запись"
      WRITE #SerialPort1 TextBox1.Text
END
```

я получаю окончательный результат, который был нужен. Теперь контроллер послушно меняет скорость переключения светодиодов.

Я не придумал в назидательных целях последнюю проблему. Все вышло само-собой. Но получилась яркая иллюстрация того, что последнее слово остается за физической реализацией. Как бы ты ни был уверен, что все правильно, только собрав устройство и проверив его работу, можно определить (да и то не всегда), что работа завершена, а устройство сделано правильно и работает.

Завершение

Конечно, у меня не было намерения показывать конкретную разработку. Мне хотелось рассказать о среде программирования Gambas, несколько больше, чем я сделал это ранее, рассказать о системе программирования PIC-контроллеров Piklab и отладчике gpsim. Более

всего мне хотелось показать, что почти любая разработка сегодня связана с программированием, и что любителям полезно и нужно научиться программировать, благо есть такие великолепные программные средства, как Gambas, Piklab и gpsim. Полнофункциональные и свободные, доступные не только профессионалам, но и любому и каждому.

Вместе с тем, как и любая разработка, описанная выше может найти практическое применение. Не столько в качестве основы функционального генератора, сколько при создании смешанных генераторов, когда одновременно нужно иметь взаимосвязанные логические (или импульсные) и аналоговые сигналы. Например, если использовать не линейную функцию управления ЦАП, а создать массив, заполненный числами, соответствующими синусоидальной функции, то на выходе генератора можно получить ступенчатое приближение синусоидального сигнала, или любого нужного сигнала. Можно использовать генератор пилообразного напряжения для управления варикапом, можно найти, я полагаю, множество применений даже для такой простой схемы.

Сомнения, появившиеся у меня, когда я еще только подумывал о написании этой книги, не развеялись, как я надеялся, в процессе работы, но только усилились. Кроме первоначальных сомнений по поводу моего непрофессионального рассказа о программировании, появились сомнения, которые касаются отношения любителей моего поколения к компьютерам и всего, что с ними связано. И любители электроники, и радиолюбители сетуют на то, что молодежь все больше замыкается в программировании, забывая о физической реализации своих проектов. Так, во всяком случае, я понял сетования американских любителей. И это справедливо. Я не так плохо умею пользоваться паяльником, но я использовал некогда, по моим подсчетам года два назад, спаянный макет для проверки всего, что было необходимо при написании этой книги, да и предыдущей. Мне не понадобилось что-то паять, если не считать простой схемы программатора, которая работала бы с Piklab. Справедливые сетования.

Linux. Я использую эту операционную систему несколько лет, не вижу какой-либо разницы с точки зрения пользователя между ней и Windows, разве, что она удобнее во многих отношениях. Когда я начинал пользоваться Linux, мне не хватало, например, англо-русского словаря, чтобы проверить, правильно ли я понимаю документацию, с которой работаю. Но очень быстро нашелся Stardict, который ничем не хуже Lingvo. Не хватало программ САПР (EDA), но со временем отыскались и они. Я говорил выше, когда рассказывал о том, как просто можно в Gambas создать полезную программу для того, чтобы научиться печатать вслепую. Я обещал, что постараюсь овладеть печатью вслепую, используя ее. Но есть и готовая программа, о которой я тоже упоминал. И все, что написано в этом послесловии латиницей, написано мной вслепую, хотя, боюсь, я не уделял даже 10-15 минут каждодневным тренировкам – не случилось, не было настроения.

Вернувшись к работе над книгой «Экскурсия по электронике» я обнаружил, что есть еще одна среда программирования в среде Linux, о которой я даже не упомянул. Она для тех, кто предпочитает Pascal и называется Lazarus.

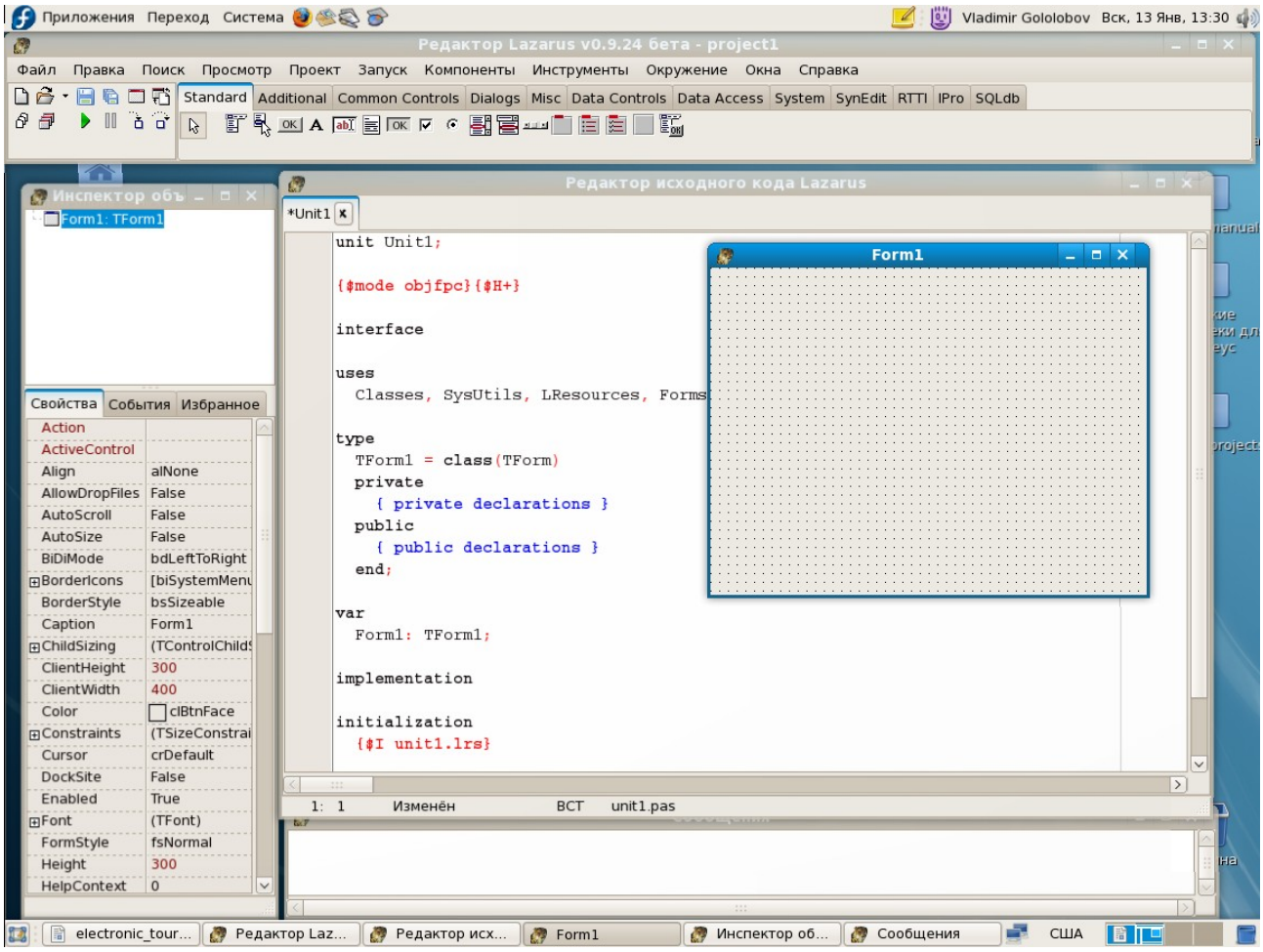


Рис. 6.7. Среда программирования Lazarus в операционной системе Linux

Если бы я знал об этом до написания этой части книги, возможно, она получилась бы несколько иной, но это мало изменило бы существо вопроса.

Мне нравится работать в Linux, я пишу об этой операционной системе, но не могу понять, почему в нашем царстве-государстве проще судить завуча школы за пиратское использование Windows, чем передать ему бесплатный дистрибутив Ubuntu. Видимо, благосостояние наших граждан так возросло, что им не нужны бесплатные программы, как бы ни были они хороши, им нравится покупать, пусть и плохое, но дорогое. Это как в старом анекдоте – война давно кончилась, а я все паровозы под откос пускаю. Видимо, даже в этом я неправ.

В результате я не столько закончил работу над книгой, сколько поспешно свернул работу над ней. Это самый мой неудачный проект. Мне жаль. Мне есть над чем призадуматься. Чем я и собираюсь заняться.

Конспекты

Gambas дружелюбен к пишущим на VB, но используя Linux

«Феноменальное количество ошибок и несообразностей, делающих Visual Basic столь очаровательным, подвигли меня начать этот проект». Вот как Benoit Minisini, 30-летний француз, живущий в пригороде Парижа, начинает описание своего проекта, названного

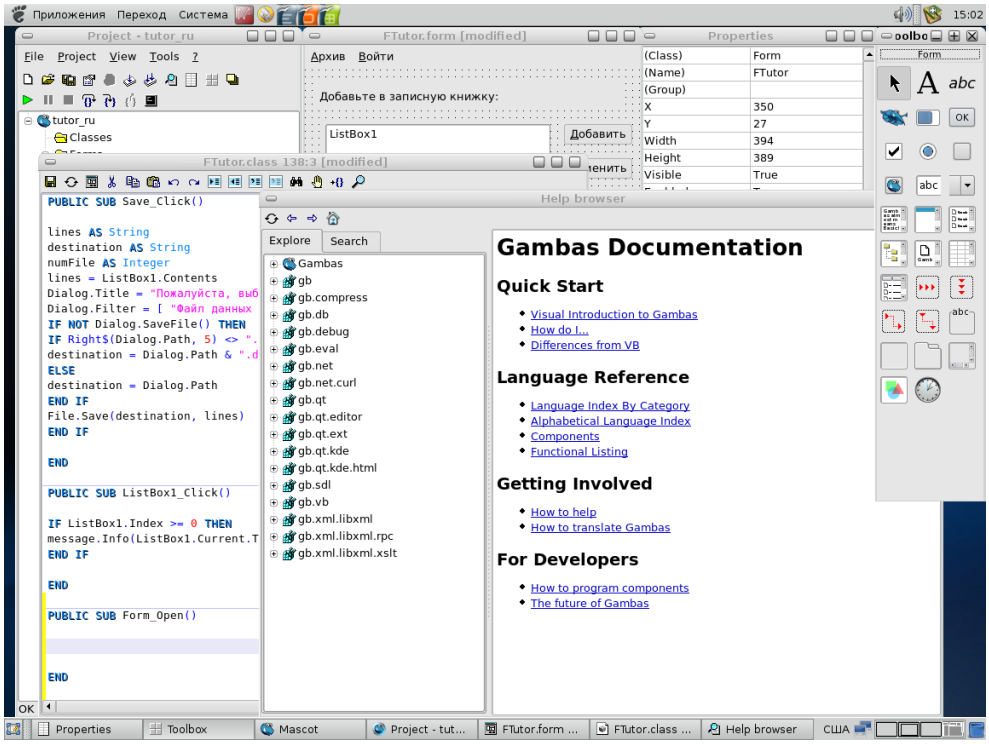
Gambas.

Gambas - это замена VB для Linux. Не клон, поскольку Benoit'y никогда не нравилась реализация Microsoft: «Кажется, Microsoft осознает низкое качество ее языка, поскольку VB.Net не поддерживает обратной совместимости с предыдущими версиями Visual Basic. Я думаю, они выбросили исходный код интерпретатора Visual Basic, и что VB .Net - это только .Net компилятор реального времени, чей синтаксис похож на Visual Basic».

Так что, Gambas предназначен быть лучше Visual Basic.

Benoit взял от VB то, что посчитал действительно полезным: язык BASIC и легкий IDE. Он также взял несколько идей от Java, создав легкий и мощный язык. Насколько мощный? Полный IDE был создан в самом Gambas, а индикатор уровня его готовности даже не достигал 1.0.

В этой статье мы постараемся описать текущий статус Gambas, и как он соотносится с VB. Мы сравним снимки экрана и исходный код, так что вы можете увидеть полную картину.



Фон

Gambas - это не первая попытка с наскока создать замену VB для Linux, но она ближе всего к завершенности.

Фактически? было много других незаконченных попыток создать клон VB для Linux: RapidQ, выкупленный RealBASIC, который сейчас только переправляет Linux запуски без IDE, Phoenix BASIC, который заморожен на 3 года, KBasic затормозился, GNOME Basic перенес фокус на предоставление VBA поддержки для приложений GNOME, XBasic готов, но *примитивнее*, чем VB, HBasic - достойный претендент, но нестабильный.

С другой стороны, есть множество языков BASIC без интегрированного IDE, и множество IDE без языка BASIC.

Gambas имеет уникальную смесь возможностей, которые помогут тысячам разработчиков на VB перебраться с Windows на Linux. Следовательно, Gambas может принести больше

приложений и пользователей в Linux.

И, наконец, разработка компонент Gambas означает для всякого наличие возможности написать GUI компонент, базируемый на GTK+ с тем же интерфейсом, что и QT, так что каждая программа Gambas может затем переключаться между QT GUI и GTK+ GUI.

Начала BASIC

Одна из первых программ, которые вы можете непременно найти в любом языке программирования, это достопочтенная «Hello World».

Hello VB World

- Начинаем с пустой формы (form)
- Размещаем клавишу команды - Command Button (Command1) и этикетку (Label1) на форме
- Пишем код:

```
Sub Command1_Click()
      Label1.Caption = "Hello World!"
End Sub
```

Hello Gambas World

1. Начинаем с пустой формы (form)
2. Размещаем клавишу - Button (Button1) и этикетку (Label1) на форме
3. Пишем код:

```
SUB Button1_Click()
      Label1.Text = "Hello World!"
END
```

Если вы не хотите создавать программу, основанную на форме (form-based), вы можете начать проект без форм и написать код:

```
SUB main()
      PRINT "Hello World!"
END
```

Есть множество прелестных простых вещиц, которые вы можете сделать с Gambas - и еще нечто, что не может Visual Basic! (В VB есть метод Print, но он обращается только к отладчику и GDI объектам, подобным принтеру, форме или вместилищу картинки.)

Отличия от VB

Поскольку Gambas не предназначался быть клоном Microsoft Visual Basic, он все еще BASIC, и есть много сходства между этими двумя языками, и много связей между функциональными возможностями. Есть, возможно, больше сходства, чем различий, но вы можете просто скопировать свои VB проекты и рассчитывать на компиляцию под Gambas.

Из раздела DifferencesFromVB:

Не специфические для языка различия

- VB вставляет код класса для каждого объекта формы в тот же самый файл, где находится определение формы. Gambas держит их раздельно в .form и .class файлах.
- Расширения файлов:

VB	Gambas	Тип файла
.vbp	.project (только .project, один на директорию)	Файл определения проекта

.bas	.module	Модуль
.cls	.class	Файл класса
.frm	.form	Файл определения формы
.frx	(все, что хотите)	Двоичные файлы ресурсов

- Проекты Gambas определены, как директория с файлом .project в ней, и все файлы находятся в этой директории. VB может иметь множество файлов проекта в каждой директории, и может вытащить одинаковые исходные файлы из разных директорий в разных проектах, что имеет и полезную и вредную сторону.
- Измерения экрана в VB представлены в «twips», единицах 1/1440 дюйма; в Gambas они сделаны в фактических пикселях.
- Управление формой в программах Gambas частное (private) по определению. Вы можете изменить это, войдя в диалог свойств проекта (Project Properties) и установив флажок «Make Form Controls Public».

VB имеет это, Gambas нет

- Вы не можете редактировать код в режиме аварийной остановки (Break mode) в Gambas; вам нужно вначале закончить выполнение программы.
- Параметры функции и процедуры передаются *по значению* (*by value)*. Они не могут быть переданы *по ссылке* (*by reference)*, как в Visual Basic. Заметьте, что VB передает параметры *по ссылке*, если вы не используете его ключевое слово `ByVal`, так что будьте внимательны, когда вы пытаетесь портировать VB проект.
- В Gambas нет такой штуки, как глобальная переменная расширения проекта (project-wide global variable). В качестве обходного пути рассмотрите создание класса, называемого Global, и объявите ваши глобальные переменные, как статические общие переменные (static public) в этом классе, а затем ссылайтесь на них, как на глобальные имена переменных (Global.variablename) в своем проекте. Это остается плохой практикой программирования, но во всяком случае они будут идентифицированы, как глобальные переменные, когда бы вы ни использовали их ;)
- Пока вы не включили «Option Explicit» (явный выбор) в VB модуль, вы не нуждаетесь в объявлении переменных до их использования. Gambas ведет себя так, как если бы «Option Explicit» было всегда включено, что создает намного лучший код при совсем небольшой дополнительной работе.
- Нет прямого Gambas эквивалента для «Index» свойства управления VB формой. Вы можете легко создать массивы управления, но вы должны сделать это в коде. Нет в настоящее время (0.62) пути сделать это графически. Таким образом, когда вы копируете управление и вставляете его обратно в форму, оно приходит иначе, чем приглашение для вас к созданию массива управления, автоматически переименовывая скопированное управление с подходящим именем.
- Вы не можете в настоящее время создавать прозрачные этикетки (transparent labels) в Gambas; фон всегда непрозрачный.
- В Gambas событие перемещения мышки (MouseMove event) обнаруживается только, когда клавиша мышки отжата, что означает, что вы совсем не можете использовать ее для традиционного эффекта выделения мышкой (другого, а не стандартного средства, связанного с каждым управлением). Исключение составляет управление *DrawingArea*,

которое имеет свойство трассировки (Tracking property), что позволяет получить событие движения мышки, даже если клавиша нажата.

Gambas имеет это, VB нет

- В отличие от VB, вам не нужно компилировать в GUI поддержку, если вы хотите написать приложение командной строки в Gambas. Только снимите «gb.qt» компонент в свойствах проекта (Project Properties) и будьте уверены, что определили Sub Main.
- Gambas придерживается концепции *control groups*, которая позволяет вам поддерживать события из любого числа разных мест управления с единственной поддержкой подпрограммой. Это удаляет излишний код, и может быть использовано для выполнения разных вещей, которые делает индексное управление в VB, и еще некоторых, которые VB сделать не может.
- Тогда как VB делает невозможным запустить программу синхронно и получить ее вывод без изучения того, как сделать вызовы API (Shell просто запускает программу в фоновом режиме), Gambas позволяет вам сделать это, используя SHELL и EXEC, управление процессами при старте с использованием Process object, и даже читая из и записывая в них, позволяя вам легко добавить функциональности со вспомогательными приложениями. Это создает условия для невероятно легкого написания внешнего интерфейса для почти любой процедуры командной строки.
- Вы можете сделать все, что выше, так же успешно со специальными файлами и Unix устройствами, такими как последовательные или параллельные порты. Используйте /proc файловой системы для написания RAID монитора, например, или используйте именованные конвейеры (pipes) для получения множественных каналов информации от прикладной части программы на любом другом языке.
- Для получения окон нестандартной формы вам достаточно установить ME.Mask свойство текущего окна к картинке, которая имеет прозрачные области. VB требует вызова API и чуть-чуть работы.
- Вы можете создавать управление и меню динамически, только обрабатывая их инструкцией NEW.
- В Gambas вы можете создавать и отображать столько много копий формы, сколько захотите.
- Вы можете вставить Gambas форму в другую: когда вы обрабатываете первую, задайте вторую, как родительскую.
- И, конечно, Gambas - это Free Software, чье расширение окружения написано в нем самом, позволяя вам подгонять его к более серьезному использованию, чем тренировки в BASIC.

Усиливая мощь Linux

«Это Unix философия: Пишите программы, которые делают одну вещь, но делают ее хорошо. Пишите программы для совместной работы. Пишите программы для поддержки текстовых потоков, поскольку это универсальный интерфейс» -- Doug McIlroy.

Одно из основных преимуществ Gambas над VB в том, что Gambas построен в соответствии с философией Unix. Это значит, вы можете усилить мощь Linux и его тысяч средств вместо написания кода.

Давайте, рассмотрим некоторые примеры.

- примеры управления другими программами с использованием конвейеров (pipes)
- cd burner? ()
- sample player? (mpg123?)
- загрузка файлов из Интернета? (то есть, lynx -source url)

Приумножая ваши текущие знания

Если вы всегда программировали на BASIC, вы почувствуете теплое ощущение от знакомого вам, когда начнете программировать в Gambas.

VB программисты почувствуют полный комфорт с самого начала.

Чтобы понять, что Gambas - это не VB, и VB программистам нужно быть готовыми к некоторым различиям. Но большинство изменений существуют по определенным причинам: чтобы сделать язык лучше. (Важно помнить, что Gambas позаимствовал некоторые хорошие идеи у Java и других языков программирования.)

С Gambas любой может начать программировать графическое приложение сразу, и это приведет больше программистов, и принесет больше приложений в GNU/Linux.

Конвертирование вашего существующего кода

Если вы VB программист, и не чувствуете особого счастья от последних изменений платформы (и сопутствующей цены), вы можете подумать о конвертировании ваших приложений в Gambas.

Разработчики Gambas уже сделали небольшой скрипт, в настоящий момент в версии 0.1, который конвертирует VB формы в Gambas.

Vb2Gb было написано на Perl по причине больших возможностей по обработке текста в этом языке, и вскоре будет реализовано в Gambas.

В VB описание формы и исходного кода смешаны в едином файле. Gambas разделяет это на два файла: описание формы и модуль класса.

Есть также несколько имен управления и свойств, которые мы можем конвертировать автоматически. Например, в VB мы имеем «CommandButtons», тогда как в Gambas (или QT?) они названы просто «Buttons»; свойство «Caption» названо «Text» в Gambas.

Резюмируя

Gambas - это самое близкое к VB в мире Linux. Проект не был изначально предназначен заменить миллионы строк VB кода, но предоставляет быстро прогрессирующий инструмент, базирующийся на языке BASIC. Тем не менее, Gambas уже представляет собой инструмент для легкой миграции с собственных приложений VB.

Gambas в постоянном совершенствовании. Его текущая версия рассматривается как alpha программа, поскольку некоторые возможности еще определяются, но Gambas уже используется в таких сложных приложениях, как его собственный IDE. Нет смысла спешить с релизом 1.0, и есть цель - сделать хорошую вещь сразу.

Gambas имеет сетевой график. Некоторые из новых возможностей, которые ожидаются в следующих выпусках:

- Компонент Network
- SDL компонент
- Персистентная система объектов
- Проектировщик отчетов
- Perl'о-подобный компонент регулярных выражений

Согласно Daniel Campos, создателю компонента Network, он достаточно стабилен, чтобы именоваться 'beta' версией, и уже предоставляет некоторые интересные функциональные возможности:

- DNS/NIS клиент
- Socket клиент : для начала TCP или UNIX socket соединения.

- Socket Server : для обслуживания TCP или (UNIX, пока нет) sockets.
- Datagram Client/Server : UDP sockets
- SerialPort : класс для управления последовательными устройствами (RS-232, и т.д.)

TO-DO: завершить статью с необузданным оптимизмом

Разработка приложений в Gambas

Руководство и пример выполнены в Gambas

Резюме: Мы собираемся создать простую программу в Gambas. Мы научимся поддерживать события и некоторые типы и техники для работы с этим удивительным инструментом.

Моя благодарность: Daniel Oxley и Dave Sharples. Они просмотрели и поправили мой бедный английский перевод с испанского.

Загружаемый исходный код (комментарии, имена переменных и т.д.) сделан на испанском. Однако код, показанный на этих страницах, транслировался.

David Asorey Alvarez. February de 2005.

Введение

Gambas - это IDE (Integrated Development Environment - интегрированное окружение разработки), ориентированное на RAD (Rapid Applications Development - быстрая разработка приложений) подобное популярным патентованным программам Microsoft Visual Basic или Borland Delphi.

В Gambas возможно разрабатывать GUI программы (Graphical User Interface - графический интерфейс пользователя) очень быстро, поскольку IDE включает конструктор форм, редактор кода, обозреватель кода, интегрированную систему помощи и т.д.

Такого рода инструменты весьма характерны для мира Microsoft Windows, на платформе Linux есть только несколько, или они на ранних стадиях разработки. Мы находим несколько, подобно Kdevelop, Kylix или VDK Builder.

Есть добрая традиция и обыкновение в мире Linux/Unix использовать множество различных инструментов, каждый из которых специализируется на решении единственной конкретной задачи (например, компилятор, редактор, отладчик - каждый из них используется отдельно). По этой причине программные IDE относительно внове для пользователей Linux/Unix.

Есть множество программистов и разработчиков, кто пользуется такого рода интегрированными инструментами, поскольку они пришли с других платформ или они чувствуют себя комфортнее с IDE.

По словам автора, Benoit Minisini: «*Gambas нацелен на то, чтобы дать вам возможность создавать мощные программы легко и быстро. Но чистота программирования остается на вашей совести...*». Язык программирования, используемый в Gambas - это версия «старого» BASIC. Возможно, что кто-нибудь будет удивлен этим выбором, поскольку BASIC кажется слишком простым и ограниченным. Должны напомнить, что одной из целей Gambas является содействие программированию не программистов.

Цель данного руководства - дать введение в Gambas IDE, и разработать простую программу с его помощью, но мы подразумеваем, что читатель в какой-то мере знаком с программированием, и что термины подобные функции, событию или переменной ему

знакомы. Gambas имеет встроенную систему помощи, предоставляющую и вводную, и более детальную документацию.

Версия Gambas, использованная в данном руководстве - это 1.0−1. Домашняя страница Gambas - http://gambas.sourceforge.net

Загрузите исходный код примера программы: agenda.tar.gz

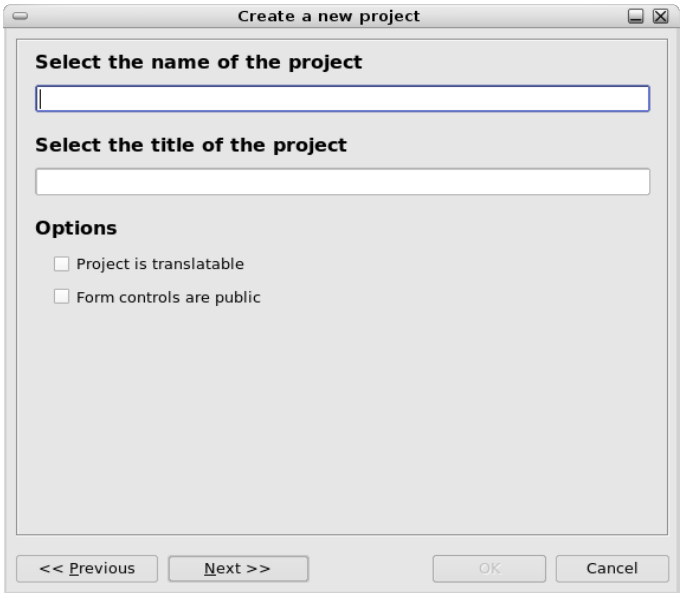
Первые шаги

В Gambas help есть документ озаглавленный «Наглядное введение в Gambas - Visual Introduction to Gambas», который рассказывает о программе, ее различных инструментах, окнах и меню. Мы не собираемся повторять эту информацию в данном руководстве.

Мы постараемся разработать законченную программу в Gambas с самого начала, и разберемся с требованиями и проблемами по мере их появления.

Наша программа будет приложением типа записной книжки или памятки. Мы сможем добавлять и удалять записи, модифицировать существующие заметки. Они будут сохраняться и читаться из файла.

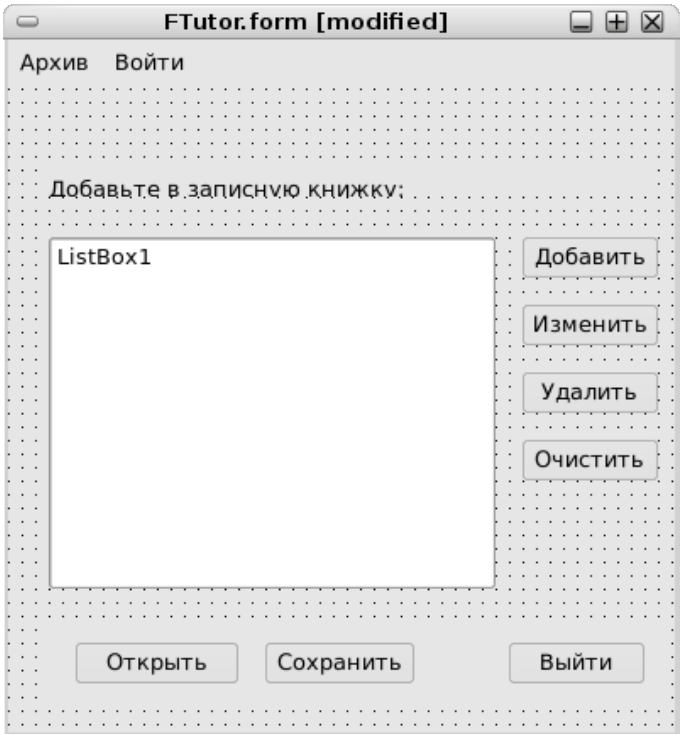
Как только Gambas откроется, мы выберем «New project - новый проект» из меню. Определим его, как graphical project, графический проект, в мастере проектов и мы затем должны обеспечить некоторую информацию, такую как имя проекта и заголовок проекта:



Есть две опции: «Project is translatable - проект транслируемый» и «Form controls are public - управление формами общее». Мы оставим эти опции не выбранными и нажмем клавишу **Next**.

Последняя задача в мастере проектов - выбрать директорию, где будет сохраняться проект. После чего откроется основное Gambas IDE. В главном окне проекта следует щелкнуть правой клавишей мышки на **Forms** и выбрать «New form - новая форма».

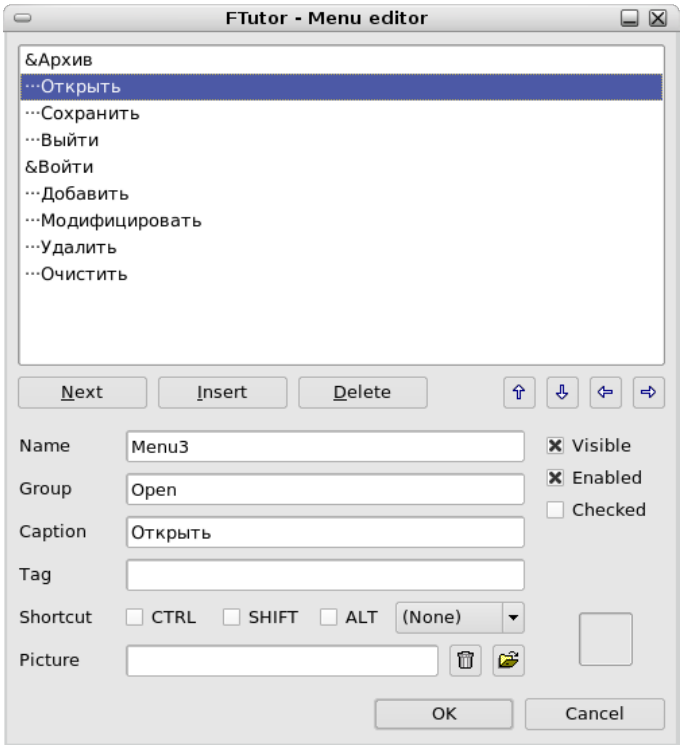
Мы собираемся разработать основную форму, которая содержит управление ListBox и несколько клавиш (open - открыть, save - сохранить, exit - выйти, add - добавить, modify - изменить, delete - удалить, ...). Нечто, похожее на это:



Можно видеть несколько общих средств управления: Label - надпись, ListBox - область списка и несколько клавиш. Мы добавляем каждое средство управления в форму, выбирая их на инструментальной панели (Toolbox) Gambas и «рисуя» контур управления на форме. Пожалуйста, отметьте, что клавиши **Open** - открыть, **Save** - сохранить и **Exit** - выйти расположены на **Panel** - панель, которая должна быть добавлена на форму первой и не непосредственно в основное тело формы.

Если мы хотим задать горячие клавиши, мы должны предпослать символ «амперсанд» (&) выбранной букве в тексте клавиши. Например, О&ткрыть, &Сохранить

Мы можем создать и отредактировать основное меню нашей программы, щелкнув правой клавишей мышки на свободном пространстве формы и выбрав в выпадающем меню «Menu editor - редактор меню»:



После создания пунктов меню вы можете заметить, что среди типичных свойств, как имя, надпись, и т.д., есть также свойство, названное «Group - группа», клавиши формы имеют это свойство Group, как вы можете увидеть, выбрав клавишу и просмотрев ее свойства в окне «Properties - свойства». Эта опция очень интересна, поскольку мы можем ассоциировать один и тот же код с разными средствами управления, имеющими одинаковые функции. Например, пункт меню «Open - открыть» и клавиша **Open** должны выполнить одну и ту же задачу: открыть файл на диске. Используя свойство Group, мы напишем код для этой задачи только один раз.

В нашей программе мы объединим клавишу и пункт меню Open в группу, названную Open, Save - сохранить меню и клавишу в группу, названную... Save, и т.д.

Теперь, если мы дважды щелкнем по клавише или одноименному пункту меню, редактор

кода откроется и курсор окажется на заголовке подпрограммы, названной общим именем группы, выбранном для средств управления, ассоциированных с ней. Например: PUBLIC SUB Open_Click() , где Open - это свойство Group, названной Open, определенное ранее.

Поддержка событий

Программы с графическим пользовательским интерфейсом обычно «event driven - движимые событиями». Что означает, когда пользователь «заставляет что-либо произойти» в окне приложения, генерируется событие. Это событие может быть связано с функцией или подпрограммой, которые отвечают на действие пользователя.

Пример: если пользователь «щелкнул» по средству управления, генерируются некоторые события - MousePress, при нажатии клавиши мышки, MouseRelease, при отпускании, и, наконец, Click (щелчок), как глобальное действие. Если пользователь сделал двойной щелчок, тогда генерируется событие DblClick. Не все средства управления отвечают на события или генерируют события, нелепо говорить о событии Resize in a button (уменьшить размер клавиши), поскольку это событие обычно генерируется, когда мы изменяем размер окна (управление Form).

В Gambas мы будем редактировать код процедуры для события (1), подобный этому:

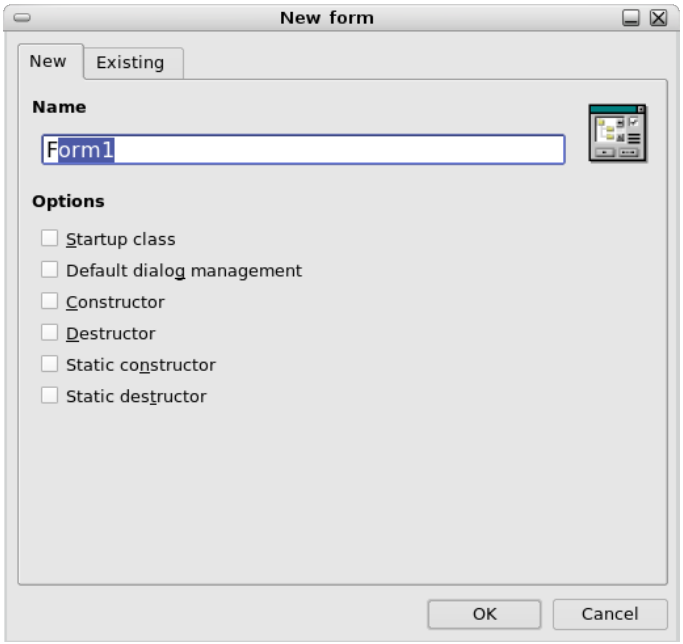
```
PUBLIC SUB Control_Event
```

Здесь Control - это имя средства управления, где генерируется событие, а Event тип события. Некоторые средства управления имеют предопределенные события. Наиболее полезное предопределенное событие для клавиши, например, это Click.

В Gambas при двойном щелчке по любому средству управления редактор кода открывается и курсор позиционируется на объявлении подпрограммы для предопределенного события. Есть исключение, если управление ассоциировано с группой действия (свойство Group определено), редактор кода показывает подпрограмму для группы действия, как упоминалось выше.

Конструирование форм

- Следует понимать при конструировании форм:
- Экран пользователя может отличаться от нашего: другое разрешение, оконный менеджер и/или размер шрифта. Не пытайтесь предельно использовать все пространство, этикетки. клавиши и другие средства управления могут обрезаться или быть неразборчивы.
- Хорошая практика оставлять основное окно программы «растягиваемым». В Gambas обратите внимание на свойство формы Border - рамка. Не устанавливайте его в Fixed - фиксировано.
- При создании новой формы есть несколько интересных опций:



Опции, относящиеся к конструктору и деструктуру (constructor и destructor), полезны для инициализации и завершения задачи в окне.

Генерируются следующие определения:

```
' Gambas class
file PUBLIC SUB _new()
END
PUBLIC SUB _free()
END
PUBLIC SUB Form_Open()
END
```

Если мы выберем «Static constructor - статический конструктор» и/или «Static destructor - статический деструктор», определения станут:

```
' Gambas class file
STATIC PUBLIC SUB _init()
END
STATIC PUBLIC SUB _exit()
END
PUBLIC SUB _new()
END
PUBLIC SUB _free()
END
PUBLIC SUB Form_Open()
END
```

Используя эти процедуры, мы можем изменить процесс открывания закрывания окна. Мы можем инициализировать средства управления, переменные и т.д. В процедуре, объявленной как STATIC, процедура имеет доступ только к STATIC переменным.

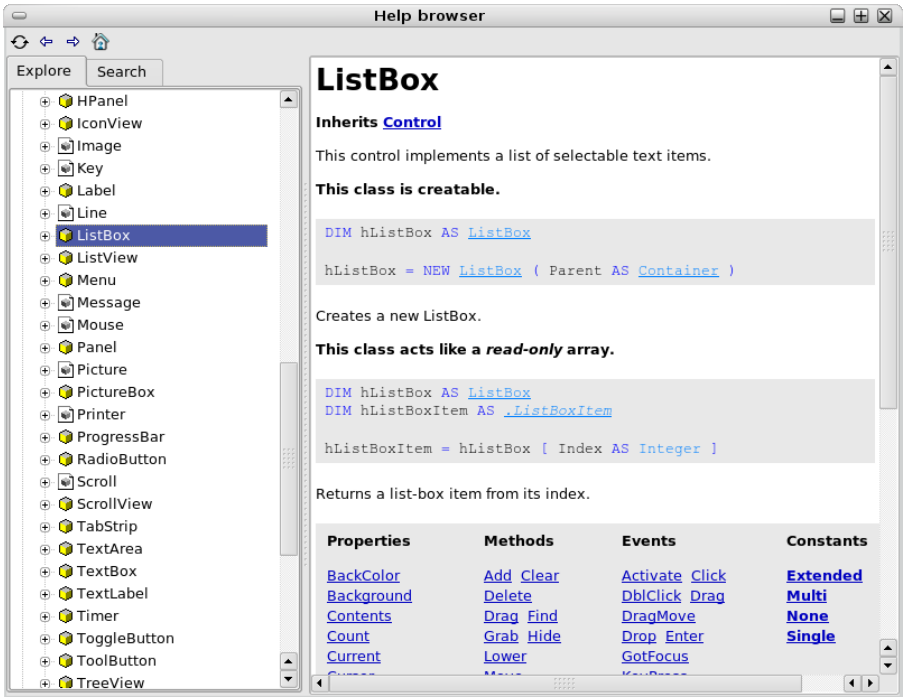
Прыжок в ...

Наша форма полностью разработана. Пришло время написать код для управления.

Первая задача - обеспечить некоторую функциональность управлению. Мы собираемся поработать с клавишами (и равнозначными пунктами меню) Add - добавить, Modify - изменить, Delete - удалить и Clean – очистить.

Действие Clean

Эта клавиша должна удалять все записи в ListBox. Для выполнения этой задачи мы поищем в системе помощи (help system) документацию, относящуюся к ListBox:



Эта документация в «дереве» под пунктом gb.qt, компонент Gambas, который включает все «видимые» средства управления (клавиши, этикетки, меню и т.д.). Прочитаем, что ListBox предоставляет метод Clean, который очищает все вводы. Это все, что мы хотели, и мы с удовольствием воспользуемся этим методом.

Двойной щелчок по клавише **Очистить** (или пункту меню «Очистить»), открывается редактор кода и курсор позиционируется на соответствующей процедуре. Мы запишем следующий код:

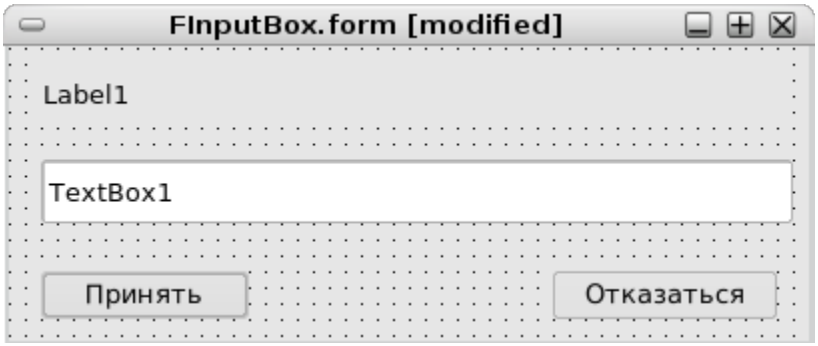
```
PUBLIC SUB Clean_Click()
ListBox1.Clean()
END
```

Очень легко :−)

Действие Add

Это действие более сложное. Пользователи будут добавлять ввод (строку текста) в ListBox, используя эту клавишу.

Gambas не предоставляет диалог типа «InputBox», так что мы собираемся создать нечто в этом роде. Мы создадим новую Form, но теперь мы зададим, что форма имеет конструктор. Почему? Потому, что в создаваемом образце мы хотим задать некоторые свойства формы, такие как заголовок, показ сообщений и определенное значение в поле текстового ввода. Вот предлагаемая конструкция:



Форма очень проста. Она имеет Label, текстовый ввод (TextBox) и две клавиши, **Accept** - принять и **Cancel** - отказаться. Разумный диалог предполагает наличие удобной для пользователя возможности отказаться с помощью клавиши **Escape** и принять с помощью клавиши **Enter**:

> Button управление имеет свойства, названные *Default* и *Cancel*. Мы установим Default в True для клавиши **Принять** и Cancel свойство в True для клавиши **Отказаться**.

Применение этих свойств приведет к тому, что когда пользователь нажмет клавишу <ENTER>, форма поведет себя также, как если бы пользователь щелкнул по клавише "Принять", а нажатие на <ESC> будет эквивалентно нажатию на клавишу **Отказаться**.

Следующая проблема, как вернуть написанный текст в диалоге в основное окно. Мы должны помнить, что в Gambas нет глобальных переменных, так что нам следует поискать другое решение. В «Tip of the day» №7, (пункт меню «? > Tips of the day - подсказки дня») есть предложение использовать переменную, объявленную как PUBLIC, так что эта переменная видима из любой точки или класса внутри программы,

Создадим новый модуль (правый щелчок по Modules > New module) и мы назовем этот модуль MCommon, например. Вот реализация модуля:

```
' Gambas module file
PUBLIC my_text AS String
```

Очень просто и легко. Теперь мы имеем переменную, которая может быть доступна из любой точки внутри программы при использовании следующей нотации: MCommon.my_text

Теперь мы запишем код для нашего диалога InputBox:

```
' Gambas class file
PUBLIC SUB _new(title AS String, message AS String, OPTIONAL text
AS String)
      ME.Caption = title
      Label1.Caption = message
      ' a String is evaluated to FALSE if it is empty:
      IF text THEN TextBox1.Text = text
END
PUBLIC SUB Button1_Click() ' This is the "Accept" button
      MCommon.my_text = TextBox1.Text
      ME.Close(0)
END
PUBLIC SUB Button2_Click() ' This is the "Cancel" button
      ME.Close(0)
END
```

Процедура _new - это конструктор. Используя его, мы можем устанавливать разные заголовки, этикетки и содержание текстовых вводов каждый раз, когда используется диалог. Более того, эти свойства устанавливаются в момент создания.

Клавиша **Принять** копирует текст в TextBox в переменную my_text, определенную в модуле MCommon и закрывает диалог. Клавиша **Отказаться** просто закрывает диалог.

Поскольку переменная MCommon.my_text в общем пользовании, мы должны помнить о необходимости clear - очистить каждый раз при ее использовании. Мы увидим это сейчас.

Процедура для клавиши **Добавить - Add** в основной форме снабжена следующим кодом. Он хорошо прокомментирован:

```
PUBLIC SUB Add_Click()
      ' Declarating our "Inputbox"
      f AS FInputBox
      ' We create the InputBox, setting the title, message
      ' and a default value: the system date and time
      ' and a small arrow
      f = NEW FInputBox("Write an entry", "Please, write here the entry to be
added:",
      CStr(Now) & " -> ")
      ' We show the InputBox
      f.ShowModal()
      ' If the user has written some text, it will
      ' be in the shared variable MCommon.my_text
```

```
      IF MCommon.my_text THEN ' An empty string is False
      ' The ListBox control has a method for adding entries: Add()
      ListBox1.Add(MCommon.my_text)
      ' We "empty" the shared variable
      MCommon.my_text = ""
      END IF
END
```

Действие Modify

Применяя эту клавишу пользователь может изменить ввод в ListBox. Если записей нет, клавиша ничего не делает, а, если пользователь не выбрал запись, он будет предупрежден об этом. Вот реализация для этого действия:

```
' "Modify" action
PUBLIC SUB Modify_Click()
      f AS FInputBox
      IF ListBox1.Count > 0 THEN ' If the ListBox is empty, its property
      ' Count is 0
      IF ListBox1.Index = -1 THEN
      ' The Index property returns the index for the selected entry.
      ' It there is not selected line, it returns -1.
      ' We warn the user.
      message.Info("You must select the line to modify.")
      ELSE
      ' The user has selected a valid entry.
      ' We show our InputBox with the text of the selected entry.
      ' The selected text is the property Text of the
      ' selected object ListBoxItem
      ' which is accesible through the property
      ' Selected of the ListBox
      f = NEW FInputBox("Modify entry", "Please, modify the selected entry:",
ListBox1.Current.Text)
      f.ShowModal()
      ' The dialog box FInputBox changes the shared variable
      ' MCommon.my_text
      ' If MCommon.my_text is not empty, we load it in the
      ' selected ListBoxItem
      IF MCommon.my_text THEN ListBox1.Current.Text = MCommon.my_text
      ' We "empty" the shared variable after its use
      MCommon.my_text = ""
      END IF
END
```

Действие Delete

Как мы видели выше, ListBox должен содержать хотя бы одну строку, а пользователь должен выделить одну строку.

Код очень похож на тот, что для действия Modify - изменить:

```
PUBLIC SUB Delete_Click()
      i AS Integer
      i = ListBox1.Index
      IF i >= 0 THEN
      ListBox1.Remove(i) ' The Remove method erases the selected line
      ELSE IF ListBox1.Count > 0 AND i = -1 THEN
      ' We check that the ListBox is not empty and
      ' that some entry is selected.
      message.Info("You must select the desired line to delete.")
      END IF
END
```

Можно видеть, что реализация для этих четырех действий общая для клавиш и их эквивалентов в меню.

Теперь мы реализуем действие, относящееся к управлению файлами (Open - открыть, Save - сохранить) и закрытию программы. Мы начнем с того, что попроще.

Действие Exit

Функция для этой клавиши (и ассоциированного пункта меню) - закрыть программу. Очень просто:

```
PUBLIC SUB Exit_Click()
      ME.Close(0) ' ME is a reference to the form itself
      FInputBox
END
```

Мы можем проявить больше дружелюбия к пользователю, добавив диалог наподобие: «Вы собираетесь выйти из программы. Вы уверены?», – и создав подходящее задание. Но, давайте, оставим эту реализацию в качестве домашнего задания для читателей.

Действие Open

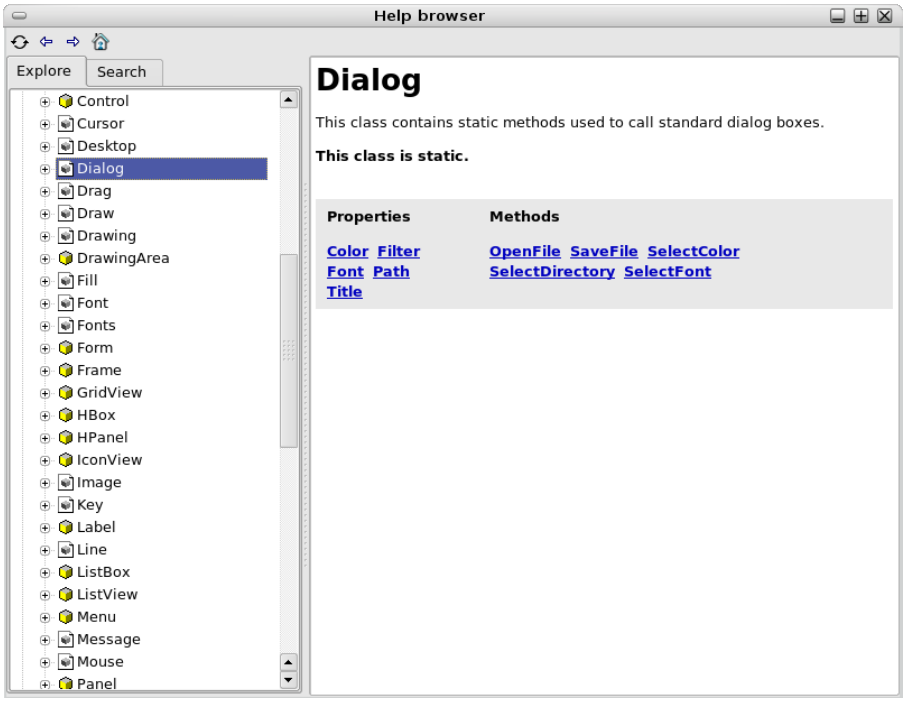
Что это действие должно сделать? Выяснить у пользователя путь к файлу, прочитать этот файл и загрузить данные в ListBox. Вот соответствующий код:

```
PUBLIC SUB Open_Click()
      DIM lin AS String
      DIM arr_strings AS String[]
      Dialog.Title = "Please select a file"
```

```
    Dialog.Filter = [ "Minder data (*.data)", "All files (*.*)" ]
    IF NOT Dialog.OpenFile() THEN
    arr_strings = Split(File.LOAD(Dialog.Path), "\n")
    ListBox1.Clean()
    FOR EACH lin IN arr_strings
    ListBox1.Add(lin)
    NEXT
    END IF
END
```

Эта часть кода представляет интересную особенность языка Gambas, «non instanceable - не запрашиваемые» или статические классы (2). Мы не можем создавать объекты этих классов, но мы можем прямо использовать их. В этом коде мы можем видеть два из этих классов в действии: класс File и класс Dialog.

Например, класс Dialog предоставляет доступ к типичному стандартному диалогу для выбора файлов, системных цветов и т.д. Это документировано в теме help, gb.qt



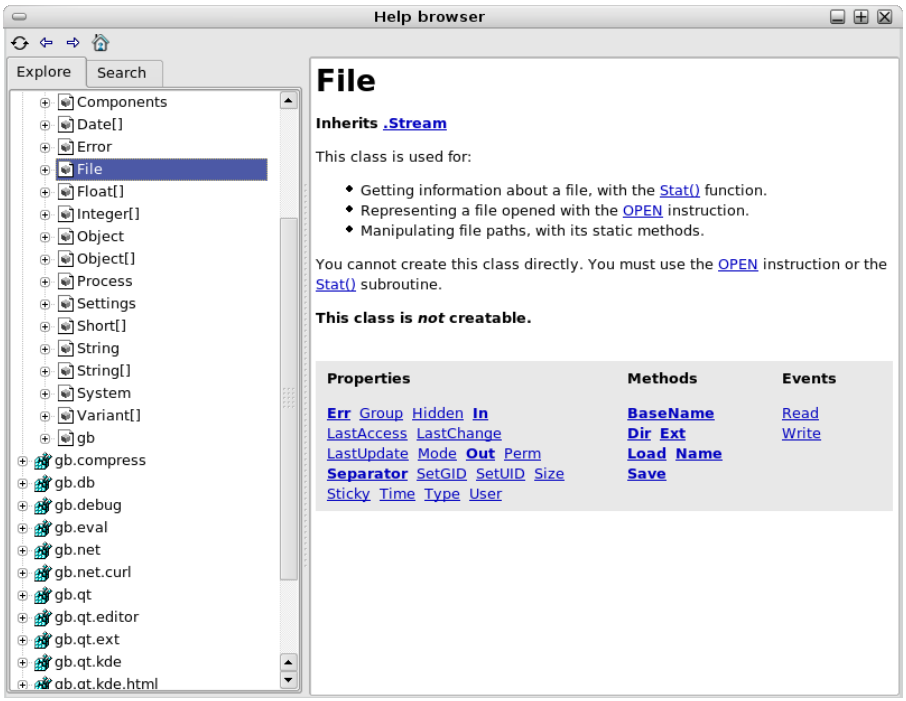
В нашей программе мы хотим выбрать файл и загрузить его содержимое в ListBox. Мы будем использовать класс Dialog следующим образом:

```
    Dialog.Title = "Please select a file"
    Dialog.Filter = [ "Minder data (*.data)", "All files (*.*)" ]
    IF NOT Dialog.OpenFile() THEN
    ' etc ...
```

Мы задали заголовок диалога, предоставили фильтр для выбора типа файла по расширению

(*.data) и, наконец, мы вызвали метод класса OpenFile(). Есть особенность в класее Dialog: если пользователь НЕ выбрал файл (то есть, пользователь нажал клавишу отказа (cancel) или клавишу ESC) возвращаемое значение в методе OpenFile() становится True. Когда же пользователь выбирает файл, мы можем получить доступ к полному пути через свойство Dialog.Path

Класс File (его документация под пунктом gb в системе помощи) предоставляет несколько методов, позволяющих нам работать с файлами:



В документации к Gambas в разделе, озаглавленном «How do I ...» есть несколько примеров кода для чтения и записи файлов. Мы будем использовать метод Load(), чьими аргументами являются путь к файлу и возвращаемая String,которая содержит все данные в этом файле. Мы можем разделить строки возвращаемых данных, используя функцию Split(). Ее аргументы - это строки, которые следует разделить и разделитель (символ новой строки в данном случае \n) и возвращаемый Array of Strings. Из этих соображений мы должны объявить переменную arr_strings as String[]:

```
    DIM arr_strings AS String[]
```

Когда мы получим все строки данных, содержащиеся в файле, мы должны очистить ListBox и добавить каждую строку, используя метод Add() ListBox.

Действие Save

Когда пользователь нажимает клавишу **Save - сохранить**, программа должна вывести содержимое ListBox в текстовый файл. Мы покажем диалог Save File, запрашивая у пользователя имя файла для сохранения данных. Вот код для этого действия:

```
PUBLIC SUB Save_Click()
    lines AS String
```

```
      destination AS String
      numFile AS Integer
      lines = ListBox1.Contents
      Dialog.Title = "Please, select a file"
      Dialog.Filter = [ "Minder data (*.data)" ]
      IF NOT Dialog.SaveFile() THEN
      IF Right$(Dialog.Path, 5) <> ".data" THEN
      destination = Dialog.Path & ".data"
      ELSE
      destination = Dialog.Path
      END IF
      File.Save(destination, lines)
      END IF
END
```

Мы хотим сохранить данные в файле с расширением .data, так что, если имя файла предоставленное пользователем не заканчивается с ".data", мы будем дописывать расширение. Мы используем метод Save(), который предоставляется классом File.

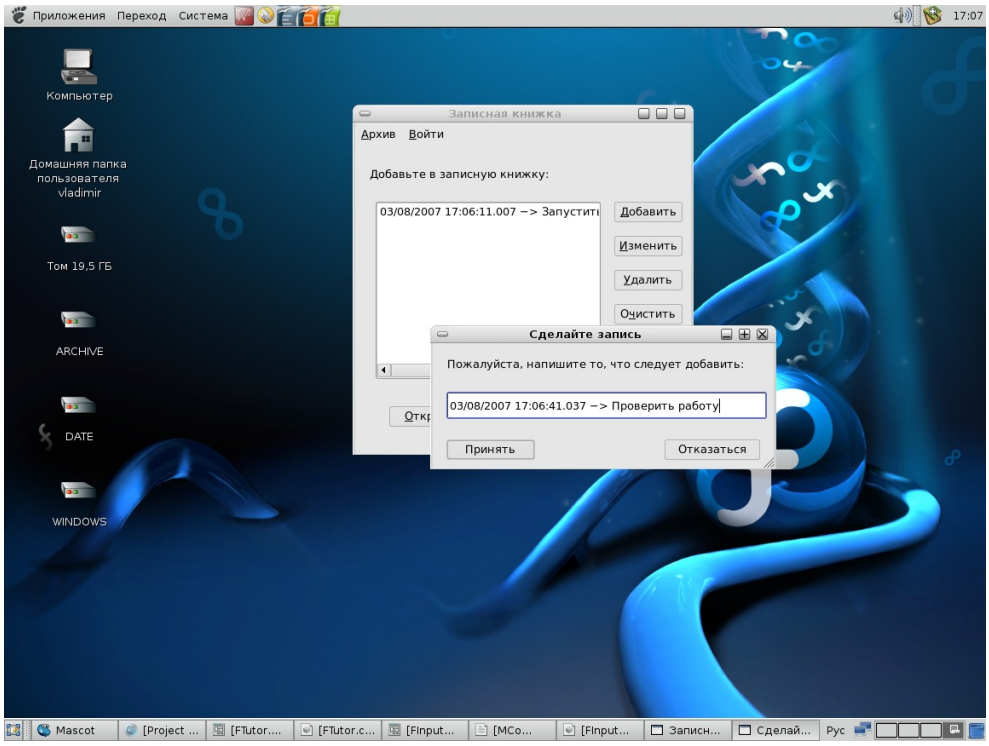
Аргументы этого метода - путь к файлу и текст, который мы хотим сохранить. Мы получаем доступ к содержимому ListBox, используя свойство Contents, которое возвращает String, с «new line» (\n) разделителем каждого ввода в ListBox.

Окончательная подгонка

Пользователю интересно увидеть весь текст записи, а listbox может оказаться не достаточно широким, чтобы отобразить его, если запись очень длинная. Мы реализуем эту возможность следующим образом: когда пользователь дважды щелкает клавишей мышки по вводу, мы покажем весь текст в окне диалога:

```
PUBLIC SUB ListBox1_DblClick()
      IF ListBox1.Index >= 0 THEN
      message.Info(ListBox1.Current.Text)
      END IF
END
```

Наша программа запущена



Развертывание завершения программы

OK. Наша программа готова. Мы можем проверить ее в любое время внутри IDE, нажав на клавишу F5.

Теперь мы хотим использовать программу, как обычное приложение, то есть, без IDE. Для выполнения этого есть опция основного меню: Project > Create executable. Этим создается «монолитный» исполняемый файл, включающий все ресурсы проекта. Этот файл не будет машинным кодом, но «байт-кодом», выполняемым интерпретатором Gambas – gbx. Это предполагает, что мы нуждаемся, чтобы этот интерпретатор был установлен в системе для запуска программ, написанных в Gambas. (Это похоже на другие языки программирования. Например, нам нужно установить Java для выполнения программ, написанных на Java).

Все дистрибутивы, в которые включен Gambas, компоненты Gambas разделены и есть пакет «Gambas runtime», который включает интерпретатор, но не полный IDE.

Мы можем создать RPM или DEB архив нашей программы. Эти пакеты будут иметь интерпретатор Gambas (gambas−runtime программа) по зависимости. Есть мастер для помощи в генерации такого пакета. Он очень прост и интуитивно понятен. Вы можете найти его в меню Project > Create installation package ...

Окончание

Мы убедились, что это очень легко создавать простые, но функциональные приложения в Gambas. Этот инструмент предлагает некоторые средства управления и предопределенные классы. Есть также множество расширений или компонентов, доступных в порядке создания клиент/серверных приложений, доступа к базам данных, приложений мультимедиа и т.д.

IMHO, Я думаю, что Gambas очень многообещающий инструмент. К счастью, Gambas' mailing−list разработчиков очень активен, а ошибки устраняются очень быстро.

Спасибо, Benoit (et col.)! Хорошая работа!

Об этом документе и его авторе

Как говорилось выше, мы работали с версией 1.0−1 Gambas в этом руководстве (предварительно скомпилированный пакет для Debian Sid). Во время написания этого документа была выпущена версия 1.0.3, и весьма вероятно, что в момент прочтения этого, появится новая версия. Весьма благоразумно почитать журнал изменений, чтобы избежать возможных расхождений от версии к версии.

Все комментарии, предложения, коррективы и улучшения этого документа приветствуются.

Мой mail – forodejazz@gmail.com

Примечания:

- События нуждаются в поддержке процедуры или подпрограммы: функция, которая не возвращает какого-либо значения.

- Я не эксперт в технологии объектно-ориентированного программирования. Мои извинения, если я использовал некорректные термины ;−)

Часть 3. Proteus в любительской практике

Завершив предыдущую книгу, я понял, что рассказал все, о чем хотел рассказать, и решил оставить сочинительство, ну, нельзя же без конца твердить об одном и том же. Вследствие чего будем считать эту книгу «виртуальной» — она как бы есть, а как бы ее и нет.

В своих книгах я старался избегать рассказов о профессиональных и коммерческих программах, равно как и об удобном, но дорогостоящем оборудовании. Причин тому несколько. Во-первых, я пишу для любителей, более того для начинающих, и еще лет пять назад мне казалось бестактным рассказывать о недоступных по цене средствах освоения и изучения электроники, и я старался рассказывать о бесплатных или доступных по цене инструментах. Во-вторых, я уже много лет использую практически только операционную систему Linux, и не нахожу причин, по которым следовало бы искать ей замену. А в Linux, свободно распространяемой операционной системе, такой обширный набор программ, что есть о чем рассказать.

Однако время идет, благосостояние граждан растет, и сегодня многим любителям доступны коммерческие программные продукты, которые могут им и нравится больше, и привлекать тем, что используются профессионалами.

Предыдущие книги, написанные мною, представляют по сути эксперименты в жанре рассказов, повестей и романов «в технической прозе». Мне приходилось для рассказа об электронике придумывать и сюжет книги, и искать героев повествования, и самому становится придуманным мною автором. Мне казалось, что увидеть электронику глазами начинающего любителя, можно только став, хотя бы отчасти, любителем, проходящим шаг за шагом, от ошибки к ее пониманию, весь путь начинающего. Не думаю, что справился с задачей, но я пытался это сделать.

Отчасти неосознанно, отчасти сознательно, я избегал излишне тщательного «разжевывания» основных понятий и терминов, поскольку мне казалось, что этим я могу обидеть читателя. Я старался вместо решений дать некий набросок, путь к цели, и предоставить читателю самому во всем разобраться, а мою помощь свести к рассказу о программах, полезных в любительской практике, об операционной системе Linux, с которой многие радиолюбители знакомы по наслышке, и к попыткам убедить читателя, что занятие электроникой не только доступно всем, но и увлекательно не менее компьютерных игр или рыбной ловли. Прямо или косвенно мои рассказы строились на тех вопросах, которые задавали начинающие на форумах радиолюбительских сайтов. Как правило, бывалые любители или профессионалы стараются помочь начинающим, но не могут в двух словах ответить на любой вопрос, даже, казалось бы, простой, и в этом случае они вполне разумно рекомендуют почитать книги, в которых все подробно описано. Таких книг, великолепных книг, написанных ранее и изданных недавно, действительно великое множество, но их не всегда найдешь в книжных магазинах, да и чтобы в книге найти ответ на интересующий тебя вопрос, нужно уметь искать, а это умение приходит с опытом, которого у начинающего может не быть. Он появится, но к тому времени, когда начинающий станет опытным или, как сейчас модно говорить, «продвинутым» радиолюбителем, а для этого надо двигаться вперед. А чтобы двигаться вперед, нужно паять и настраивать схемы, нужно читать книги, нужно искать друзей в сообществе радиолюбителей и обсуждать с ними свои реальные или надуманные проблемы, нужно, и важно, задавать вопросы и получать ответы на них.

Многие вопросы, которые я встретил на любительских сайтах, на первый взгляд казались мне столь очевидны, вернее, очевидными казались ответы на них, что я не обращал внимания на эти вопросы. Но сегодня, желая восполнить пробелы, оставленные в прежних

рассказах, поразмыслив над этими простыми вопросами, я пришел к выводу, что очень ошибался, когда считал их не заслуживающими внимания.

Простые вопросы и скучные ответы на них — вот содержание этой книги.

Что такое интегрирующая и дифференцирующая цепь?

Небольшое отступление, как предисловие к этой главе — Proteus это коммерческая программа класса EDA, упоминание о которой я встречал чаще в сочетании с программированием AVR микроконтроллеров. Но программа позволяет работать не только с микроконтроллерами, но и с аналоговыми, и с цифровыми устройствами. Поэтому все схемы, о которых пойдет речь далее, можно проверить в этой программе.

Так что такое интегрирующая и дифференцирующая цепь? Тысячи раз — не считал, но думаю, не меньше — я встречал эти термины, не задерживаясь на них. Видимо, не реже употреблял их, не задумываясь об их смысле. Если бы не встреченный мною вопрос, мог бы гордо пробежать мимо еще не одну тысячу раз. А есть ли, чем гордиться?

Интегрирующая цепь — здесь два слова, два понятия, и базовым я определил бы слово «цепь», подразумевающее электрическую цепь, то есть, несколько электрических элементов, соединенных некоторым образом друг с другом. Конечно же, можно любые элементы электрической схемы соединять произвольно, не заботясь о правильности подобного соединения, при этом результат образует некую электрическую цепь, которая может работать, может не работать, но останется электрической цепью. Но на практике электрические цепи образуют в совокупности некоторое полезное устройство, а потому элементы любой схемы имеют назначение и добавляются в схему осмысленно. Формально при производстве стараются минимизировать количество элементов любой электрической схемы, но производство дело тонкое, и не всегда формальный подход — наилучший путь к успеху. Специализация, а сегодня бессмысленно пытаться обойтись без нее, приводит к тому, что инженеры разных специальностей даже в одной области деятельности могут работать совершенно по-разному. В первую очередь это относится к расчетам электрических схем. Как часто приходится инженеру этим заниматься, зависит не только от его специализации, но и от предприятия, на котором он работает. На одних предприятиях существуют отлаженные методики расчетов, которые с появлением компьютеров сводятся для специалиста к подстановке рекомендованных параметров, на других предприятиях предпочитают использовать типовые решения, а где-то для расчетов набирают группы профессионалов-математиков. И хотя каждый инженер в студенческие годы изучает и математику, и физику, далеко не каждый инженер каждодневно использует полученные знания в своей работе. Важным для него остается то, что он привыкает к терминологии своей сферы деятельности, и каждое понятие, встречаемое им или используемое им, в той или иной мере ему знакомо. Подобно сороконожке, если он будет задумываться над каждым понятием, глубоко и серьезно, он никогда не сможет выполнить свою работу. Видимо по этой причине, встретив вопрос любителя, он посоветует ему поискать ответ в соответствующем учебнике.

У каждого из нас есть свои предпочтения в части учебников. Я не исключение. Встретив вопрос, ставший названием этой главы, я заглянул в один учебник, в другой, полистал справочники и учебник по математике и понял, что вопрос гораздо интереснее, чем может показаться на первый взгляд. Если ко второй составляющей термина можно быстро подобрать, пусть не исчерпывающий, но приемлемый ответ, то объяснить, что такое «интегрирующая» по отношению к электрической цепи, сложнее. Понятно, что в основе лежит такое математическое понятие, как интеграл. Математика очень хорошо организованная наука, а язык математики настолько удобен, что сегодня им охотно пользуются не только в технических, но и в гуманитарных науках, однако полновесный ответ

на вопрос, что такое интеграл, может потребовать ответа в объеме целой книги.

Можно, конечно, привести пример простейшей интегрирующей цепочки, сказав — вот подобные электрические цепи и называют интегрирующими. Такой ответ я сейчас проиллюстрирую.

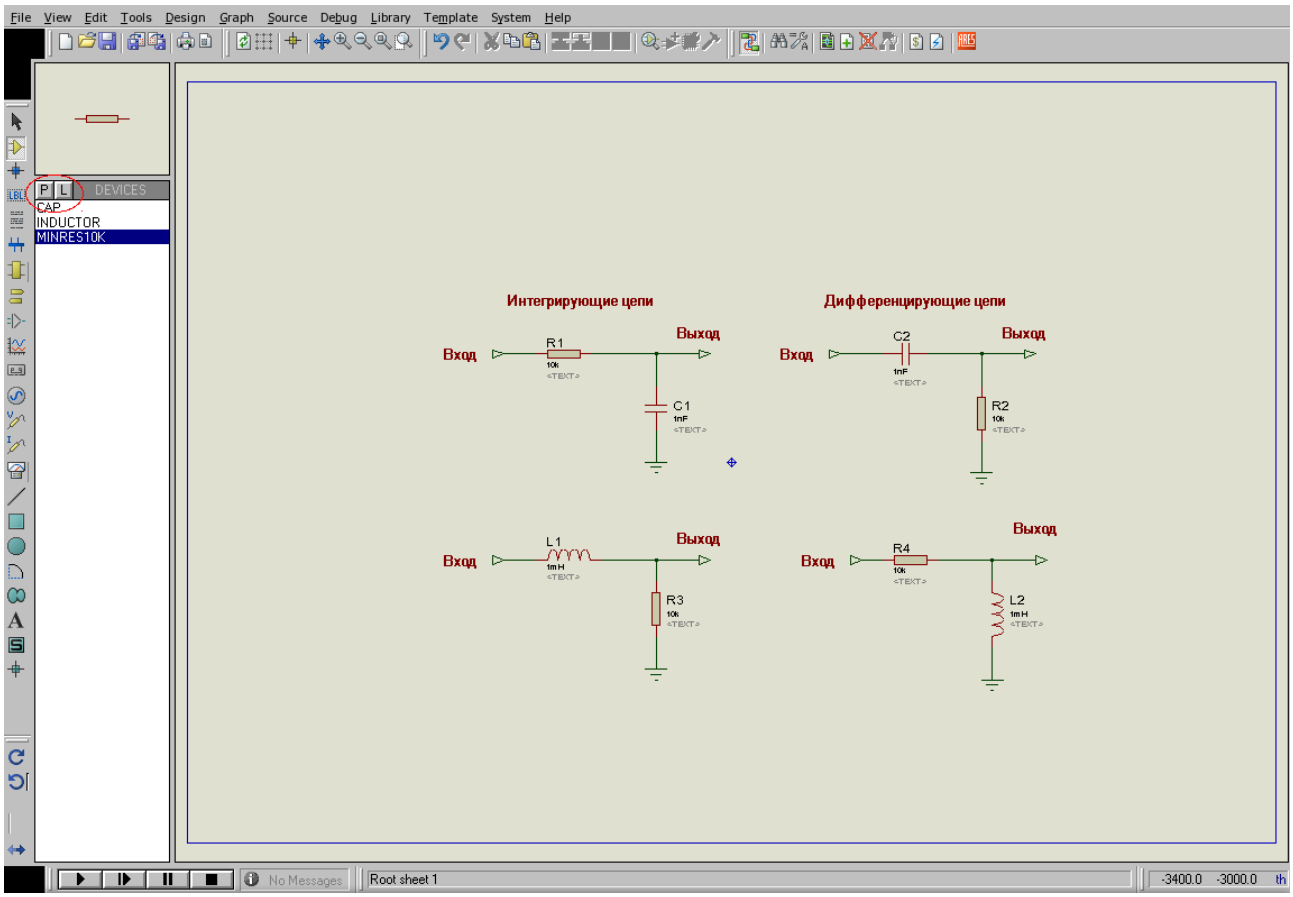


Рис. 1.1. Пример интегрирующей и дифференцирующей цепей

Это не единственные примеры цепей, но остановимся пока на них, чтобы попытаться соотнести то математическое и то электротехническое, что есть в понятии интегрирующей цепи. Неопределенный интеграл от некоторой функции можно записать следующим образом:

$$\int f(x)dx = F(x) + C,$$

где $F(x)$, если она существует, это функция, производная от которой будет подынтегральной функцией, а C — неопределенная постоянная.

И со сделанным оговорками это не вполне строгое определение, которое отсылает нас к поиску определения производной функции. То есть, если мы можем найти функцию $F(x)$, продифференцировав которую получим функцию $f(x)$, то мы сможем сказать нечто более определенное об интеграле от этой функции.

Решив, что я запутался сам и могу запутать читателя, я отложил в сторону справочники по математике и обратился к учебнику, по которому некогда учился сам, по теоретическим основам электротехники. Уж там-то я быстро найду ответ на вопрос.

К моему большому удивлению, просмотрев оглавление, я обнаружил, что первое упоминание относящееся к вопросу, который меня интересует, встречается, примерно, в середине весьма пухлого тома и описывается следующим образом, относительно

интегрирующей цепи, изображенной выше.

> *...напряжение на выходе при определенных условиях пропорционально интегралу от входного напряжения... Uвых(t) =* $1/RC \int Uвх(t)\,dt$

И далее следует ряд условий. В итоге я готов признать, что для понимания ответа на столь, казалось бы простой вопрос, вопрошающему следует знать определенный объем математики, и значительный объем теоретической электротехники. При всем моем уважении и к математике, и к электротехнике, и к знаниям вообще, я не готов подвергнуть не только начинающего, но даже опытного любителя таким испытаниям.

Мало того, если я посоветую подойти к вопросу чисто практически, то есть, взять сопротивление, конденсатор, макетную плату и собрать схему, то что можно наблюдать в этой схеме при использовании обычных приборов?

Наиболее часто используемые в электротехнической практике источники напряжения это постоянное напряжение, синусоидальное и импульсное напряжение. Оставив в стороне последнее, посмотрим, что бы мы увидели в экспериментах с первыми двумя источниками. Кстати, я хотел бы немного рассказать о начальном этапе работы с программой Proteus.

Я использую эту программу (версии 7.2) в операционной системе Linux, где она работает с помощью эмулятора Wine. Редактор электрических схем называется ISIS. После запуска он доброжелательно предлагает посмотреть множество примеров электрических схем, построенных в этой программе. Если вы только начинаете пользоваться программой, не пренебрегайте такой возможностью. С программой приходит много интересных примеров, которые помогут понять и увидеть, как работает программа.

Если в основном меню, в разделе **File** выбрать **New Design...**, то можно выбрать размер будущего листа проекта.

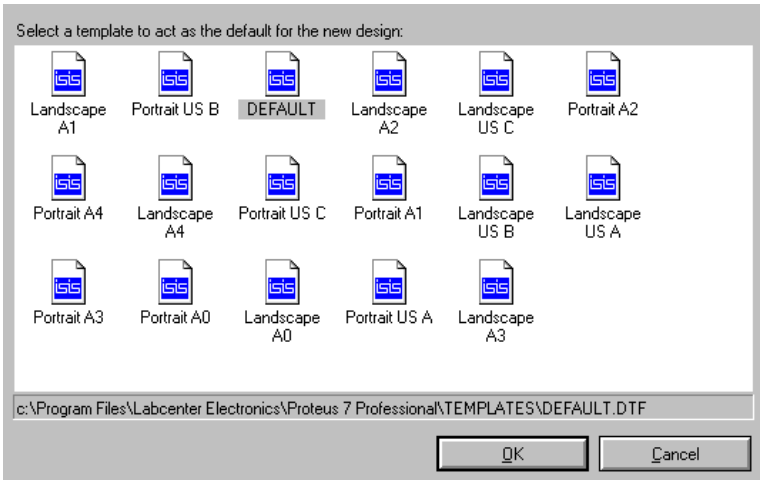


Рис. 1.2. Начало работы над проектом в программе Proteus

Для размещения элементов схемы, если вы начинаете самостоятельную работу, можно воспользоваться инструментальным меню в левой части экрана. Клавиша с изображением микросхемы открывает путь к компонентам, но для выбора компонента следует воспользоваться либо клавишей **P** в следующем окне под обзорным видом рабочего поля, либо клавишей **L** рядом с ней (на рис.1.1 они выделены овалом). Для чего служат эти клавиши можно увидеть в строке состояния (в нижней части экрана) при наведении курсора на эти клавиши. Первая, **P**, для установки компонента в схему, вторая для входа в менеджер библиотек. Но и нажав на клавишу **P**, вы окажетесь в среде многочисленных доступных

компонент разных производителей. Те, кто пользуется программой, даже советуют не заниматься поиском вручную, а задать нужный компонент в окне поиска. На рисунке ниже это окно обозначено словом **Keywords**. В правой части диалогового окна выбора компонент отображается графический вид, а ниже корпус компонента. Последнее, в основном, обусловлено наличием в программе, как в любой интегрированной среде разработки, второй части программы, ARES, для разводки печатных плат просмотра трехмерного вида готовой конструкции.

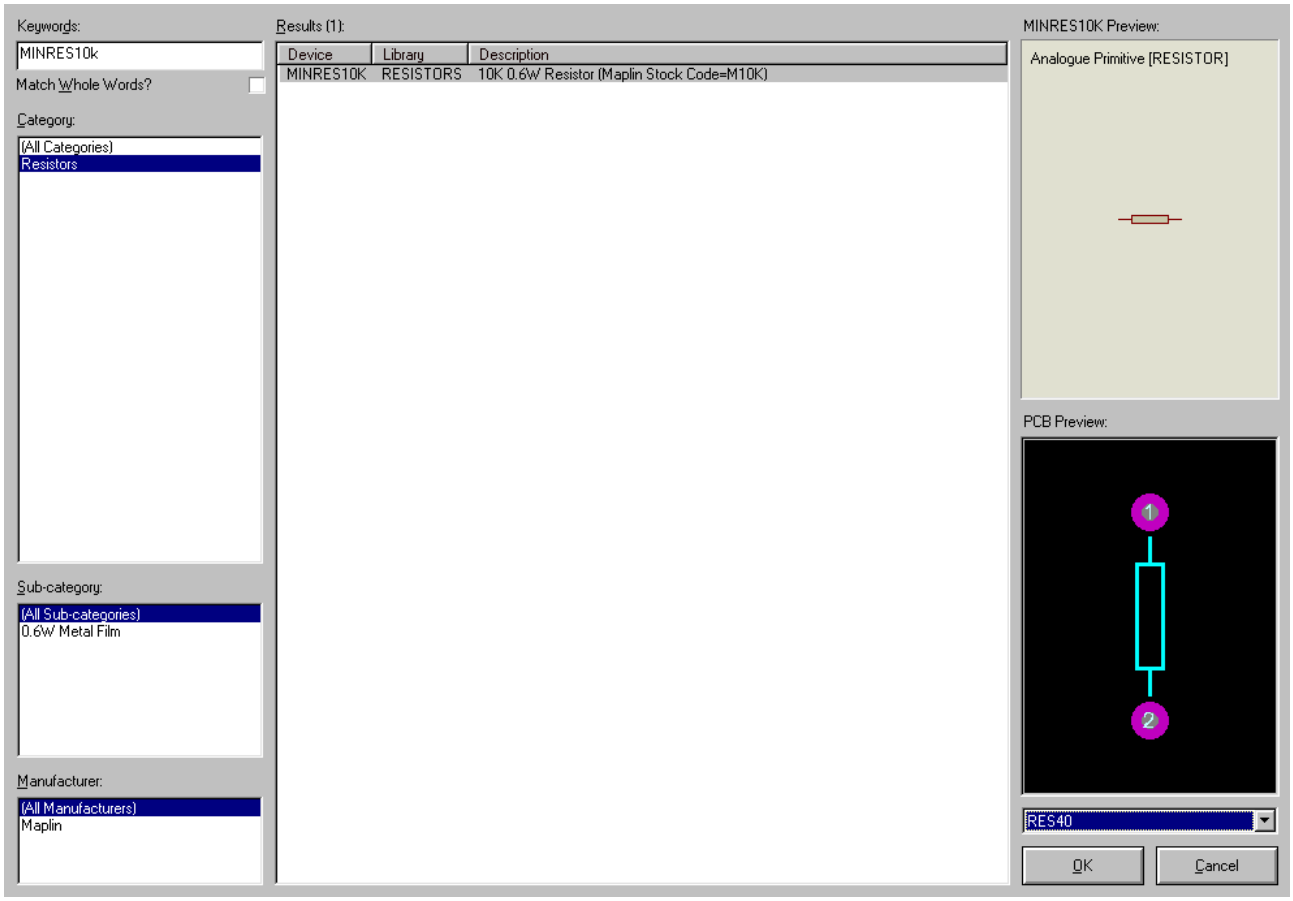


Рис.1.3. Поиск и выбор нужных компонентов схемы в программе Proteus

Если вы согласны с выбором, то клавиша **ОК** подтвердит, что можно продолжить работу. По утверждению опытных пользователей, с которым я согласен, программу можно использовать и как вполне удобный справочник, поскольку все компоненты имеют описание их назначения. Тоже приятно, согласитесь.

После выхода из окна диалога выбора вы возвращаетесь в рабочее окно, курсор принимает вид маленького карандаша, а после щелчка левой клавиши мышки на рабочем поле чертежа к курсору оказывается привязан контур элемента, который вы перемещаете в нужное место и устанавливаете повторным щелчком.

Для соединения можно использовать клавишу прорисовки линии на левом инструментальном меню. Или сразу после установки компонента в нужное место подвести курсор мышки к его выводам, курсор в этот момент имеет вид карандаша, когда он над точкой соединения, та превращается в квадратик, и можно начинать соединение.

Вернемся, однако, к испытаниям интегрирующей электрической цепи с помощью доступных нам источников сигналов. Нарисуем испытуемую цепь.

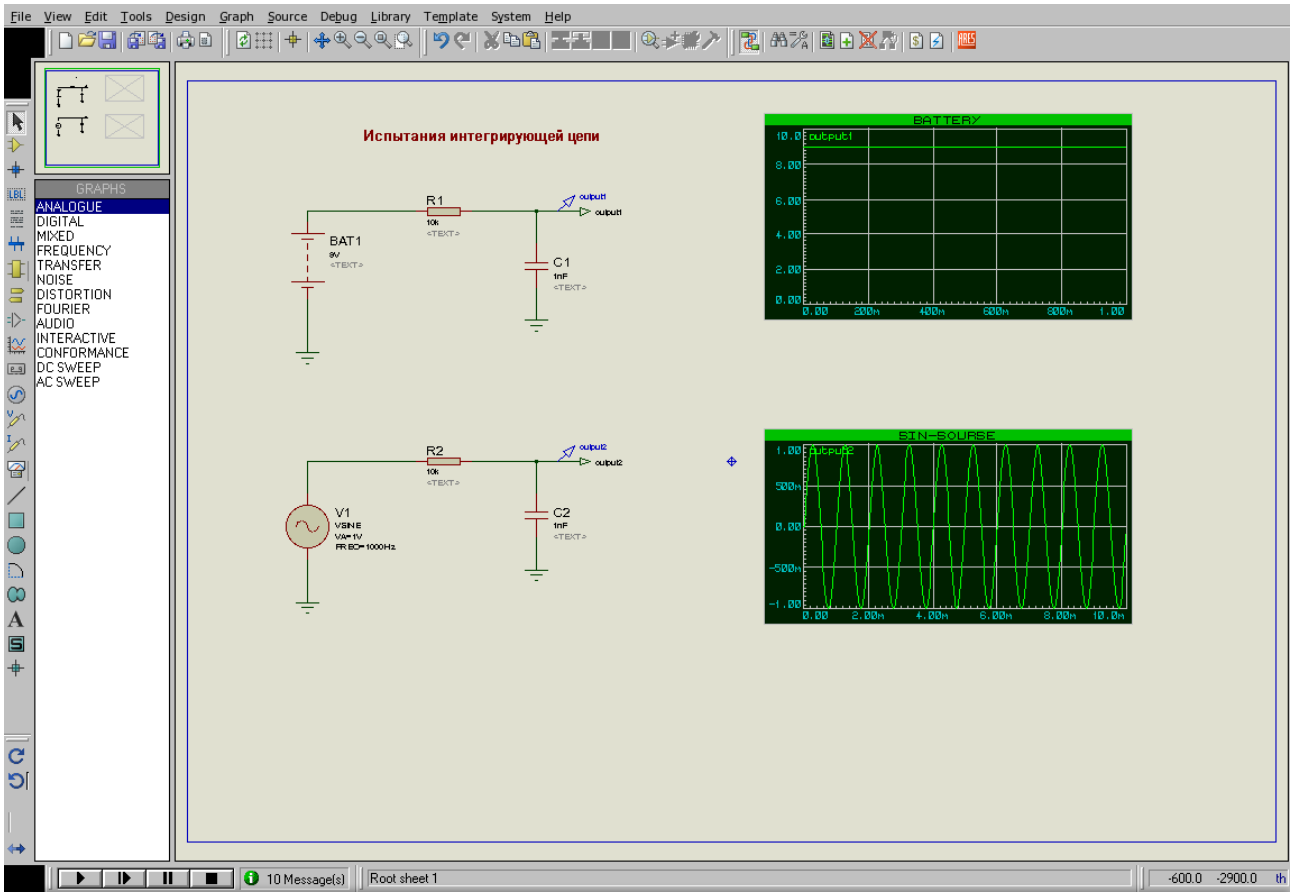


Рис. 1.4. Испытания интегрирующей цепи

Ни испытание интегрирующей цепи с помощью источника постоянного напряжения, ни испытание синусоидальным сигналом нисколько не помогают понять, чем эта цепь может отличаться от дифференцирующей. Есть еще возможность испытать цепь сигналом импульсного генератора. Но прежде мне хотелось бы рассказать, как получен предыдущий рисунок.

Чтобы не рисовать все заново, я использую проект, изображенный на рис.1.1. Удалив лишнее, я с помощью курсора выделяю часть рабочего поля содержащую интегрирующую цепь и либо правой клавишей мышки вызываю выпадающее меню, щелкнув на выделенной области, где выбираю пункт копирования в буфер обмена **Copy To Clipboard**, либо выбираю эту процедуру в разделе **Edit** основного меню, когда выделенная область подсвечена. Скопировав, я могу перенести рисунок: щелчок левой клавишей мышки на свободном месте рабочей области редактора, появившийся привязанный к курсору контур располагается удобным образом, повторный щелчок оставляет схему в редакторе.

На всякий случай я меняю обозначения резистора и конденсатора — щелчок по ним правой клавиши мышки после выделения открывает выпадающее меню с пунктом **Edit Properties** для доступа к диалоговому окну свойств компонента. В нем можно задать обозначение элемента, его значение. Осталось удалить ненужные элементы: надписи, порты, — на освободившемся месте должны расположиться источники.

При работе с редактором схем рабочее поле выглядит следующим образом (кружками я выделил клавишу входа в библиотеку компонентов и на инструментальной панели клавишу для отыскания «земли»):

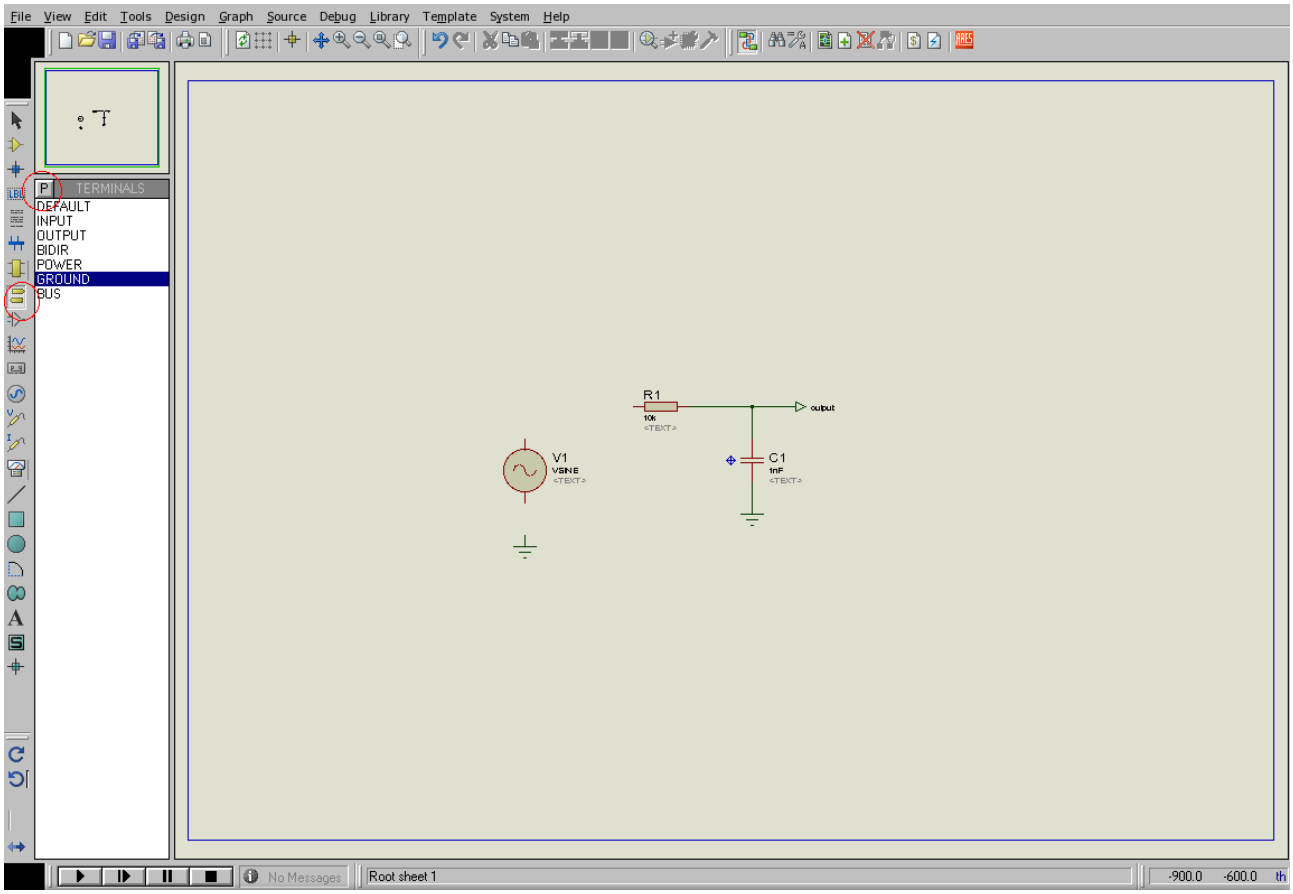


Рис. 1.5. Создание схемы в редакторе схем

К портам *output* на каждой из схем я добавляю с помощью пробника напряжения **Voltage Probe Mode** на левой инструментальной панели метки, которые обозначаю **output1** и **output2**, собственно, порты вывода я мог бы и убрать, поскольку они носили чисто декоративный характер и не нужны для работы, но они и не мешают. А метки пробника мне понадобятся в дальнейшем для графического вывода. Рядом с пробником напряжения, чуть ниже, есть и пробник тока, но информация о токе мне пока не нужна. А нужные мне источники напряжения я отыскиваю в библиотеке компонентов (клавиша **P**, затем раздел **Simulator Primitives**), эти компоненты, как и другие, имеют свойства, напряжение для батарейки и напряжение и частота для генератора синусоидального напряжения, которые можно изменить с помощью выпадающего меню, используя пункт **Edit Properties**.

Некоторые затруднения вызывает поиск «земли», обозначения общего схемного провода. Но и его удается отыскать с помощью кнопки левого инструментального меню, названной **Terminals Mode**, восьмая сверху. Метка пробника устанавливается на провод в нужном месте. Кстати, если нажимать кнопки инструментального меню, задержав курсор на них на некоторое время, то появляется надпись с назначением этой кнопки. Примерно, так:

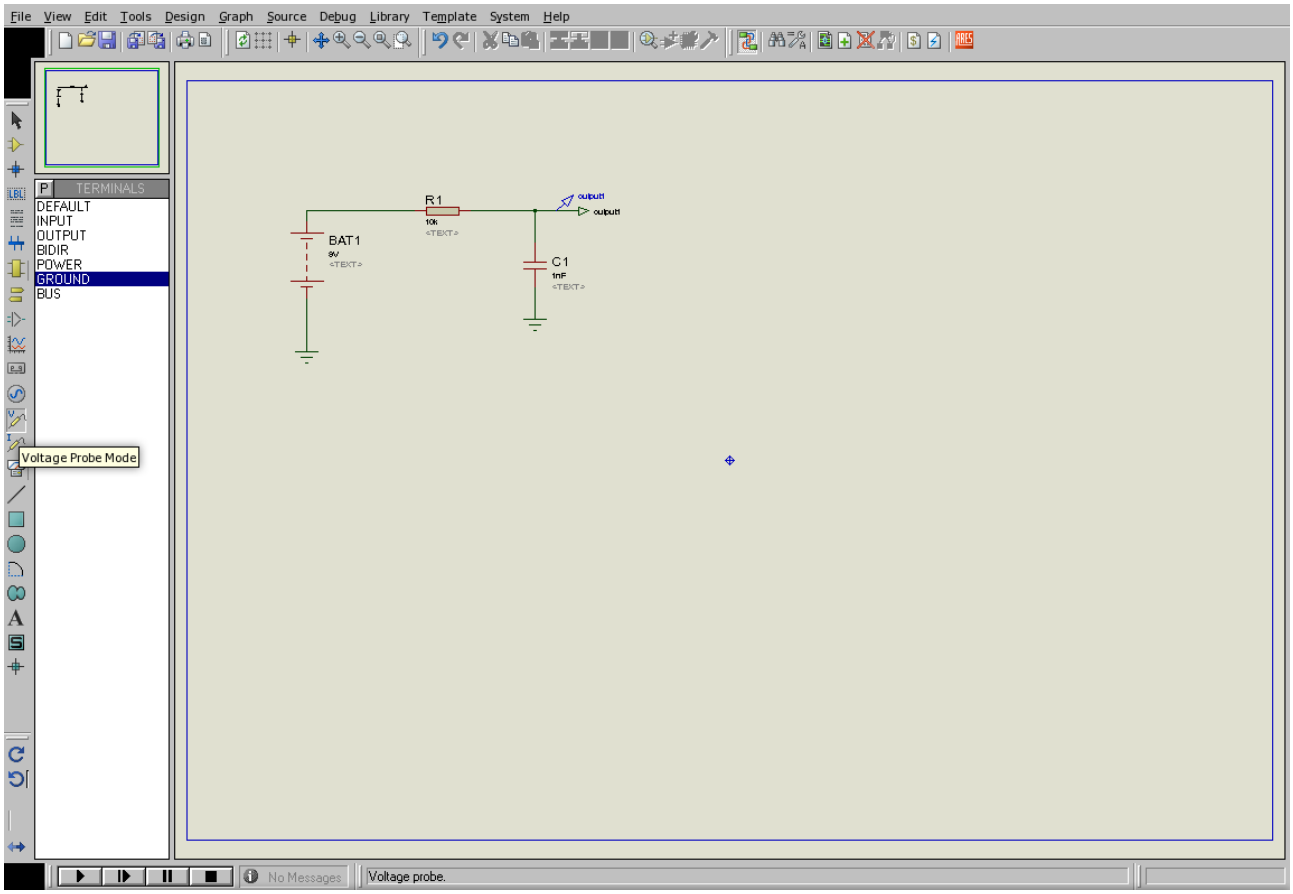


Рис. 1.6. Выбор пробника напряжения на инструментальной панели

Для наблюдения за результатами проб напряжения в правой части рабочего поля следует установить графические, скажем, окна. Десятая сверху клавиша левой инструментальной панели **Graph Mode** превращает курсор в инструмент задания размеров этого окна. Щелкнув левой клавишей мышки можно рисовать прямоугольник нужного размера, а повторный щелчок фиксирует этот размер. Диаграмма, как и другие компоненты, имеет свойства, диалоговое окно которых можно выбрать из выпадающего меню после выделения окна графика.

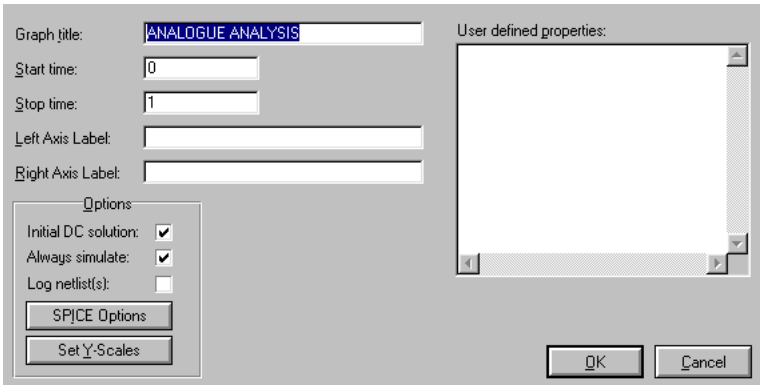


Рис. 1.7. Диалоговое окно свойств графика

Верхнее поле ввода текста **Graph Title** для названия графика, далее следует поле ввода времени начала, затем времени завершения симуляции. Здесь-то лучше задать разумное время окончания, скажем, при частоте синусоидального испытательного генератора 1 кГц

достаточно задать 1-2 мС (2ms). Программа не любит, похоже, пробелов между цифрой и единицей измерения, но ей достаточно одной буквы, например *m*, чтобы определить единицу *ms*. По умолчанию время окончания симуляции равно секунде, можно оставить его таким, но результат вас «приятно» поразит, хотя ничего страшного не произойдет. Прежде, чем получить графическое отображение результатов, вы должны быть уверены, что задали параметры испытательного сигнала, используя диалоговое окно свойств компонента VSIN. Обычная процедура задания свойств компонента через, например, выпадающее меню щелчком правой клавиши мышки по выделенному объекту.

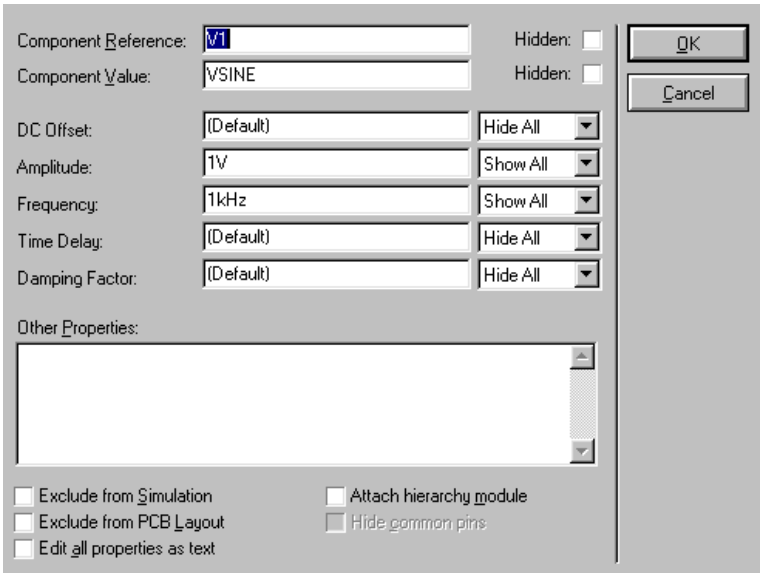


Рис. 1.8. Диалоговое окно задания параметров испытательного сигнала

Если вы хотите, чтобы параметры отображались на схеме, используйте выбор в правом окне параметра **Show All**, если нет, оставьте задаваемое по умолчанию **Hide All**.

Вы проверили параметры испытательного сигнала, задали название и времена графика, но пока ничего не происходит. Чтобы увидеть нужный вам график, вам нужно, щелкнув правой клавишей мышки по графическому окну, выбрать в выпадающем меню пункт «Добавить пути».

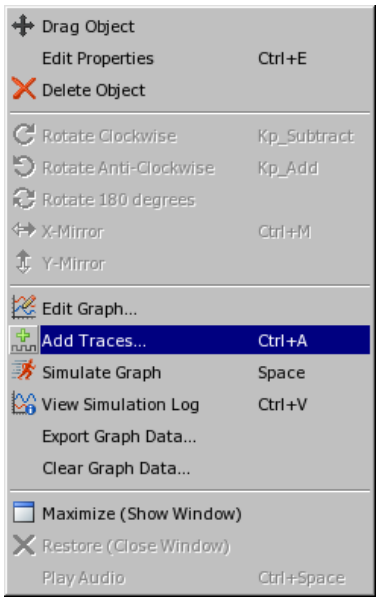


Рис. 1.9. Выпадающее меню работы с графиками

Этот пункт меню приводит вас в диалоговое окно, где вы выбираете, какие из заданных пробниками сигналов вас в настоящий момент интересуют. В верхнем поле текстового ввода **Name:** вы можете ввести удобное название графика. В моей версии программы работает только латиница, но, возможно, есть русифицированные версии.

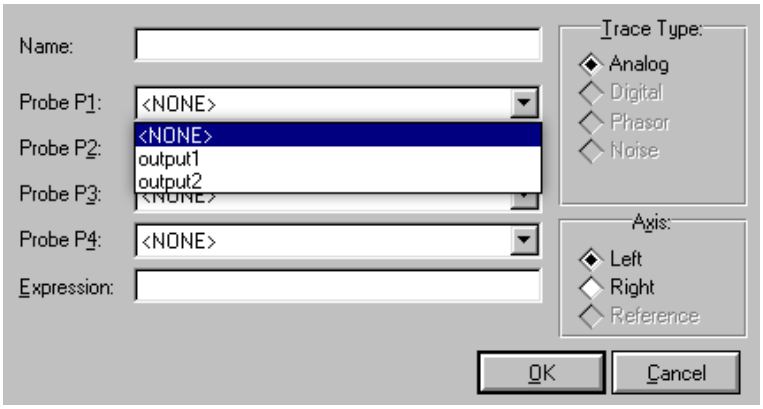


Рис. 1.10. Диалоговое окно выбора отображаемых кривых

В открывающемся окне выбора «проб» (клавиша со стрелкой вниз) вы можете выбрать нужный график. Все задав, все проверив, вы выбираете (рис. 1.9.) **Simulate Graph** в выпадающем меню, что завершает ваши труды в части получения результатов симуляции.

Что мне больше всего понравилось сразу и безоговорочно, так это возможность нарисовать две электрические цепи рядом и посмотреть их работу без открывания нового проекта. Вы можете использовать каждое из графических окон для получения вида кривой, которая вам нужна в данный момент. Если вы, положим, поменяете время симуляции, то программа предложит повторить симуляцию. Достаточно согласится с этим, и вы получите новый график. Я грозился, что вы будете удивлены, если зададите время по умолчанию, сделаем это, чтобы получить график:

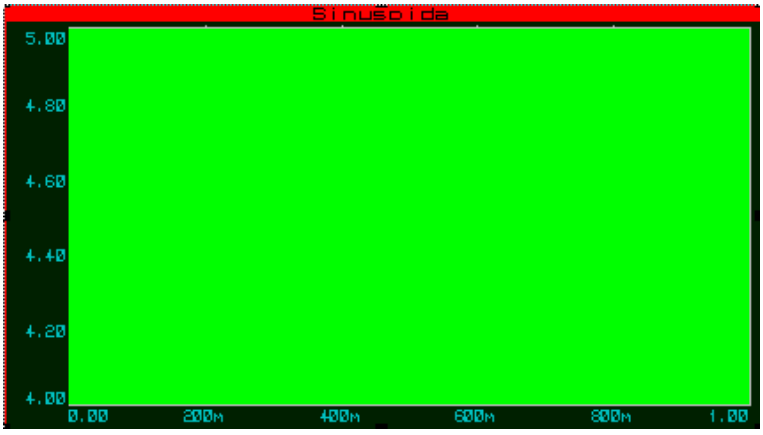


Рис. 1.11. Новый вид графика после повторной симуляции

После запуска симуляции обратите внимание, что на панели состояния отображается процесс (или прогресс) симуляции.

Не дело, думаю, вы согласны со мной, задавшись вопросом об интегрирующей цепи, так долго обсуждать программу, но и программа, и опять, я надеюсь, вы согласны со мной, очень даже интересная!

Итак, два эксперимента: с источником постоянного напряжения и испытательным синусоидальным сигналом, — пока не принесли положительного результата. Но, это я точно знаю, интеграл от синуса дает (минус) косинус. Проведем проверку этого предположения, добавив еще одну пробную точку на выходе генератора синусоидального напряжения. Его частоту я увеличу до 100 кГц, позже поясню зачем, создам два графических окна для отображения входного и выходного сигнала, тоже позже поясню почему, и проверю, так ли это?

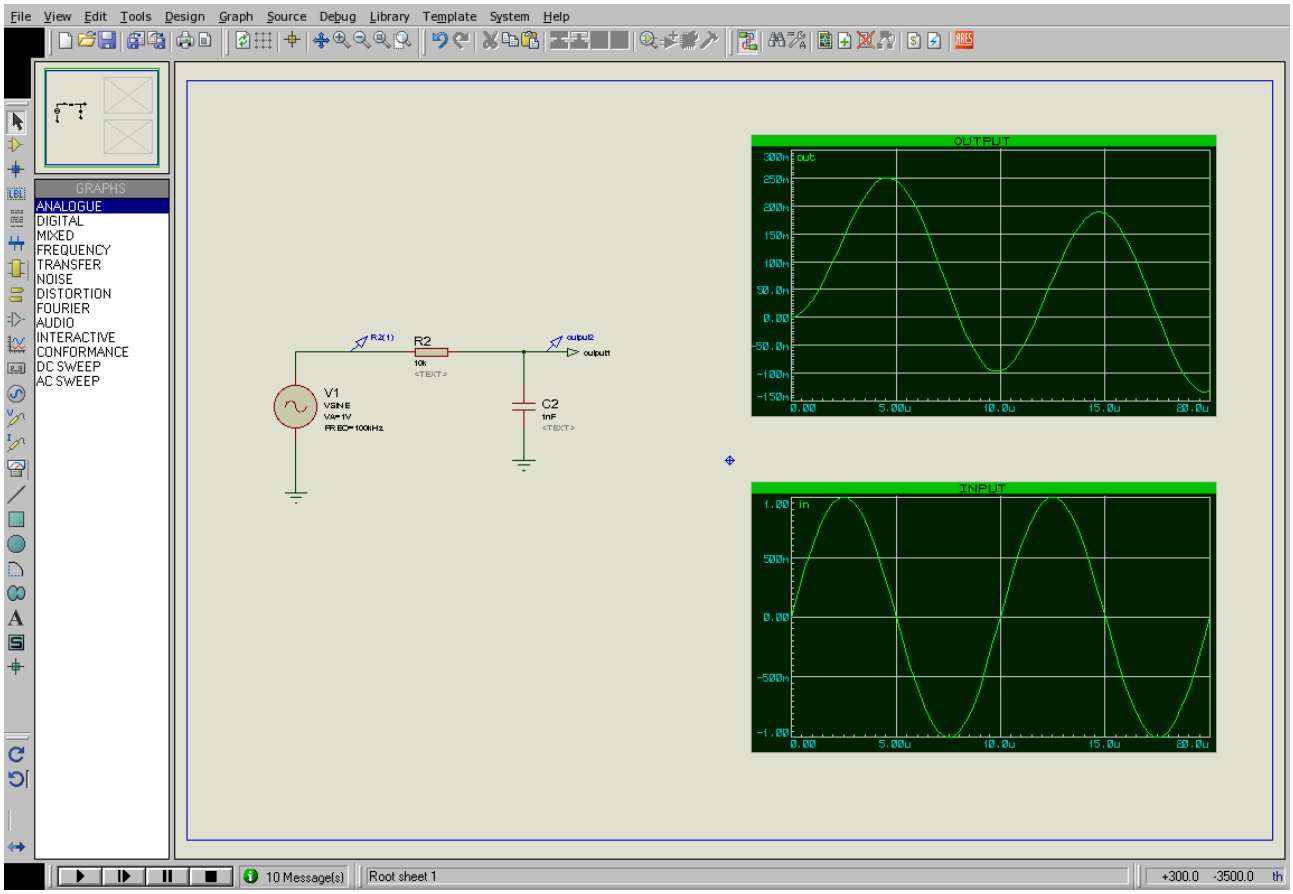


Рис. 1.12. Входной и выходной сигналы интегрирующей цепи

Сигналы синуса и косинуса отличаются фазой колебаний, и это ясно видно на графиках. Можно было бы совместить графики в одном окне, но сигнал на выходе меньше входного — резистор и конденсатор помимо того, что образуют интегрирующую цепь, образуют делитель напряжения, который и уменьшает напряжение на выходе при частоте 100 кГц. Теперь, почему я выбрал частоту 100 кГц. Выписывая выше фразу об интегрирующей цепи из курса теоретической электротехники, я прибавил от себя о дополнительных условиях. Вот как это выглядит в оригинале:

> *Чтобы интегрирование происходило с относительно малыми искажениями, между частотой сигнала ω (2πf) и параметрами интегрирующей цепи должно выполняться соотношение ωRC >> 1.*

Другими словами частота f >> 1/2πRC. Сопротивление в схеме 10 кОм, конденсатор 1 нФ. Я выбрал частоту, которая (значительно?) больше расчетной.

Для импульсного напряжения похожее условие RC >> tимп.

Я откладывал это испытание импульсным сигналом в первую очередь оттого, что

предварительно хотел заметить, когда мы проводили испытание интегрирующей цепи с помощью батарейки, мы не видели начальный момент, когда батарейка только подключается к RC цепи. Как вы знаете такой момент называется переходным процессом, а наблюдая его мы увидели бы процесса заряда конденсатора через резистор, проходящий по экспоненте. Ничто не мешает вам провести такой эксперимент, используя генератор «ступеньки», но похожий результат получится и при использовании однополярных прямоугольных импульсов. Поэтому можно сразу провести последнюю проверку.

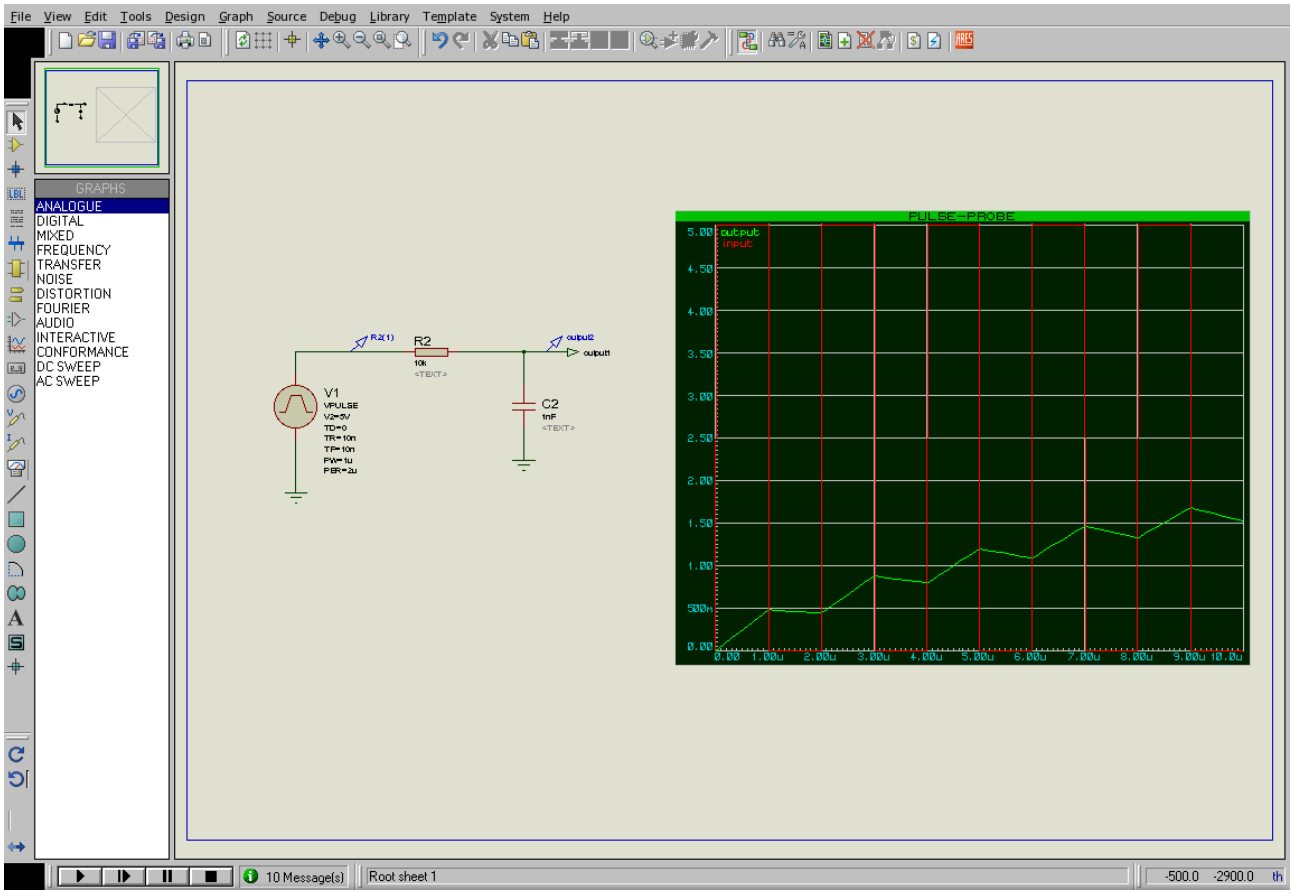


Рис. 1.13. Реакция интегрирующей цепочки на прямоугольные импульсы

То, что видно на графике, больше напоминает мне знакомую картину проявления интегрирующих свойств RC цепи, скажем, на выходе выпрямителя, или при использовании в качестве накопителя в устройстве измерения частоты сигнала. Но не помню, чтобы я в этом случае задумывался об интегрирующей цепи. А напрасно.

Еще один график, который часто приходилось использовать, при этом даже временами разглагольствуя об RC эквивалентной цепи, это амплитудно-частотная характеристика устройства. И в этом случае интегрирующие свойства определяют многие параметры цепи, а вот тот факт, что речь идет об интегрирующей цепи, как-то забывался.

Попробуем нарисовать амплитудно-частотную характеристику интегрирующей цепи. Для графика можно выбрать подходящий вид диаграммы, но прежде заменим примитив симуляции VSIN генератором. Для установки генератора предназначена клавиша **Generator Mode**, расположенная чуть ниже кнопки выбора графика. Это понадобится для вывода характеристики в относительных единицах, децибелах. Типы генераторов, представленные в библиотеке, перекрывают все обычные нужды для моделирования электрических схем. На рисунке ниже в поле компонентов отображен весь список.

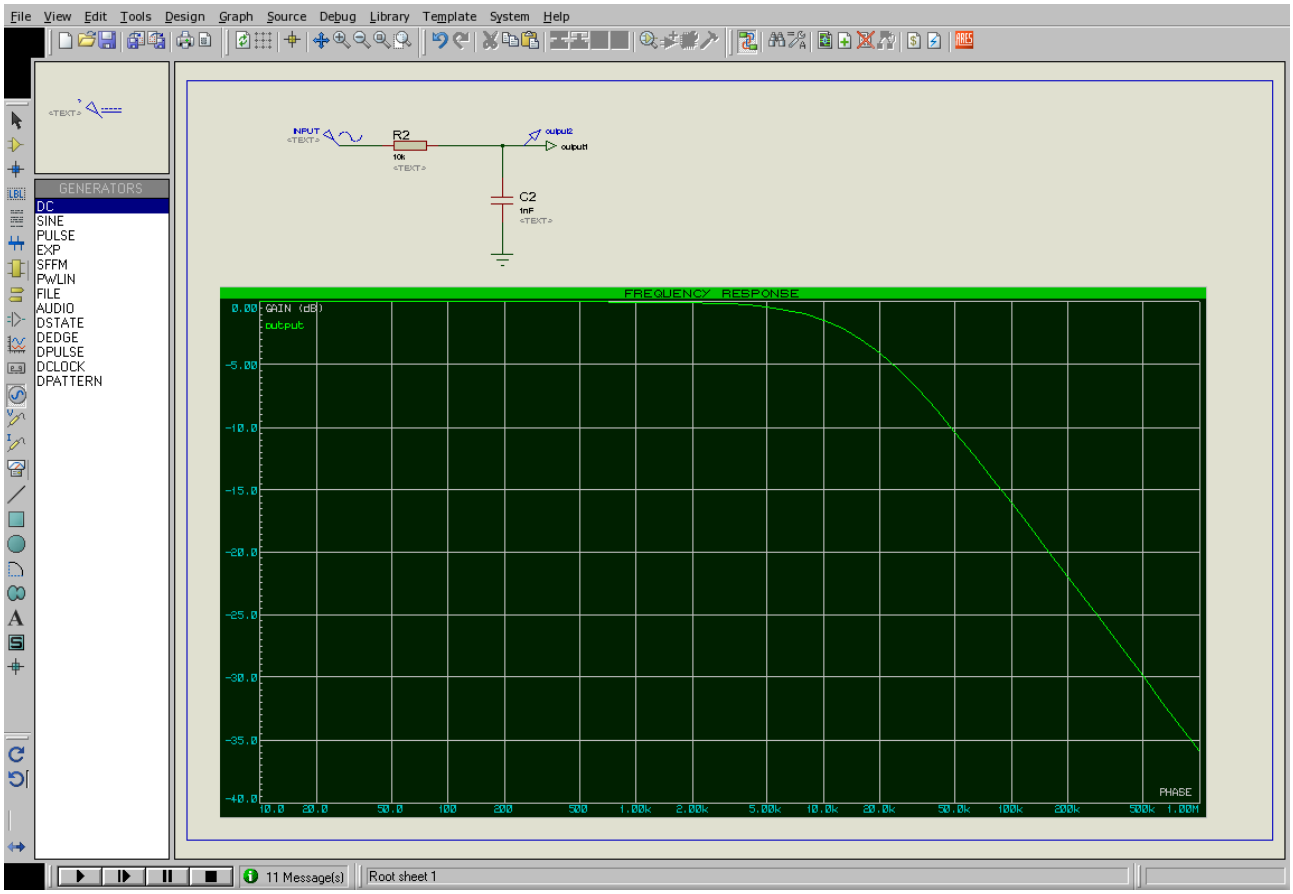


Рис. 1.14. Выбор генератора для построения АЧХ

Выбор характера графика можно сделать после нажатия на кнопку **Graph Mode** из появляющихся рядом доступных вариантов: ANALOGUE, DIGITAL, MIXED и т.д.

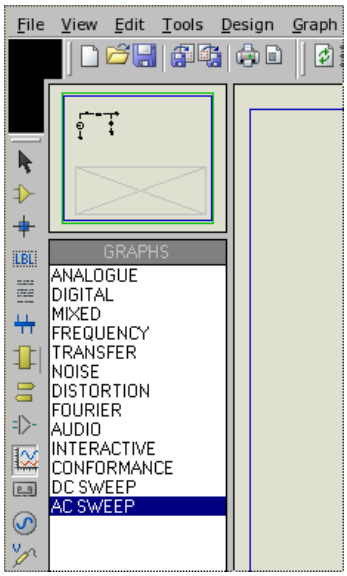


Рис. 1.15. Выбор характера графика симуляции

Как видно на графике рис. 1.14, спад частотной характеристики происходит, как и положено, со скоростью 20 дб/дек, а частота среза, на уровне -3 дБ, примерно 15-16 кГц. Это значение можно получить из соотношения между параметрами RC цепи и частотой. И в этом

плане меня интересует еще два момента, касающихся АЧХ — как поведет себя программа, если добавить еще одну RC цепь с другой частотой среза, и может ли программа показать фазочастотную характеристику RC цепи. Дело в том, что не всегда эти характеристики удается получить в программе моделирования. Две последовательно включенные RC цепи должны дать спад АЧХ после второй частоты среза со скоростью 40 дБ/дек.

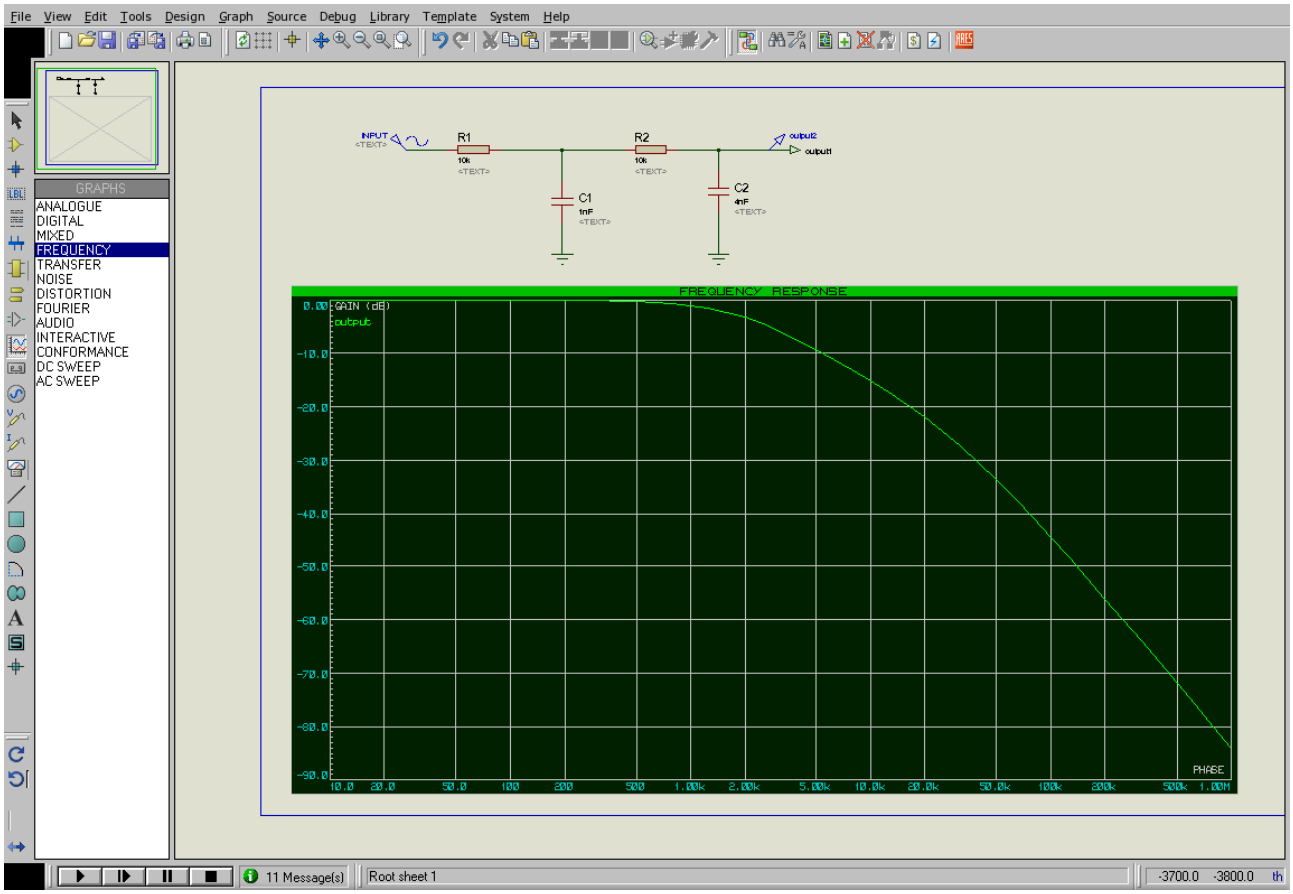


Рис. 1.16. Амплитудно-частотная характеристика двух RC цепей

Для получения фазочастотной характеристики руководство пользователя советует:

> *In a frequency graph, the two y-axes (left and right) have special meanings. The left y-axis is used to display the magnitude of the probed signal, and the right y-axis the phase. In order to see both, we must add the probes to both sides of the graph. Tag and drag the OUT probe onto the left of the graph, then drag it onto the right. Each trace has a separate colour as normal, but they both have the same name. Now tag and drag the U1(POS IP) probe onto the left side of the graph only.*

Используя эту подсказку, хотя в руководстве оговорено, что не обязательно удалять другие графики, я удаляю старый график и начинаю новую жизнь с установки нового частотного графика. На графике, используя правую клавишу мышки, из выпадающего меню выбираю **Add Traces...**, где в диалоговом окне, как и прежде задаю имя графика и для левой оси выбираю метку *output*. Затем, повторно запустив **Add Traces...**, я добавляю для правой оси кроме *Probe 1:*, как и в предыдущем случае, еще один параметр *Probe 2:*, где как и в свойствах диаграммы выбираю *INPUT*. Осталось задать свойства диаграммы с указанием *INPUT* в поле *Reference:* и запустить симуляцию графика, выбрав из выпадающего меню **Simulate Graph...**

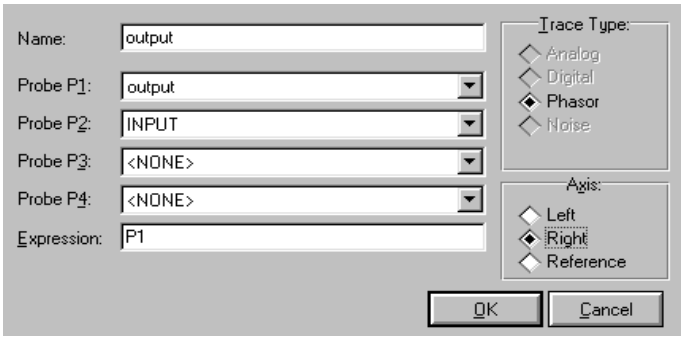


Рис. 1.17. Диалоговое окно добавления графика

Знание амплитудной и фазочастотной характеристик важно при выборе глубины общей отрицательной обратной связи в многокаскадных усилителях. И приятно, что программа позволяет это сделать.

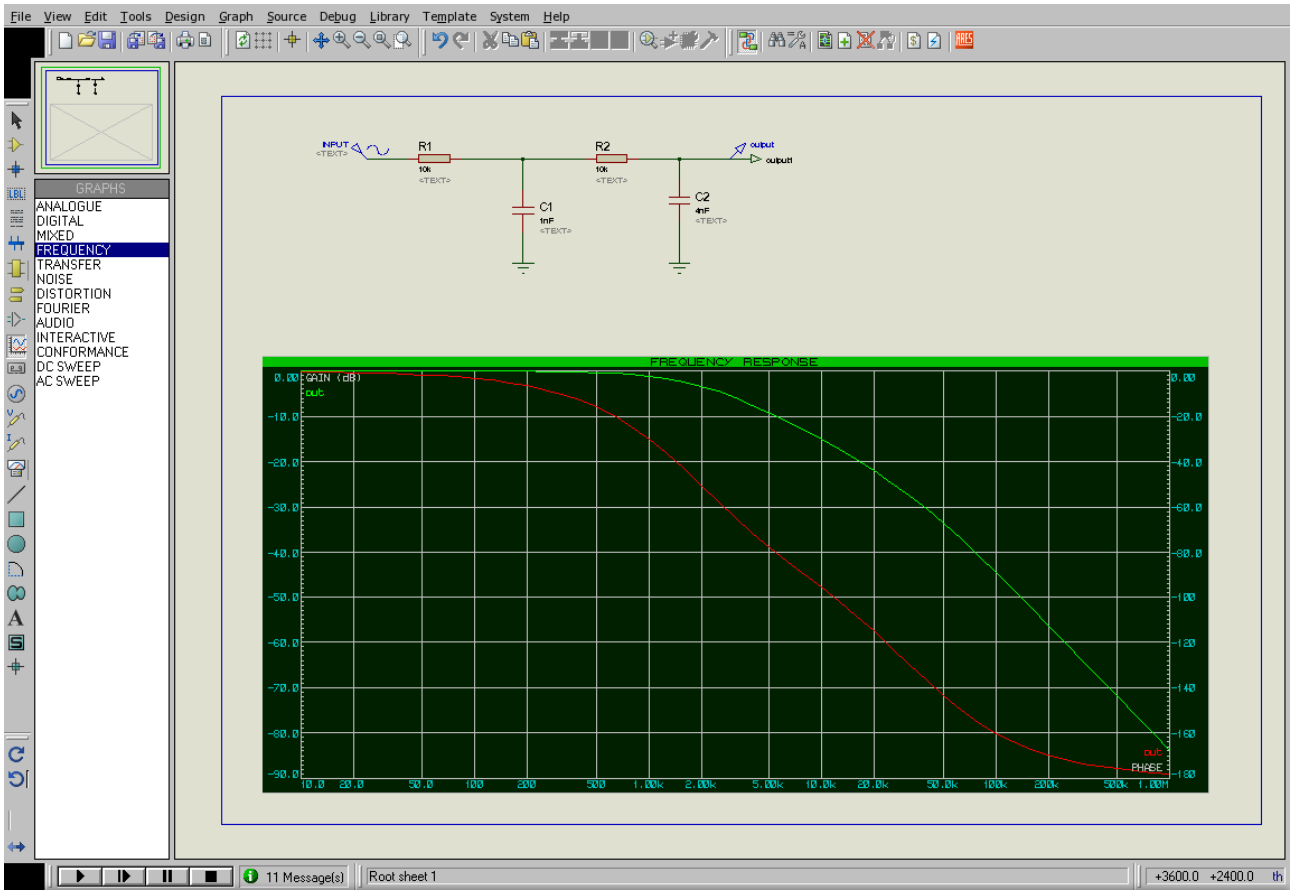


Рис. 1.18. Амплитудная и фазочастотная характеристики двух RC цепей

Все эксперименты, проделанные для интегрирующей RC цепи, можно проделать и для LR цепи, и все, что сказано для интегрирующей цепи, можно отнести к дифференцирующей цепи. Обычно производная некоторой функции характеризует скорость изменения функции. При испытаниях дифференцирующей RC цепи с источником постоянного тока, подобно верхней схеме на рис. 1.4, на выходе напряжение отсутствует. Действительно, скорость изменения постоянной величины нулевая, что и отражается и в этом испытании, и в том, что производная от постоянной равна нулю. Использование источника синусоидального напряжения, как и в случае интегрирующей цепи, если не обращать внимание на фазу

выходного сигнала, не очень показательно. Производная от синуса — косинус. А как выглядит реакция дифференцирующей цепи на прямоугольные импульсы, проще посмотреть, чем прочитать рассказ об этом.

Не сомневаюсь, что не я один забыл со студенческих времен операторную форму уравнений, описывающих электрические цепи, и преобразования Лапласа, и отвечать на такой простой вопрос: «Что такое интегрирующая цепь?» — я бы не счел нужным, уж очень все очевидно. Интересно, много ли еще подобных элементарных вопросов можно услышать от начинающих любителей?

Почему не выпрямляет диод?

Этот вопрос поставил меня поначалу в тупик. Схема, которую я получил в готовом виде вместе с вопросом, на первый взгляд выглядела привычно, а симуляция процесса явно демонстрировала отсутствие выпрямления. И схема и симуляция были сделаны в другой программе, но как это выглядело, я проиллюстрирую в программе Proteus. Верхняя схема относится к тому, как я представлял себе схему и результат симуляции, нижняя к тому, что получалось в действительности.

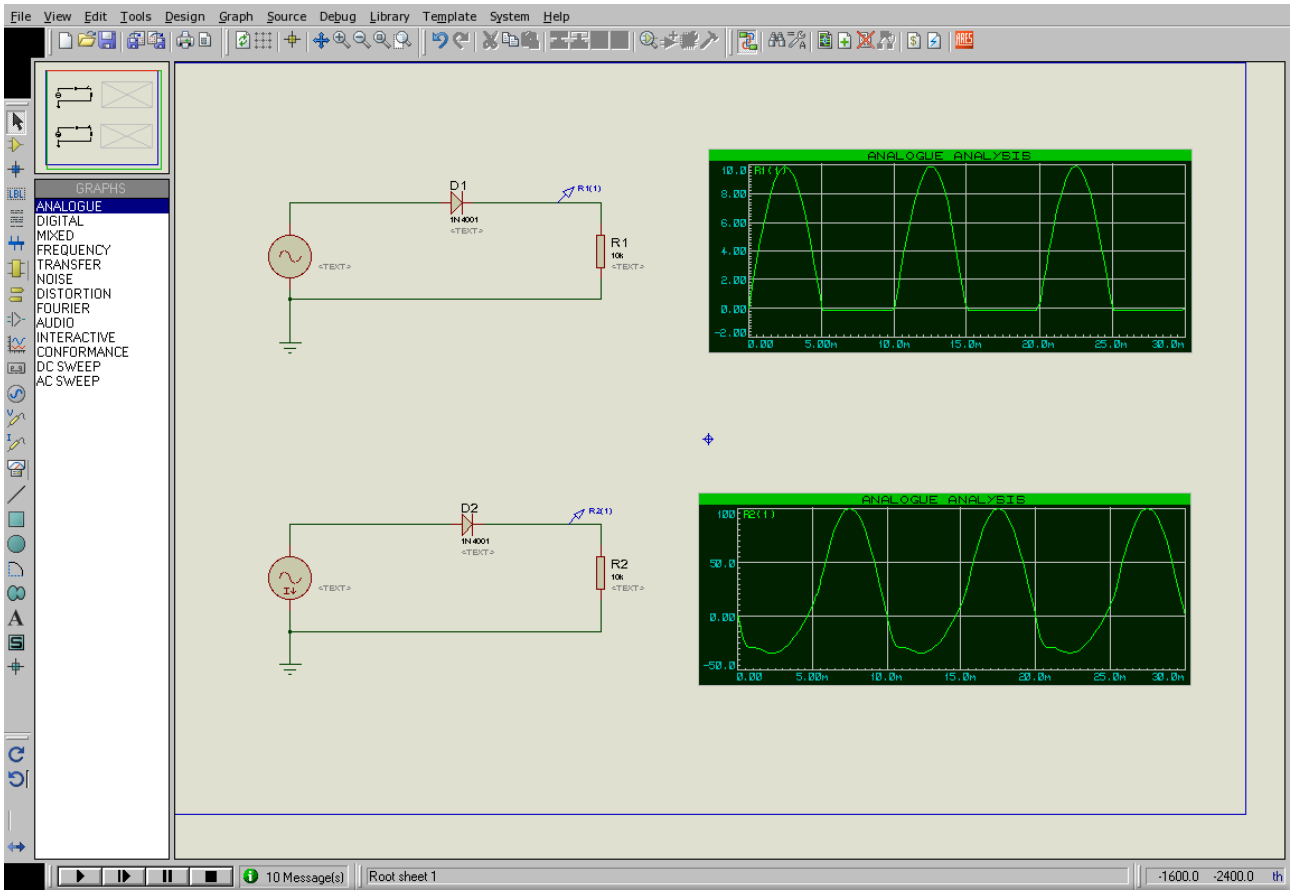


Рис. 2.1. Симуляция выпрямления переменного напряжения 100 Гц

Что поучительного в этих двух схемах? Первая, как можно понять из рисунка, работает от генератора напряжения, тогда как вторая от генератора тока. Разница между этими двумя источниками в том, что первый призван поддерживать напряжение, которое не зависит от сопротивления нагрузки, то есть, иметь очень малое внутреннее сопротивление, а второй должен поддерживать ток, независящий от нагрузки, то есть иметь очень большое внутреннее сопротивление. При этом его напряжение при обратном включении диода (для обратной полу-волны) будет расти так, чтобы обеспечить необходимый ток в нагрузке.

Мне не приходилось собирать стабилизаторы тока для каких-либо своих нужд, и редко приходилось пользоваться стабилизатором в режиме стабилизатора тока, но схемы стабилизатора тока есть, а мне было бы интересно посмотреть, как работает такая схема в Proteus.

Поиск схемы стабилизатора несколько затянулся, мне не хотелось использовать микросхему стабилизатора напряжения, но через несколько минут я нахожу подходящую схему, рисую ее в Proteus, добавляю два прибора для измерения напряжения и тока на

выходе, и начинаю эксперименты.

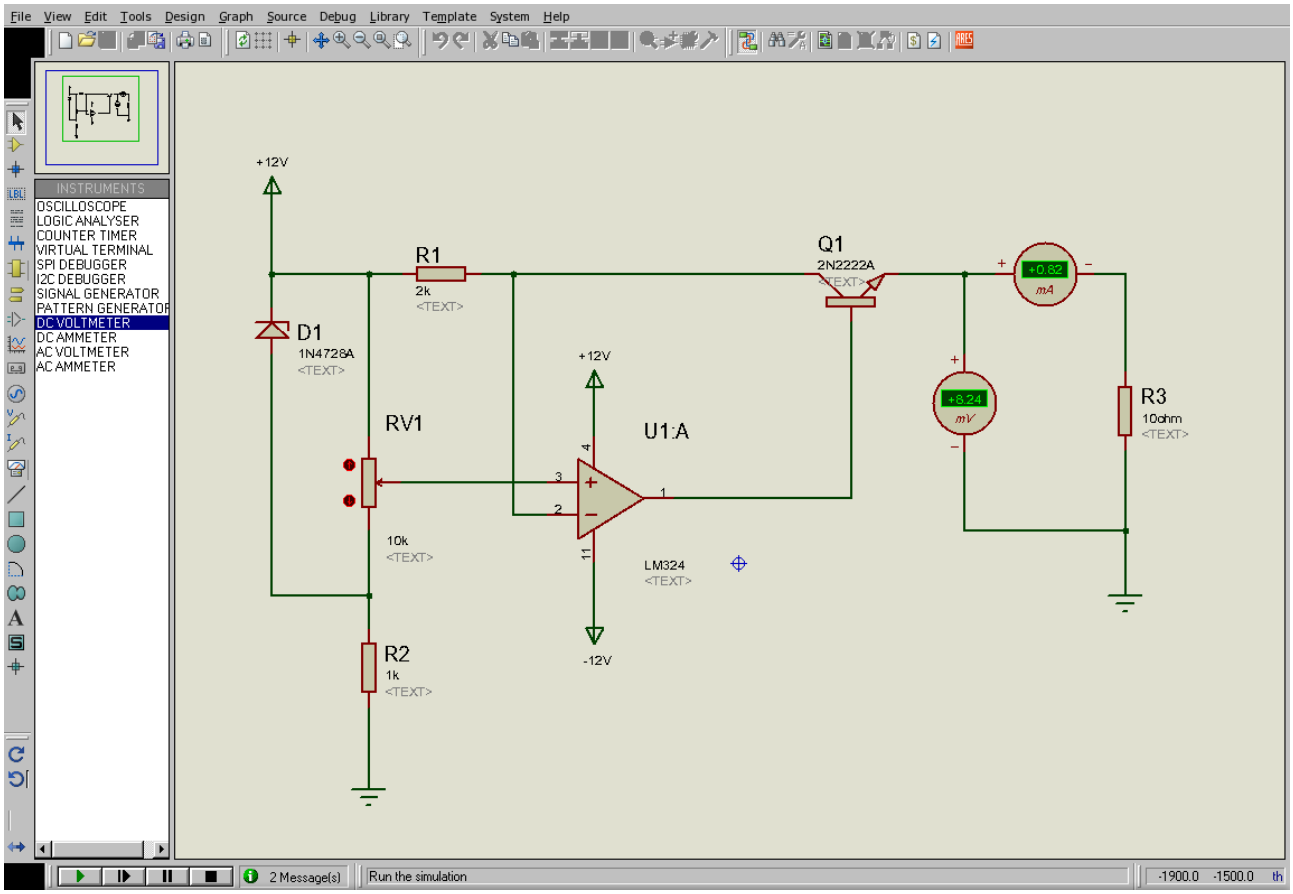


Рис. 2.2. Эксперименты со схемой стабилизатора тока

Опорное напряжение, формируемое стабилитроном D1 и снимаемое с делителя напряжения на потенциометре RV1 на прямой вход операционного усилителя, сравнивается с падением напряжения на резисторе R1 от тока, практически, равного току в нагрузке R3. Если сопротивление нагрузки изменяется, операционный усилитель изменяет напряжение на выходе так, чтобы вернуть ток в нагрузке к заданному значению.

Сейчас сопротивление нагрузки 10 Ом. Используя возможности программы Proteus, тот факт, что потенциометр интерактивный, я могу задать ток, который мне представляется удобным для последующих опытов. При наведении курсора на точки управления рядом с потенциометром он превращается в плюс в одной из них и в минус в другой. Щелчком по этим точкам можно регулировать положение выходного вывода потенциометра.

Отображение значений на дисплее вольтметра и амперметра при желании можно сделать крупнее. Это приводит к общему укрупнению схемы, но позволяет лучше разобрать детали. Достаточно щелкнуть левой клавишей мышки в нужном месте чертежа, а затем колесиком увеличить или уменьшить масштаб отображения.

Если при этом не получается вернуть общий вид схемы к первоначальному, можно использовать окно панорамирования, слева вверху. Щелчком левой клавиши мышки по этому окну можно вызвать рамку привязки, перемещая эту рамку по общему виду выбрать нужный ракурс, что отображается в окне редактирования схемы, а повторным щелчком зафиксировать это положение.

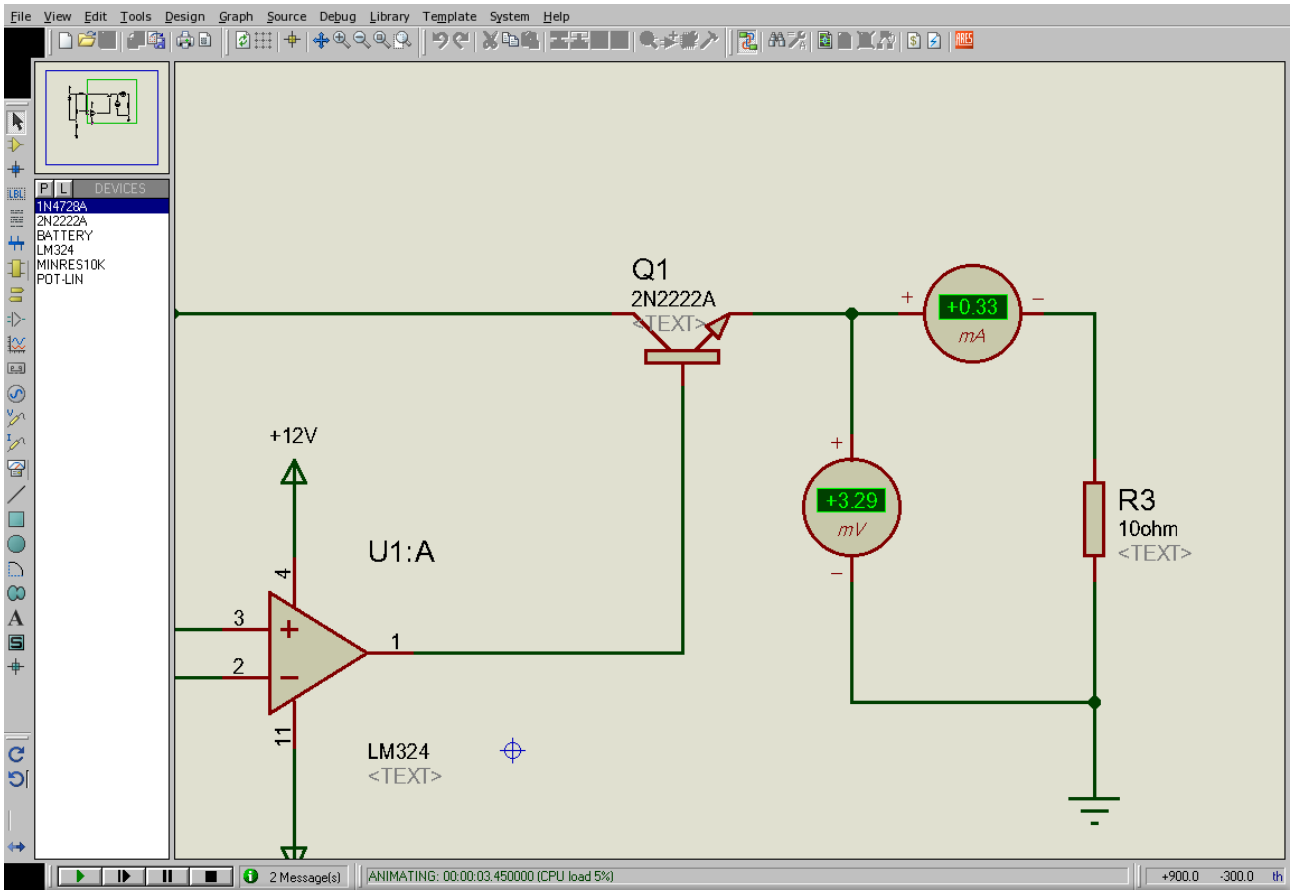


Рис. 2.3. Выбор тока в нагрузке

Теперь можно увеличить значение резистора нагрузки до 100 Ом или 1 кОм. Я догадываюсь, что никакая программа не отменяет законов электротехники, и возможности регулирующих элементов при питании от 12 вольт отнюдь не безграничны, но попробую сразу увеличить сопротивление нагрузки до 10 кОм. Первое, что приходится при этом сделать, милливольтметр, который показывал напряжение 3.29 мВ, оказывается перегружен, это изменить в свойствах вольтметра его пределы, выбрав *Volts*. Повторный запуск показывает напряжение на выходе 3.29 В, а ток, как и прежде 0.33 мА.

Как и положено стабилизатору тока, он поддерживает ток в нагрузке заданной величины, хотя сопротивление нагрузки изменилось в 1000 раз. Совсем неплохо.

Начинающий вполне может, используя возможности программы, исследовать эту схему, чтобы понять, как она работает, как влияет каждый из элементов схемы на результаты ее работы. Наиболее очевидное применение схемы — измерение величины сопротивления. Если вместо сопротивления нагрузки R3 включать неизвестное сопротивление, а выходной ток задать удобным образом, например, 1 мА, то измеряя падение напряжения с помощью вольтметра, как это показано на схеме выше, можно по его показаниям прочитать величину измеряемого сопротивления. Для удобства подобного рода измерений можно сделать несколько пределов измерений с помощью переключателя, обеспечивающего разные выходные токи. Программа позволяет провести все предварительные эксперименты, получить, практически, готовое решение, которое останется проверить на макетной плате, при необходимости провести коррекцию элементов схемы, при применении отечественных аналогов может возникнуть необходимость в подобной коррекции, а затем либо собрать схему в окончательном виде, либо развести печатную плату для схемы во второй части

системы с названием Ares.

Такой, вот, пример применения Proteus в любительской практике. Но кроме практического применения программа позволяет ответить на многие вопросы, больше подходящие цели обучения, что несомненно поможет любителю в освоении электроники, особенно если будет сочетаться с проверкой наиболее значимых экспериментов на реальной макетной плате с использованием реальных приборов. Конечно, можно и не использовать программу, но проведение очень большого количества реальных испытаний может привести в уныние даже очень увлеченного и упорного любителя. Всегда полезно разнообразить подходы, поскольку смена обстановки благотворно сказывается на поддержании интереса к предмету.

Особую пользу любитель может извлечь, когда он пытается решить свои задачи при использовании элементов электрической схемы в предельных или близких к ним режимах. Реальные пробы в этом случае требуют либо очень тщательного продумывания, либо способны привести к значительным тратам на покупку деталей, раз за разом выходящих из строя. Обычно схожая ситуация возникает в такие моменты, когда, собрав готовую схему, любитель обнаруживает, что, например, транзистор сильно греется. Это может обнаружиться случайно и не сразу после первых испытаний. Первые быстрые испытания схемы могут обнадеживать: схема хорошо работает, не потребовала настройки. Любитель принимает решение сделать печатную плату, вкладывая много труда и упорства в эту работу. Но, перенеся схему на печатную плату, он включает ее надолго и обнаруживает тревожащие его обстоятельства. Тот факт, что транзистор греется, может не иметь особого значения, если это не сказывается роковым образом на выходных параметрах схемы, если это не приводит к выходу транзистора из строя. Оценить возможные последствия удобнее в программе, где можно не только измерить рассеиваемую транзистором мощность, но и оценить влияние температуры на окружающие его элементы схемы без риска окончательно «доконать» транзистор. Хотя сам транзистор в выбранном режиме может «безболезненно» греться, увеличение, скажем, сопротивления рядом расположенного резистора может приводить в определенных условиях к увеличению мощности рассеивания на транзисторе, а этот лавинообразно происходящий процесс не только расстроить работу схемы, но и вывести ее из строя.

Иногда подобные процесс происходят настолько быстро, что наблюдение их в реальных условиях сопряжено с большими трудностями. Особенно это касается переходных процессов, начинающихся при включении схемы. Сейчас достаточно много устройств с бестрансформаторным питанием. Даже исключая сложности налаживания подобных устройств, связанные с опасностью поражения электрическим током, трудно наблюдать переходные процессы при подключении к сети. Даже в профессиональной разработке учет самых неблагоприятных обстоятельств в этом случае оценивается вероятностью проявления этих самых неблагоприятных обстоятельств, положим, когда при включении схемы напряжение в сети равно амплитудному значению, к которому добавляется импульс помехи. Программа позволяет растянуть время, добавить источник помех и тщательней рассмотреть процесс.

Но даже вне таких случаев, с которыми любитель, видимо, никогда не столкнется, программа позволяет задавать множество вопросов из разряда «А если...», вопросов, которые обычно не приходят в голову или требуют трудоемкого ответа, тогда как многие вопросы, подобные тому, что послужил названием этой главы, позволяют лучше понять многие аспекты работы реальных устройств, а порой становятся базой для новых решений, для создания новых полезных устройств.

Как работает транзистор?

Первое, что приходит в голову, когда слышишь подобный вопрос, это рассказать об устройстве транзистора: p-n переходах, их объединении в трехслойную конструкцию и т.д. Физика полупроводников, если подходить к вопросу серьезно, достаточно сложна и требует хотя бы начальных знаний о квантовой физике. И это касается только вопроса методичности изложения, тогда как и сама квантовая физика, как, впрочем, и классическая теория электричества, порою не в состоянии ответить на все возникающие вопросы. В итоге, чаще приходится просить принять что-то на веру после обширных математических выкладок и многочисленных поясняющих рисунков, а это никак не способствует пониманию существа вопроса.

Но действительно ли спрашивающего интересует физика полупроводников? Кого-то, может быть, и интересует, но большая часть вопрошающих, как мне кажется, больше склонна получить ответ на другой вопрос: как осмысленно использовать транзистор в схемах?

Транзистор — один из наиболее употребительных активных элементов электронных схем. В последнее время схемы часто строятся с использованием микросхем, а подход к их созданию требует только знания свойств и функциональных возможностей микросхемы, но следует забывать, что и свойства и функциональные возможности микросхемы обусловлены свойствами скрытых в ней компонент, где транзисторы продолжают играть значительную роль. Так что вопрос о работе транзистора не утратил актуальности. Но с учетом «микросхемного» подхода к созданию устройств рассмотрение свойств и функциональных возможностей транзисторов мне кажется более актуальным, чем физических принципов, лежащих в основе их работы, особенно для любителей.

Чаще всего транзистор используется для усиления сигнала. И хотя сигналы бывают разные, наиболее простые эксперименты можно осуществить с усилением синусоидального сигнала. А Proteus предоставляет все необходимое для этого.

В одном из весьма аргументированных сообщений, встреченных мною на форуме, где обсуждалась работа с Proteus, говорилось, что эта среда разработки предназначена для работы с цифровой техникой и микроконтроллерами, поэтому аналоговые схемы в ней исследовать нет резона. Меня заинтересовало, можно ли рассказать о применении транзисторов с помощью программы Proteus? Попробую это сделать.

Итак. Усиление сигнала можно рассматривать как усиление сигнала по току, усиление по напряжению и усиление по мощности. Усиление сигнала по току у транзистора обусловлено его свойством — ток коллектора и ток базы связаны соотношением $Iк = K*Iб$. При этом, если ток базы изменяется по какому-то закону, то ток коллектора изменяется по тому же закону, то есть, соотношение выше можно рассматривать для каждого момента времени. Вот, собственно, что я посчитал бы необходимым ответить на вопрос о том, как работает транзистор.

При работе с симметричными сигналами транзистор, как правило, включают так, чтобы напряжение на коллекторе было равно половине напряжения питания. В простейшем случае это достигается подбором резистора в цепи базы.

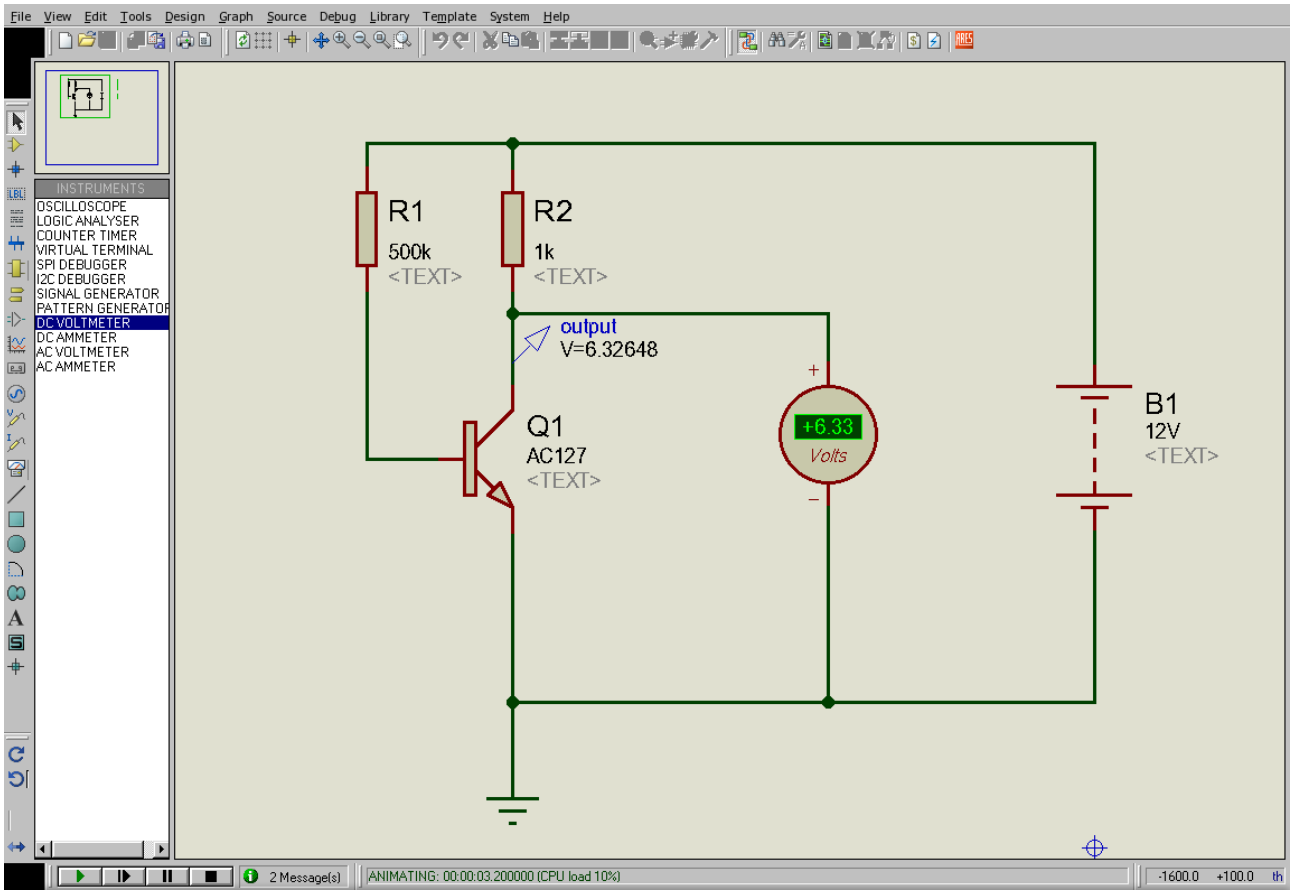


Рис. 3.1. Задание рабочего режима транзистора

Если в такой схеме менять величину сопротивления R1, что в Proteus достигается щелчком правой клавиши мышки по этому компоненту с последующим выбором из выпадающего меню пункта **Edit Properties**, открывающего, в свою очередь, диалоговое окно свойств резистора, где и задается величина сопротивления, так вот, если менять R1 то можно получить разное напряжение на коллекторе транзистора.

Однако гораздо полезнее подключить к схеме предыдущего рисунка генератор синусоидального напряжения, используя клавишу **Generator Mode** (иконка на левой инструментальной панели в виде кружка с синусоидой). Если теперь с помощью клавиши **Graph Mode** нарисовать график, можно выбрать **ANALOGUE** из представленных возможностей, добавить пробник напряжения, обозначив его метку как *output*, то после настройки графика, в его свойствах я задаю время 10 мС (10m), так как я задал для генератора синусоиды 10 мВ (10m RMS) и частоту 1 кГц (1k), добавить кривую для графика, используя пункт выпадающего меню **Add Traces...**, то теперь можно наблюдать выходной сигнал после запуска симуляции в пункте выпадающего меню **Simulate Graph** при разных значениях сопротивления, чтобы оценить, как влияет выбор рабочей точки на получающийся результат.

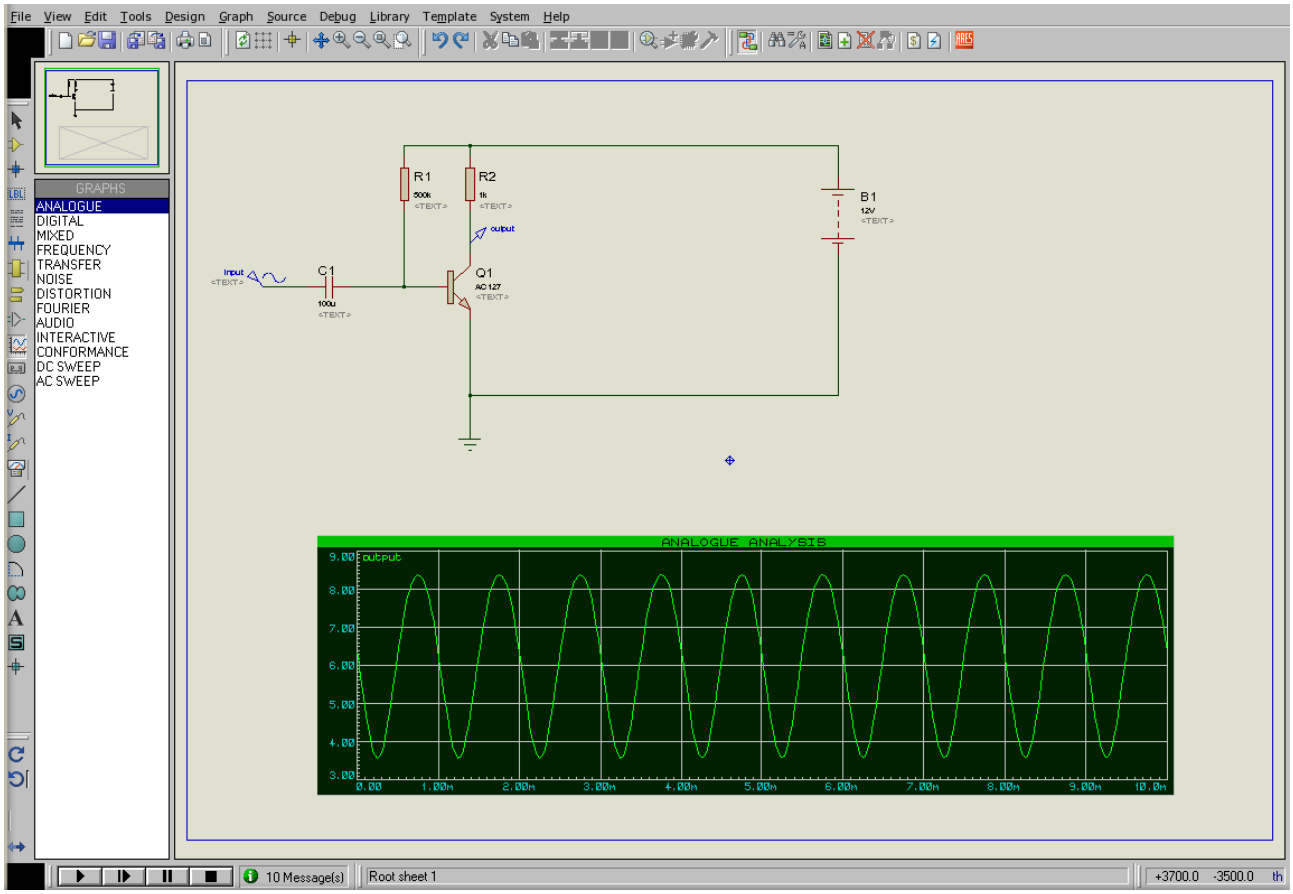


Рис. 3.2. Наблюдение синусоидального сигнала на коллекторе транзистора

Зачем на входе транзистора конденсатор? Чтобы сопротивление генератора, а генератор имеет некоторое внутреннее сопротивление, не меняло заданный режим. Конденсатор не пропускает постоянный ток, значит не изменит наших настроек. Можно включать разные источники сигнала, можно менять сопротивление в цепи коллектора, можно наблюдать многое в программе Proteus, и можно проверить, действительно ли между током базы и током коллектора есть соотношение, о котором было сказано в самом начале, и можно проверить, действительно ли ток (ток, а не напряжение, как у меня) коллектора повторяет закон изменения тока базы. Кстати, можно проверить и фазовые соотношения между напряжениями на базе транзистора и напряжением на его коллекторе. Это удобно сделать добавив второй график для сигнала *input* на рис.3.2.

Я же хочу проделать другие испытания. Если верить рассказам о Proteus, которые я нашел в Интернете, то работа усилителя не зависит от того, какой транзистор вы используете. Выбирая разные транзисторы из библиотеки компонентов, я хочу посмотреть на амплитудно-частотные характеристики получающихся усилителей. Для этой цели я использую ту же схему, добавлю в свой набор некоторое количество транзисторов, затем, меняя транзисторы, посмотрю, действительно ли их АЧХ одинаковы?

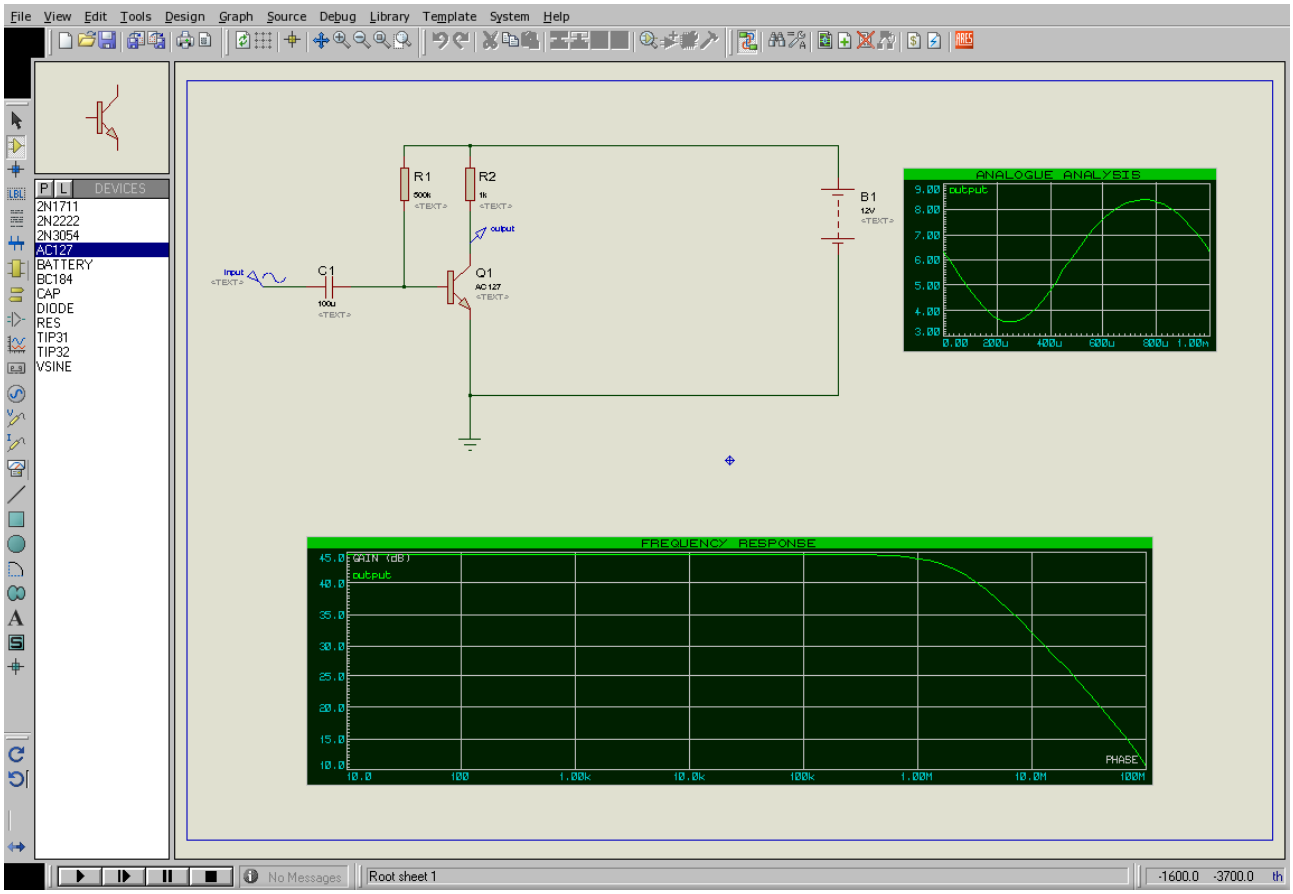


Рис. 3.3. Испытания разных транзисторов в Proteus

Для транзистора AC127, как это видно из графика, частота среза примерно 5 МГц. Похоже ли это на правду? Не хочу заниматься расчетами, но если современные транзисторы малой мощности имеют граничную частоту при включении с общей базой порядка 300 МГц, а усиление около 100, то граничная частота должна получиться около 3 МГц.

Когда рассказывают о строении биполярного транзистора, то обязательно упоминают о том, что он имеет две пограничные области на стыке полупроводников разных типов проводимости, очень напоминающие по свойствам заряженные конденсаторы. Этому свойству транзистор обязан своим поведением при усилении сигналов разных частот. Его поведение можно моделировать используя RC цепь. Амплитудно-частотная характеристика интегрирующей RC цепи и однокаскадного усилителя на транзисторе будут обладать одинаковыми свойствами. Можно сравнить графики рис. 1.14 и предыдущего, чтобы увидеть наличие верхней граничной частоты в обоих случаях и спада амплитудно-частотной характеристики со скоростью 20 дБ на декаду. Величина эквивалентного конденсатора зависит от конкретной модели транзистора. Если заменить одну модель транзистора другой, то можно ожидать, что амплитудно-частотная характеристика каскада изменится, если, конечно, у них различается такой параметр, как граничная частота усиления.

Поэтому я хочу заменить транзистор на TIP31.

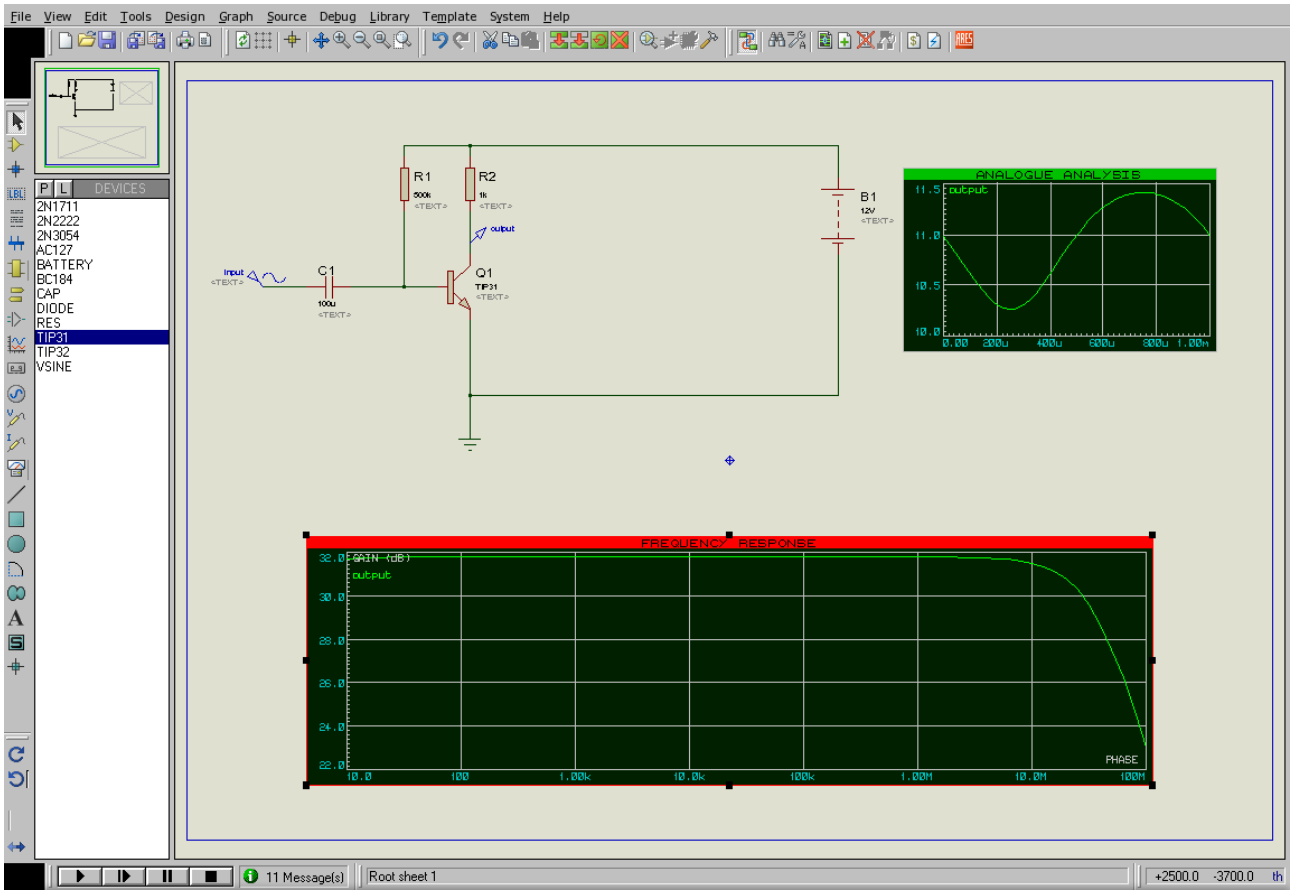


Рис. 3.4. Амплитудно-частотная характеристика после замены транзистора

Не знаю, как у вас, а у меня верхняя граничная частота «улетела» за 10 МГц. Не уверен я теперь, что Proteus не годится для аналогового симулирования схем. Чтобы развеять свои сомнения я верну транзистор AC127, а в цепь эмиттера включу резистор. Этот резистор, удобнее рассмотреть его работу в схеме рис.3.1, приведет к тому, что напряжение база-эмиттер транзистора изменится. На нем будет падать напряжение, которое нужно вычесть из напряжения между базой и общим проводом, чтобы получить напряжение база-эмиттер. Входным напряжением для транзистора служит именно напряжение база-эмиттер. Таким образом, резистор в цепи эмиттера уменьшает входной сигнал для транзистора. Он, резистор, является резистором обратной связи — мы часть выходного сигнала (а на резисторе в цепи эмиттера в значительной мере сказывается именно выходной сигнал) сложили с учетом фазы со входным сигналом, дополнение «с учетом фазы» в данном случае указывает на то, что обратная связь будет отрицательной. А, насколько я знаю, отрицательная обратная связь должна расширить диапазон рабочих частот каскада усиления, то есть, верхняя граничная частота должна увеличится. Проверим, так ли это?

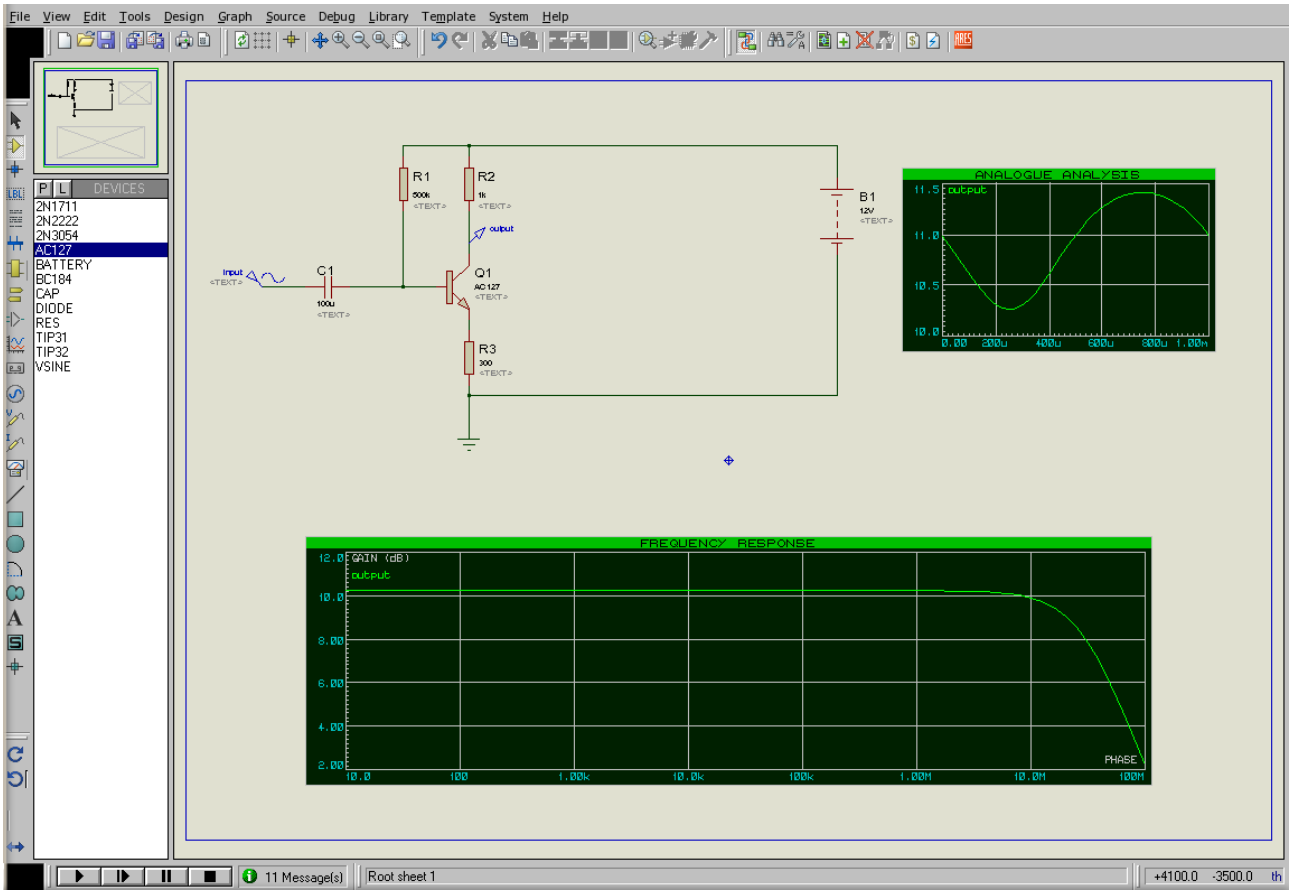


Рис. 3.5. Амплитудно-частотная характеристика с отрицательной обратной связью

Нисколько я не развеял сомнения, верхняя частота среза каскада вновь оказывается за 10 МГц, как и предписывает ей теория и практика. Видимо профессионалов не устраивает точность моделирования сравнительно с расчетами или практическим выполнением схем, но в любительской практике, если проверять результаты моделирования на макетной плате, программа окажется достойным помощником.

Проведем еще один эксперимент, который отчасти отвечает на вопрос о применимости Proteus к аналоговым схемам, отчасти на вопрос о том, как работает транзистор?

В самом начале я говорил, что ток базы и ток коллектора связаны соотношением, но никак не назвал это соотношение. Коэффициент «К» — это статический коэффициент усиления по току. Можно встретить его в виде Вст и в виде h_{21}. Это связь между постоянным током базы и коллектора. Но при работе транзистора в схеме нас больше может заинтересовать динамическая связь этих токов. Посмотрим, может ли Proteus помочь нам в этом.

Но предварительно, поскольку мы этого не сделали, найдем этот самый статический коэффициент усиления по току, как отношение постоянного тока коллектора к току базы в выбранном режиме. В схеме рис.3.1 я добавлю два измерителя тока, амперметра, один в цепь базы, другой в цепь коллектора. В свойствах этих амперметров (правый щелчок, в выпадающем меню свойства, затем окошко **Display Range**) я заменю тот, что в цепи базы на микроамперметр, а в цепи коллектора на миллиамперметр.

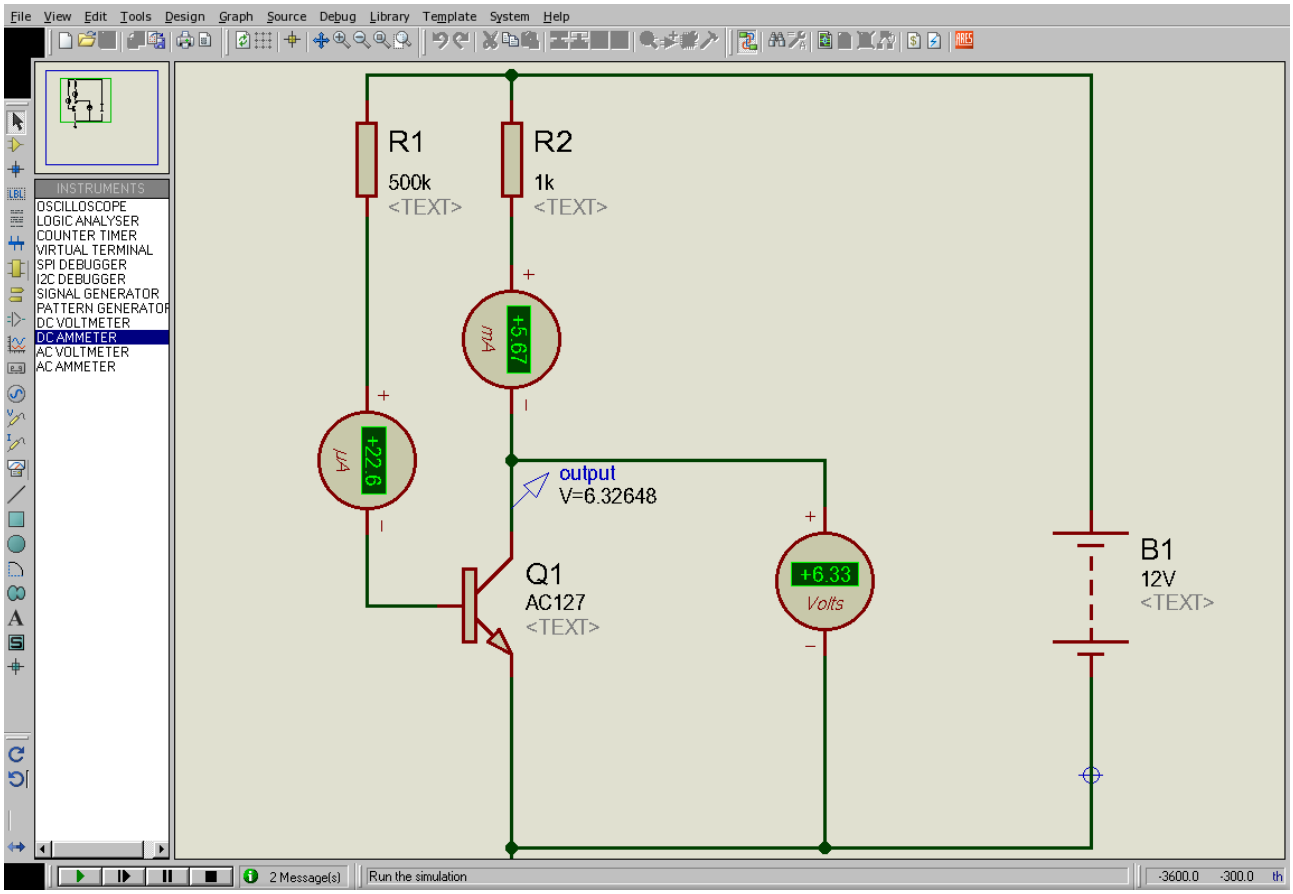


Рис. 3.6. Измерение статического коэффициента усиления по току

Теперь можно разделить 5.67 мА на 22.6 мкА, что даст значение коэффициента, примерно, 250.

Мне хотелось бы проделать нечто подобное со входным и выходным током схемы на рис. 3.4. Токовый пробник к входной цепи добавляется и графика работает, а вот графика, если добавить токовый пробник в коллекторную цепь, работать не хочет. Но это не слишком огорчает меня, поскольку токовый пробник в общей цепи вполне меня устроит, ток в общей цепи — сумма токов базы и коллектора, но ток базы много меньше тока коллектора, так что для ориентировочных расчетов можно взять их сумму.

Можно, конечно, попытаться разобраться, отчего не хочет симулироваться график, если токовый пробник устанавливать в цепь коллектора. К этой проблеме можно вернуться позже, либо не рассматривать это в качестве проблемы до того момента, когда в таком измерении возникнет жестокая необходимость. Пока можно обойтись тем, что есть.

В общем рабочем поле графики немного маловаты, и если это, как мне в данном случае, мешает определить величины, можно выбрать из выпадающего меню после щелчка правой клавиши мышки по графику пункт **Maximize (Show Window)**, что приведет к появлению окна просмотра с большим графиком.

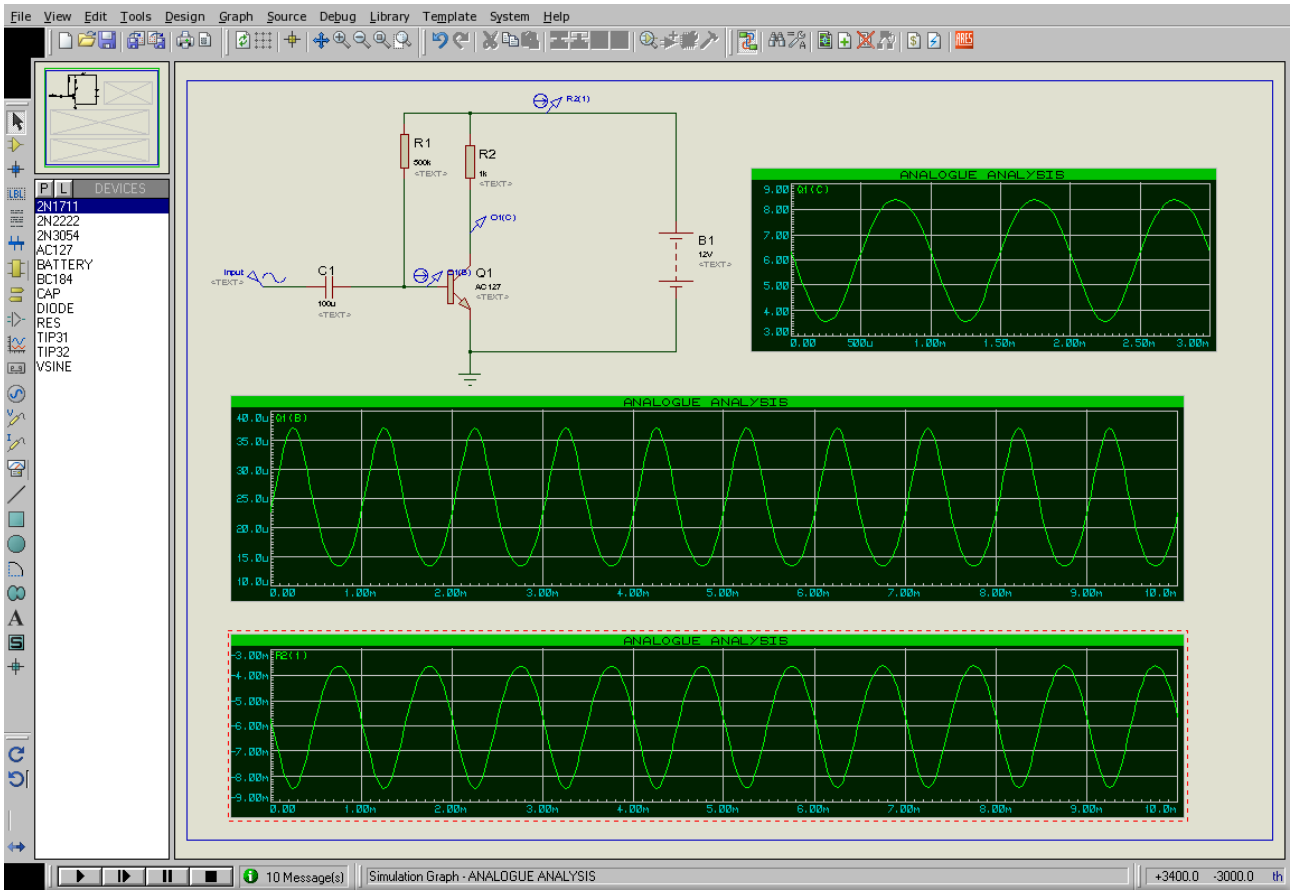


Рис. 3.7. Токи во входной и выходной цепях усилителя

Самый верхний график показывает напряжение сигнала на коллекторе транзистора. В окне просмотра легко выясняется, что двойная амплитуда сигнала около 8.5 - 3.5 = 5 В. Соответственно амплитуда должна быть 2.5 В. Прав я или нет, но при сопротивлении нагрузки равном 1 кОм ток через это сопротивление должен быть 2.5 мА.

Следующий график показывает токовый сигнал базы транзистора, двойная амплитуда которого 24 мкА, а амплитуда 12 мкА.

Последний график — это общий токовый сигнал, как алгебраическая сумма базового и коллекторного токов, который я, ничтоже сумняшеся, принимаю за выходной ток с амплитудой 2.5 мА. В этом случае усиление по току, как простое отношение выходного тока ко входному, будет около 208. Это близко к статическому коэффициенту усиления по току. Кроме того, зная, что входной сигнал равен 10 мВ (RSM) эффективного значения или 14 мВ амплитудного, а выходной сигнал 2.5 В, можно получить усиление по напряжению около 178. Это значение, выраженное в децибелах, дает величину 45 дБ. Это же значение присутствует на амплитудно-частотной характеристике этой схемы. Расчетное значение усиления по напряжению получается около 200. Пока похоже.

В одном из справочников приводится расчетное значение усиления по напряжению как отношение величины сопротивления в коллекторной и эмиттерной цепи для рис. 3.5. В данном случае это будет 1000/300 = 3.3 или в децибелах 20log(3.3) = 10.4. Это значение присутствует на амплитудно-частотной характеристике.

Что ж, был бы рад сказать, что убедился, с аналоговыми схемами работать нельзя, но не убедился пока. Увы!

Микроконтроллеры и Proteus

На одном из форумов я встретил сообщение о разочаровании в Pic-контроллерах, человек, отправивший это сообщение, решил перейти на AVR-контроллеры.

Не знаю, что вызвало его разочарование, но сама постановка вопроса мне не кажется правильной. Попробую пояснить, что я имею ввиду.

Как любой компонент электронной схемы: транзистор, резистор, индуктивность и т.д., — микроконтроллер выбирается на соответствующем этапе разработки устройства. При выборе учитываются многие факторы, порой выбор зависят от храктера предприятия, осуществляющего проект. Один из факторов при выборе конкретного типа — это выполнение задачи данным элементом. Если он выполняет свою задачу, не выходит ни по стоимости, ни по габаритам за пределы заданных параметров, то выбор между PIC и AVR контроллером может определяться только тем, что специалисты предприятия имеют большой опыт работы с одним типом и малый с другим, что совсем не характерно даже для предприятия среднего размера. Или выбор может быть обусловлен возможностями постоянного поставщика элементной базы. Но в любом случае невозможно говорить о разочаровании. Другое дело, если предприятие-изготовитель микроконтроллера рекламировало его, делая упор на свойствах, которых не обнаруживается в готовом изделии. Здесь вполне уместно говорить о разочаровании.

Вместе с тем, вопрос о микроконтроллерах дает возможность поговорить о тех свойствах программы Proteus, которые лично меня очаровывают, я не побоюсь этого слова.

Я пока оставлю вопрос о возможности полной разработки электронной схемы, базирующейся на микроконтроллере, включая написание и отладку программы на ассемблере или языке высокого уровня для микроконтроллера, с тем, чтобы вернуться к нему позже. Но даже если для создания программы использовать, положим, MPLAB или Piklab для Pic-контроллера, программа ISIS, составная часть Proteus, позволяет проверить работу не только контроллера, но всего устройства. Для начала очень простой пример. В программе KTechlab я некогда приводил пример создания работающей программы для игрушечного светофора. Нет сомнений, что не составляет особого труда проверить работу этой программы в средствах отладки той программы, в которой код получен, но меня очень порадовало, что в Proteus есть такой элемент, как симулятор светофора. Достаточно выбрать в компонентах микроконтроллер PIC16F628A, для которого создавалась программа светофора, найти в разделе **Miscellaneous** библиотеки компонентов элемент **TRAFFIC LIGHTS**, чтобы после добавления hex-файла программы с помощью диалогового окна свойств микроконтроллера (щелчок правой клавишей мышки и выбор **Edit Properties** из выпадающего меню), где файл добавляется обычным образом через проводник, если воспользоваться кнопкой с иконкой папки в наборе для **Programm File**:, и соединения выводов запустить симуляцию и увидеть воочию работу светофора.

И не то, чтобы это было целью, не то, чтобы я хотел показать безграничные возможности среды разработки Proteus, но легче оценить работу программы при полной симуляции работы устройства, и приятно, что разработчики среды проектирования учитывают нужды даже потребителей простых решений.

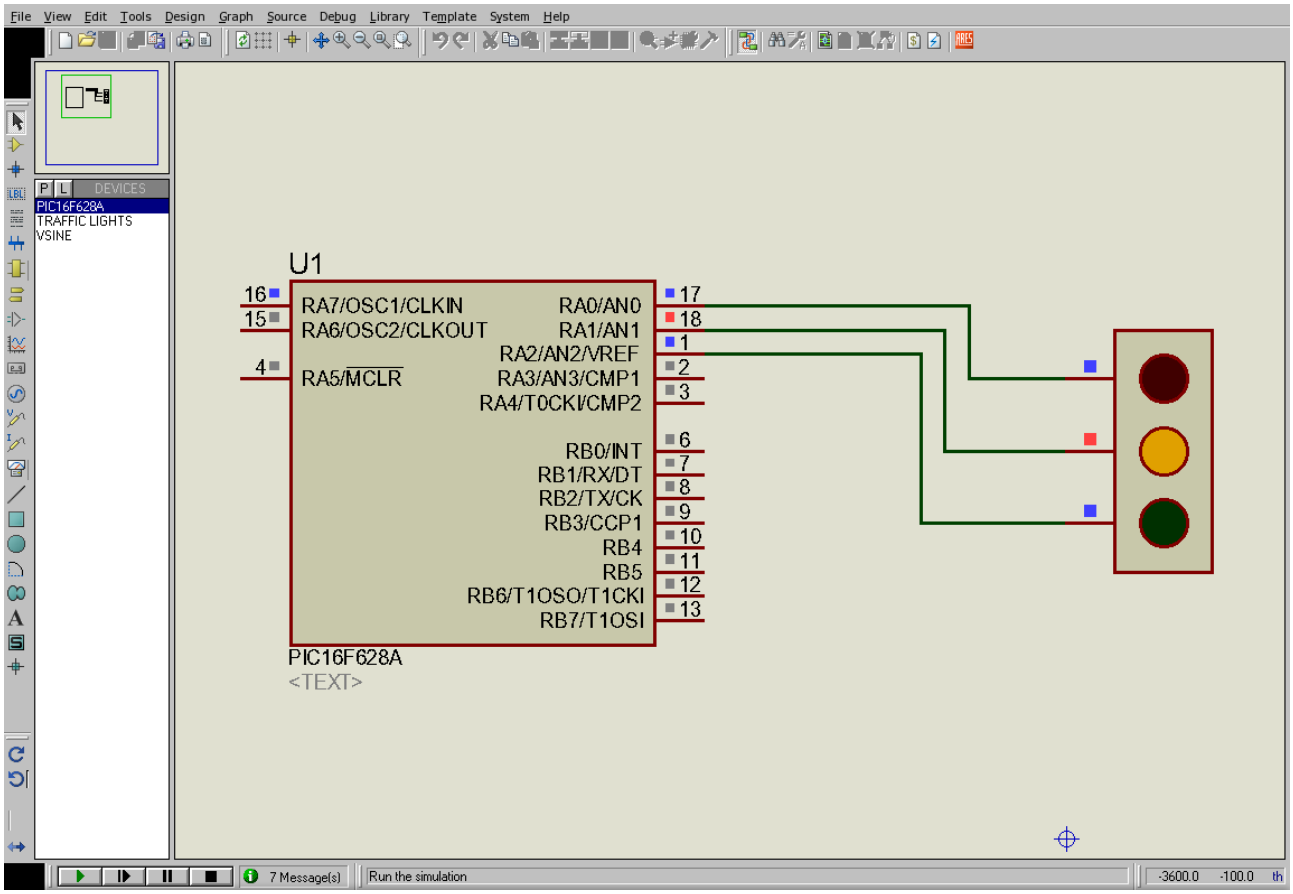


Рис. 4.1. Симуляция работы устройства с микроконтроллером

Чтобы показать, что возможности проверки в Proteus позволяют быть применены к более сложным решениям, я использую другую программу, созданную ранее для PIC16F628A. Несколько предварительных слов о программе и схеме.

Схема на рисунке ниже и программа для микроконтроллера предназначались для демонстрации начинающим любителям электроники, что они могут не только повторять готовые простые электрические схемы, но и подступиться к разработке более сложных устройств. Устройств, которые можно реализовать в виде готового изделия, чтобы порадовать младшего брата или сестру в качестве забавной игрушки, или могут быть использованы для автоматизации каких-то домашних процессов. Само устройство, к проверке которого в Proteus я перехожу, это релейный модуль. Он должен получать команды извне по интерфейсу RS485 в виде строк R00$0N от компьютера или центрального управляющего устройства, по которым включать (или выключать) соответствующее реле. Последнее, если выбрать реле с контактами, способными коммутировать, например, обычную лампу для освещения, будет включать или выключать свет. Команда вида R00$0S запрашивает состояние соответствующего реле, на которое контроллер отвечает R00#0N, если реле включено, или R00#0F, если выключено. Для упрощения рисунка я не добавлял микросхему интерфейса RS485, но это мало влияет на конечный результат. Схема изображена с двумя реле, лампы на схеме, опять-таки для упрощения схемы, 12-вольтовые и подключены к тому же источнику питания, что и реле. В качестве источника команд использован терминал программы Proteus, а сигналы команд в линии отображаются на экране осциллографа.

Более подробное (может быть, излишне подробное) описание модулей есть в моей книге «Умный дом своими руками», и здесь я опущу все детали, включая программу, но результат эксперимента в программе Proteus приведу. Мне нравится то, как Proteus справляется с

задачей.

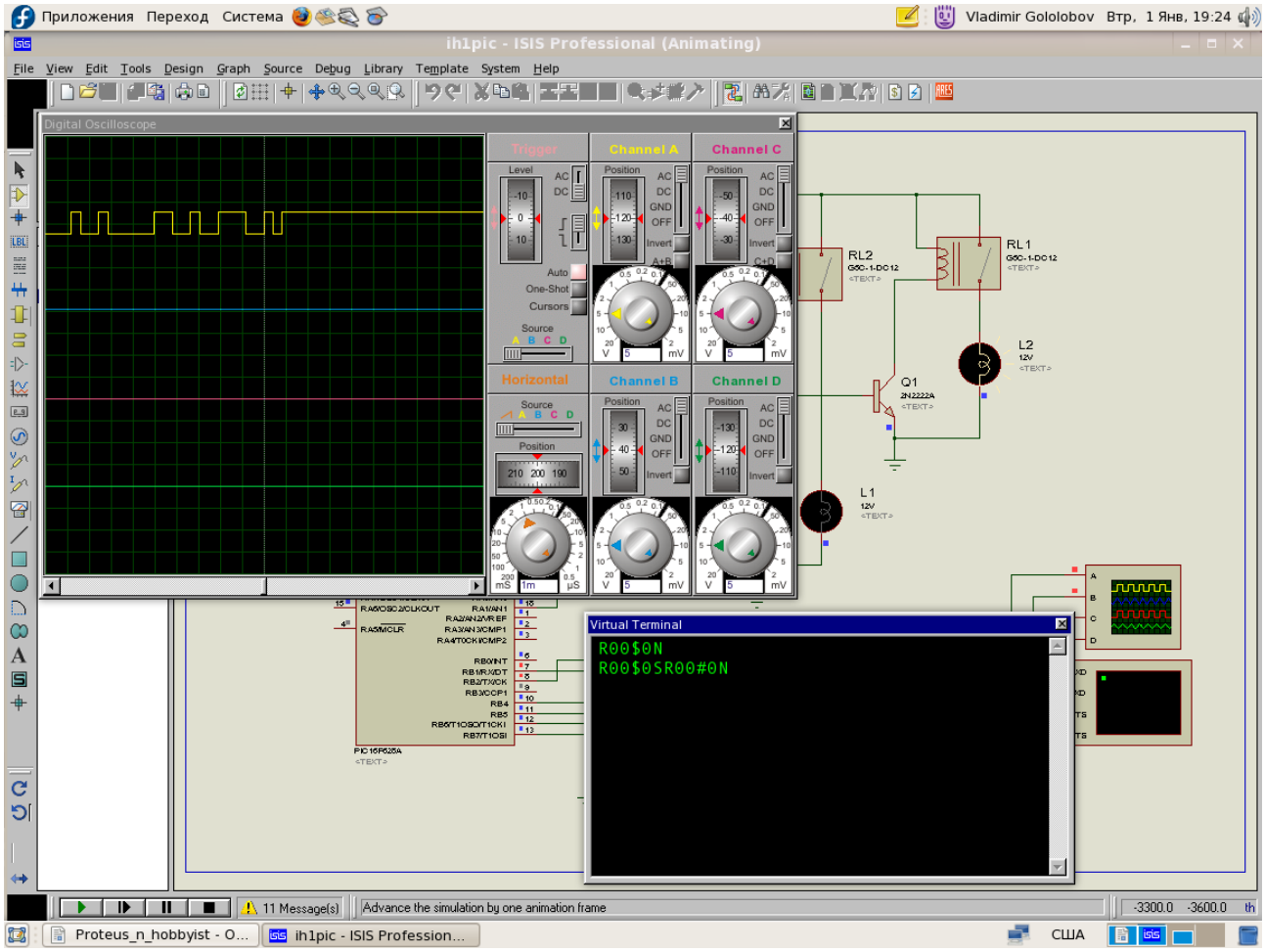


Рис. 4.2. Эмуляция работы релейного модуля на PIC16F628A

Как видно на рисунке лампа послушно включается и выключается командами терминала системы проектирования Proteus.

Я допускаю, что профессионалов может что-то не устраивать в работе программы. Возможно, никакая программа не может быть столь универсальна, чтобы удовлетворить всем потребностям всех профессионалов, но мне она очень нравится, и я пока не придумал, что бы такого она не могла сделать. Более того, ранее мой знакомый, занимавшийся конвертером Чука, сообщил мне, что симуляция этого конвертера в программе Qucs не проходит. Собственно, уж не знаю правильно ли в количественном плане, но в качественном и в программе Qucs симуляция проходит, но без применения источника переменного напряжения и выпрямителя. Профессионалы, работающие с силовой электроникой, справедливо предпочитают такие программы, как PSIM и SwCADIII, специализирующиеся на этих задачах, но меня заинтересовало, пройдет ли симуляция в Proteus.

Я вновь не готов утверждать, что графика в точности соответствует реальной работе устройства, для этого мне понадобилось бы потратить некоторое время на чтение документации по этому вопросу, что не отвечает моим намерениям, но симуляция этой схемы проходит, а результаты очень похожи на те, что получены в других программах.

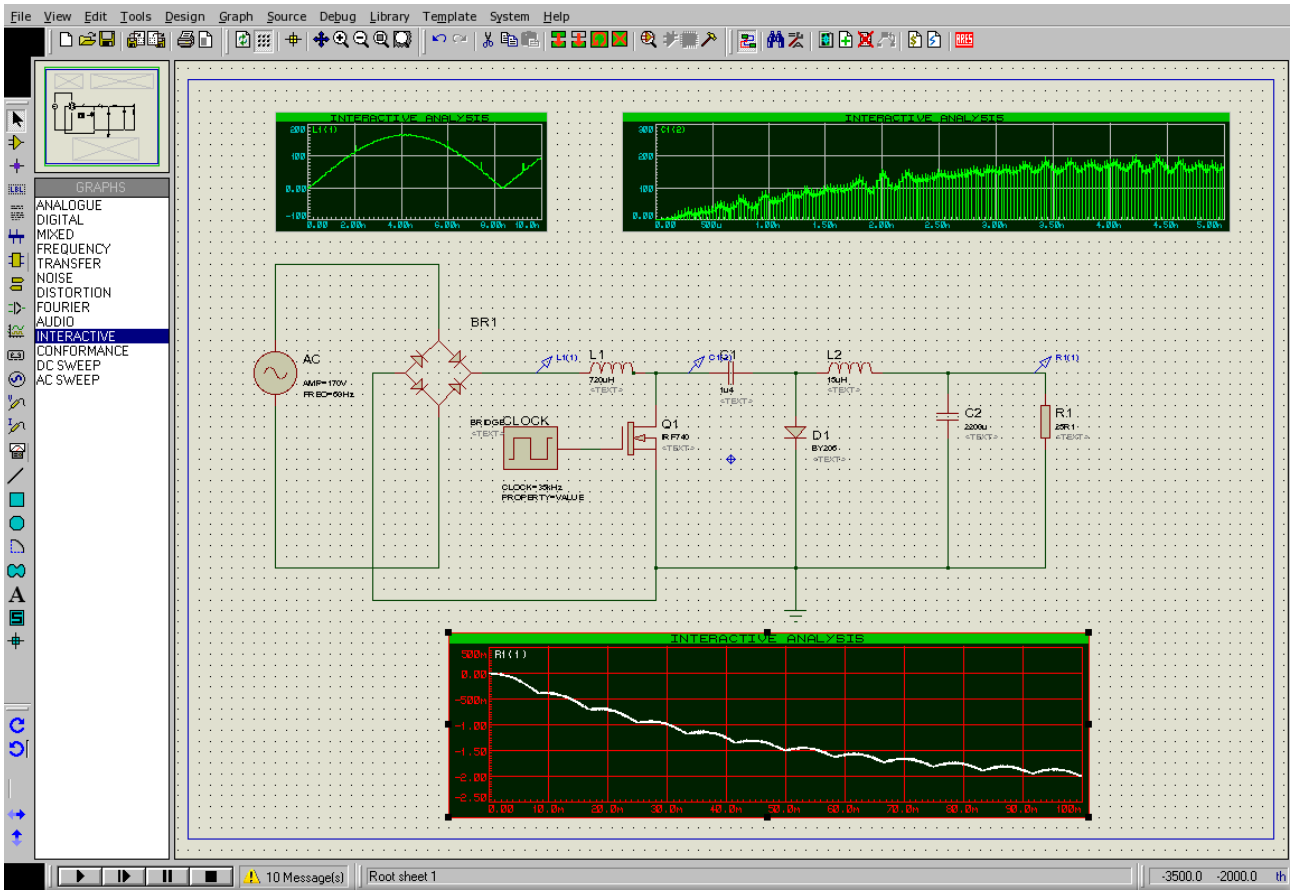


Рис. 4.3. Симуляция работы конвертера Чука в программе Proteus

Столь обширная область применения Proteus, по моему мнению, должна в полной мере удовлетворять запросам любителей. Профессионалы, я думаю, с удовольствием будут использовать программу там, где она их устраивает, и искать другие решения, там где они хотели бы большего, в частности создавая свои версии САПР. Но для этого должны быть серьезные причины.

Однако вернемся к микроконтроллерам. В поставку системы Proteus входит множество примеров и, в частности, программа для микроконтроллера PIC16F84. Следуя руководству пользователя, находим раздел SAMPLES, затем Tutorials. В нем готовый проект Traffic.DSN и исходный текст на ассемблере TL.ASM:

```
            LIST    p=16F84 ; PIC16F844 is the target processor

                    #include "P16F84.INC" ; Include header file
                    CBLOCK 0x10   ; Temporary storage
                    state
                    l1,l2
                    ENDC

                    org     0           ; Start up vector.
                    goto    setports    ; Go to start up code.

                    org     4           ; Interrupt vector.
halt                goto    halt        ; Sit in endless loop and do nothing.

setports            clrw                ; Zero in to W.
                    movwf   PORTA       ; Ensure PORTA is zero before we enable it.
                    movwf   PORTB       ; Ensure PORTB is zero before we enable it.
```

```
                bsf     STATUS,RP0   ; Select Bank 1
                clrw                 ; Mask for all bits as outputs.
                movwf   TRISB         ; Set TRISB register.
                bcf     STATUS,RP0    ; Reselect Bank 0.

initialise      clrw                  ; Initial state.
                movwf   state         ; Set it.

loop            call    getmask       ; Convert state to bitmask.
                movwf   PORTB         ; Write it to port.
                incf    state,W       ; Increment state in to W.
                andlw   0x04          ; Wrap it around.
                movwf   state         ; Put it back in to memory.
                call    wait          ; Wait :-)
                goto    loop          ; And loop :-)

                ; Function to return bitmask for output port for current
state.
                ; The top nibble contains the bits for one set of lights and
the
                ; lower nibble the bits for the other set. Bit 1 is red, 2 is
amber
                ; and bit three is green. Bit four is not used.
getmask         movf    state,W        ; Get state in to W.
                addwf   PCL,F          ; Add offset in W to PCL to calc. goto.
                retlw   0x41           ; state==0 is Green and Red.
                retlw   0x23           ; state==1 is Amber and Red/Amber
                retlw   0x14           ; state==3 is Red   and Green
                retlw   0x32           ; state==4 is Red/Amber and Amber.

                ; Function using two loops to achieve a delay.
wait            movlw   5
                movwf   l1

w1              call    wait2
                decfsz  l1
                goto    w1

                return


wait2           clrf    l2
w2              decfsz  l2
                goto    w2
                return
                END
```

Далее, следуя инструкции, выберем исходный файл (если это уже не сделано в проекте): раздел основного меню Source пункт Add/Remove Source Files, где в окне диалога можно выбрать и исходный файл, и нужный компилятор.

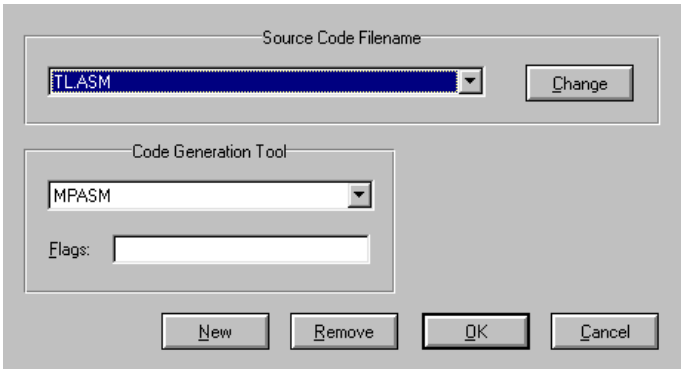


Рис. 4.4. Диалоговое окно выбора исходного файла и компилятора

И завершает подготовку добавление hex-файла к схеме через выпадающее меню и раздел редактирования свойств компонента, где можно указать файл TL.HEX.

Программа содержит преднамеренную ошибку, о чем есть предупреждение в руководстве:

> *There is, in fact, a deliberate mistake in the above code...*

Кроме того, руководство сообщает, что hex-файл получен с помощью MPASM, и в случае использования другого компилятора, вам следует:

> *Note that If you are planning to use a new assembler or compiler for the first time, you will need to register it using the* ***Define Code Generation Tools*** *command.*
>
> *Заметьте, что если вы намерены использовать новый ассемблер или компилятор, вначале вам нужно зарегистрировать его, используя команду* ***Define Code Generation Tools****.*

Запуск симуляции клавишей **Play** приводит к тому, о чем сообщает руководство, зажигаются два огня светофоров и ничего больше не происходит — работает преднамеренная ошибка.

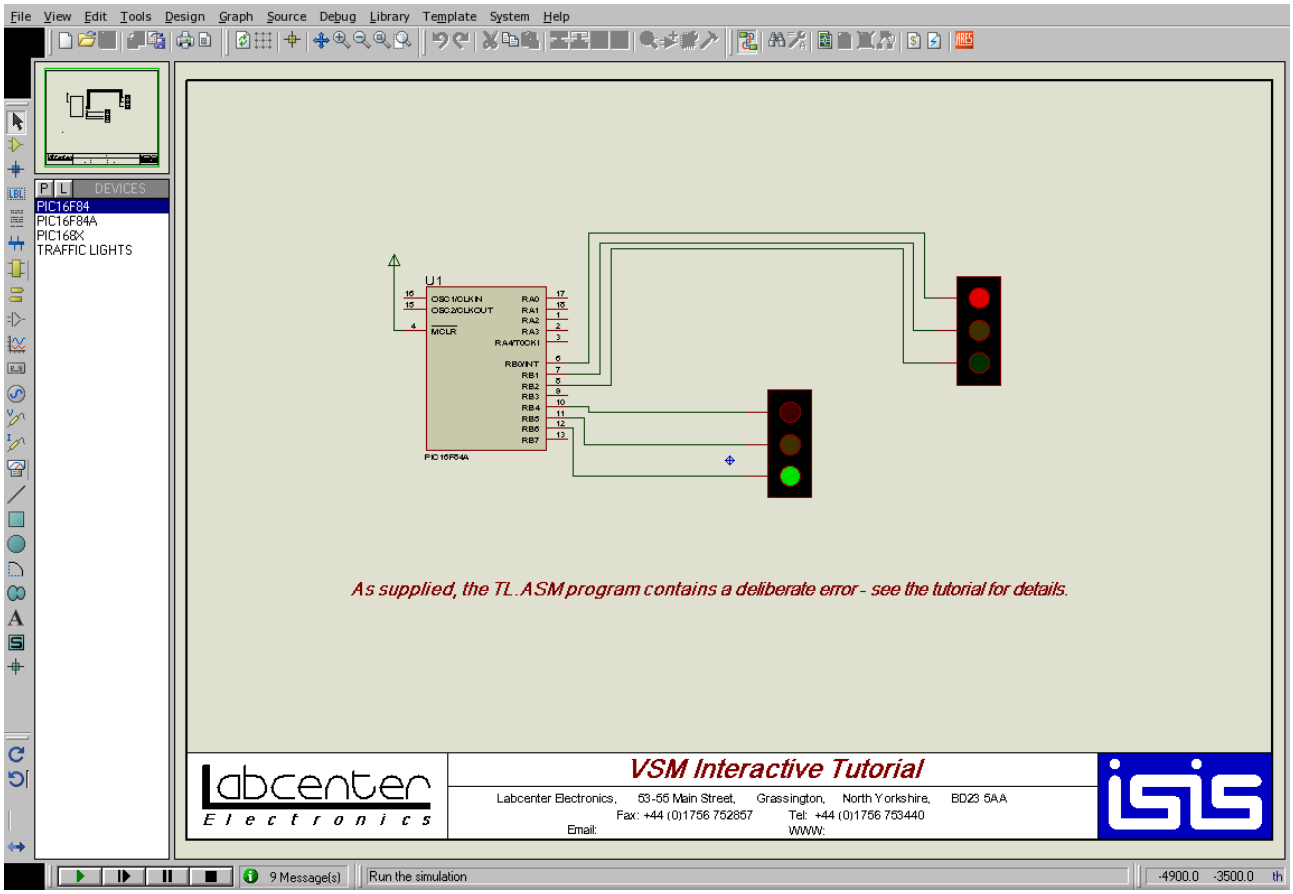


Рис. 4.5. Пример для отлаживания программы из набора Proteus

Последуем далее за руководством. Для отладки рекомендуется нажать Ctrl+F12 или выбрать соответствующий пункт в разделе основного меню **Debug**. В этом же разделе после того, как откроется окно с кодом ассемблера можно открыть другие окна отладки: окно состояния регистров, наблюдения и т.д. Откроем рекомендуемые руководством окна наблюдения.

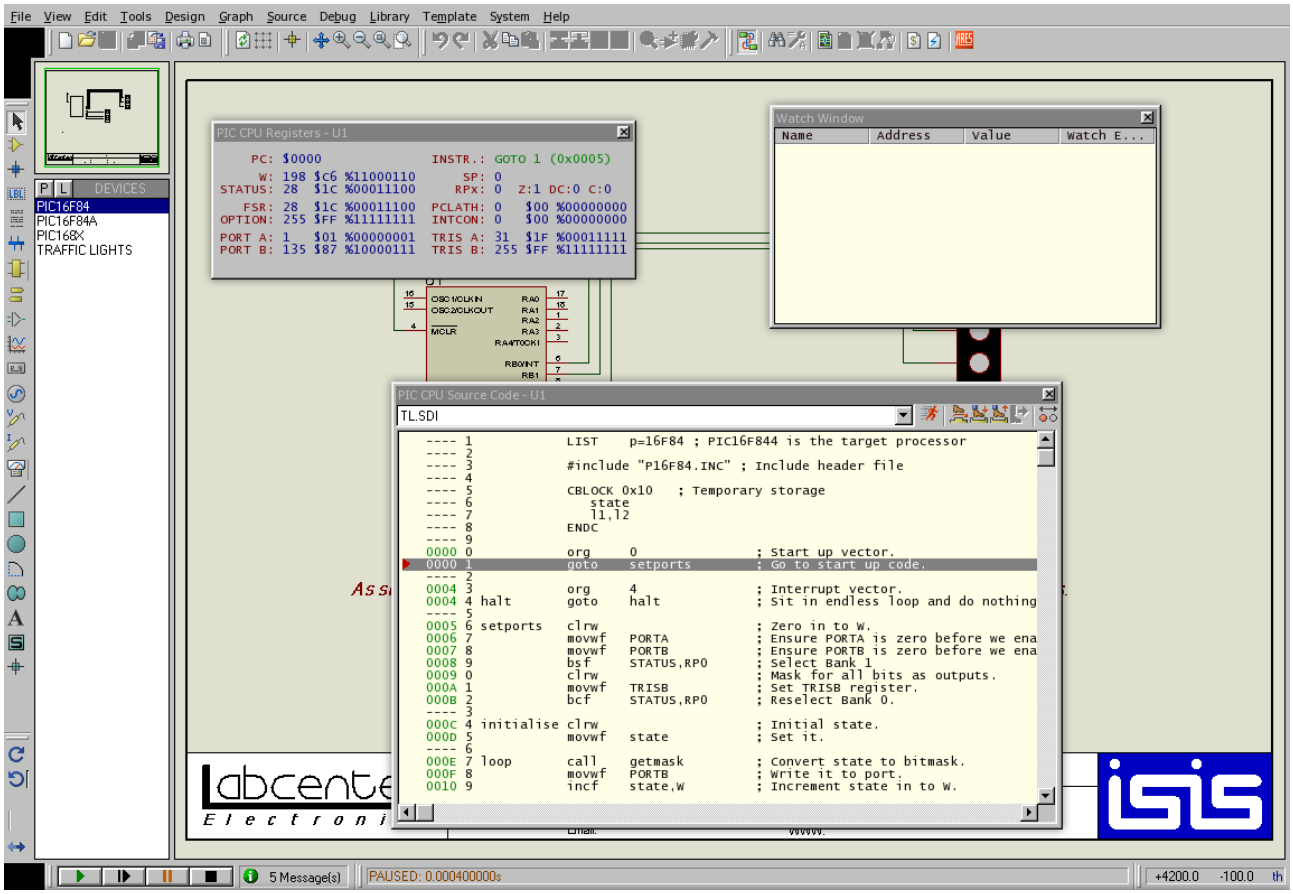


Рис. 4.6. Окна наблюдения за состоянием микроконтроллера при отладке

В окне с исходным кодом можно задать или снять точку останова, используя клавишу F9 или выбирая эту команду из выпадающего меню, получаемого после выделения левой клавишей мышки нужной строки, по которой следует щелкнуть правой клавишей мышки. Руководство рекомендует выбрать строку 000E, где начинается цикл (отмеченный в ассемблерном коде *loop*). После задания точки останова клавишей F12 запускается симуляция, доходящая до точки останова, что можно видеть в строке состояния, где показано, что остановка симуляции по точке останова произошла при состоянии счетчика контроллера 000E, что соответствует заказанной строке для остановки. Клавиша F11 или соответствующий пункт в меню **Debug** позволяют сделать шаг внутрь цикла, красная стрелка, отмечающая перемещения в окне исходного кода, переместится соответственно ниже. При этом, если посмотреть на регистр *w* в окне отображения состояния регистров контроллера, то можно увидеть, что регистр очистился, что соответствует выполнению команды *clrw*.

Далее программа должна перенести содержимое регистра *w* в PORT A, то есть, очистить порт A. Продолжая выполнение инструкций можно в этом убедиться. Кроме использования основного меню и горячих клавиш для пошаговой отладки можно использовать инструментальное меню окна исходного кода. В окно наблюдения, используя выпадающее меню, можно добавить наблюдение за определенным адресом или именованной переменной, выбирая формат отображения. Как описывается далее в руководстве, предопределенная ошибка находится по адресу 0011, где операция *andlw* записана в виде 0x4 вместо 0x3.

Система имеет встроенный текстовый редактор, открывающийся после двойного щелчка по имени ассемблерного файла в пункте основного меню *Source*. В Linux это не получается, появляющийся редактор исчезает немного «помучившись». Но текст ассемблера можно

открыть любым другим редактором, тем более, что после исправления его предстоит транслировать с помощью, я полагаю, MPASMWIN. Возможно, встроенный редактор позволяет сделать это без выхода из Proteus, но невелик труд использовать и внешние средства. Правда, пакет MPASM, входящий в поставку системы, который можно найти в папке Tools, приходится перенести на диск C:. Без этой процедуры появляется ошибка при трансляции, но, тем не менее, ошибка исправлена, текст оттранслирован. Теперь в программе Proteus можно заменить hex-файл новым и запустить программу на исполнение.

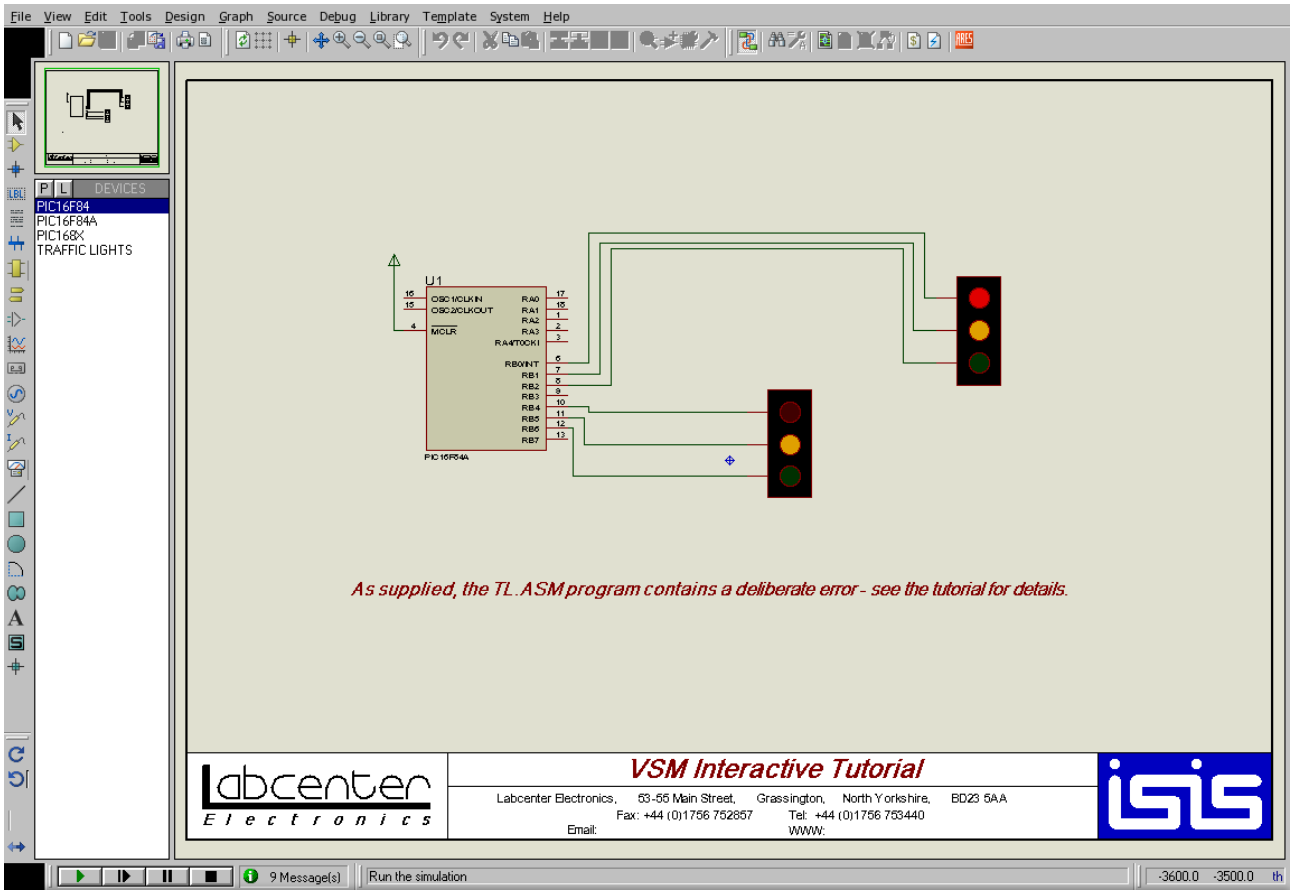


Рис. 4.7. Выполнение демонстрационной программы после исправления ошибки

Программа работает, огни светофора последовательно переключаются. При необходимости в отладочных операциях, а иногда такая необходимость возникает, можно использовать для наблюдения сигналов осциллограф или графические средства отображения сигналов после расстановки пробников напряжения в нужные точки, например, на выводы микроконтроллера. При выборе типа отображения графика для демонстрационной программы естественно применить *DIGITAL*. Дальнейшая работа с графиком проводится обычным образом: рисуется окно графики, добавляются трассы наблюдения и запускается симуляция графика. Для удобства наблюдения сигналов в свойствах микроконтроллера можно изменить частоту с 10 кГц на 1 МГц.

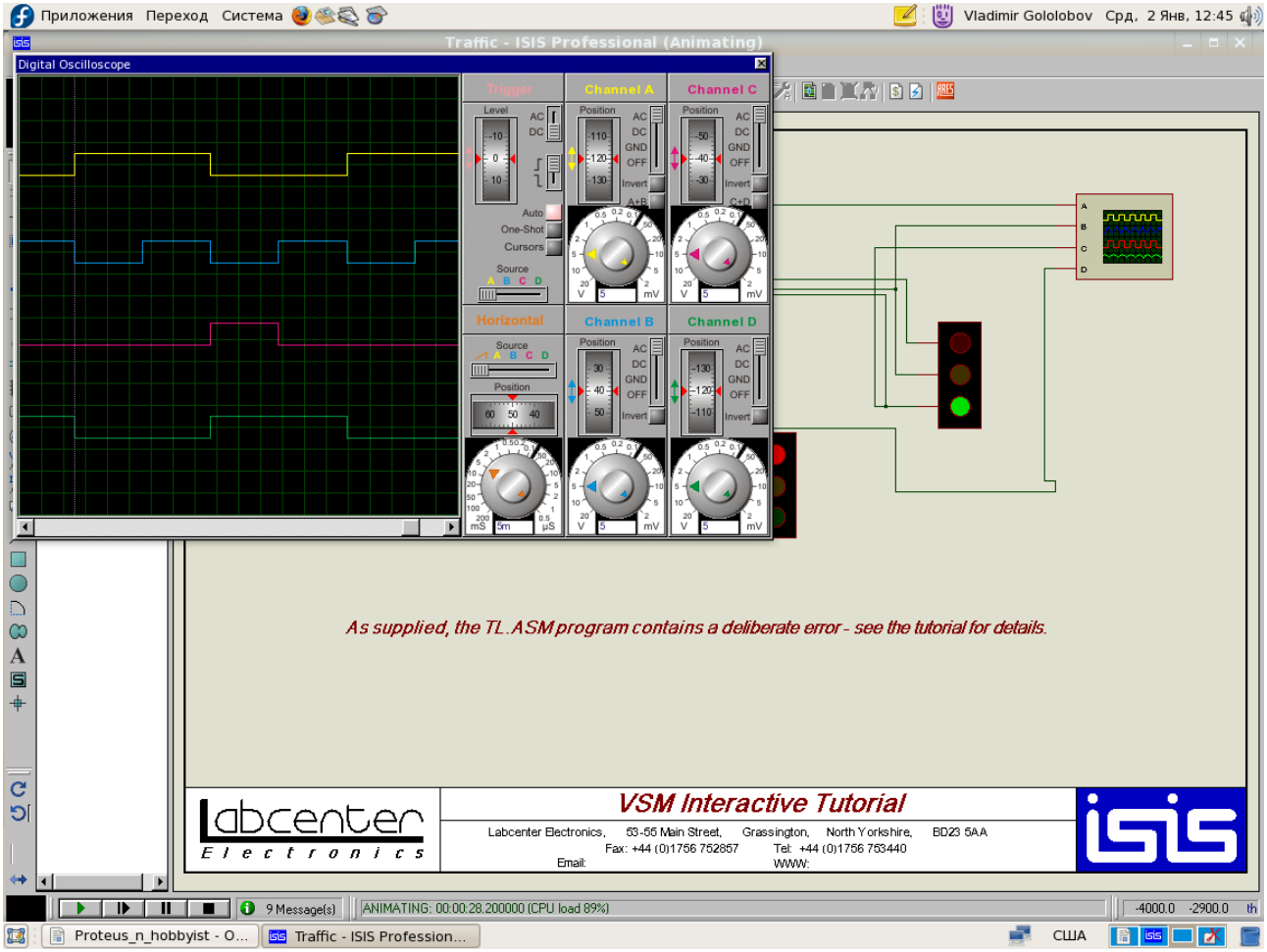


Рис. 4.8. Наблюдение сигналов на выводах контроллера с помощью осциллографа

Словом, среда разработки Proteus предлагает все удобные средства для разработки устройств, использующих микроконтроллеры, включая средства отладки программы. В простых случаях, как в демонстрационной программе, преимущества Proteus не столь очевидны, но если вы используете достаточно много дополнительных элементов, возможность увидеть не только сигналы на выводах микроконтроллера, но и их «дальнейшую судьбу», невозможно переоценить.

Остается добавить, что я использовал систему Proteus в Linux под эмулятором Wine, который изначально предназначался для Windows игр. Система разработки достаточно сложна, она никак не задумывалась для запуска в Linux, в Windows ее работа значительно удобнее, например, в плане исправления ошибок в тексте во внутреннем редакторе — достаточно исправить ошибку, запустить команду **Build All** в разделе **Source**, чтобы получить все необходимые для дальнейшей работы файлы.

Использование Proteus в Linux, готов согласиться, чистейшей воды спекуляция. Но даже то, что удалось попробовать, а в этом я убедился собственноручно, вполне достаточно для любительской практики. Мне кажется, если купить Proteus, то будет смысл «повозиться» с тем, что не работает в Linux, возможно, оно заработает. Например, не слишком утруждая себя можно заменить встроенный редактор на внешний, в качестве которого подойдет *notepad*, блокнот Windows. Достаточно в разделе **Source** основного меню зайти в диалог настройки с помощью пункта **Set External Text Editor...**, где указать блокнот в качестве внешнего редактора и теперь можно открыть ассемблерный текст для правки в блокноте. Или, не

проходит трансляция исходного текста, значит можно использовать MPASMWIN и MPASDDX из перенесенной в корневой каталог диска C:\ папки MPASM. Эти изменения производятся в диалоговом окне настроек через раздел **Source** основного меню, где есть пункт **Define Cod Generation Tools...** Диалоговое окно после внесенных изменений может выглядеть следующим образом:

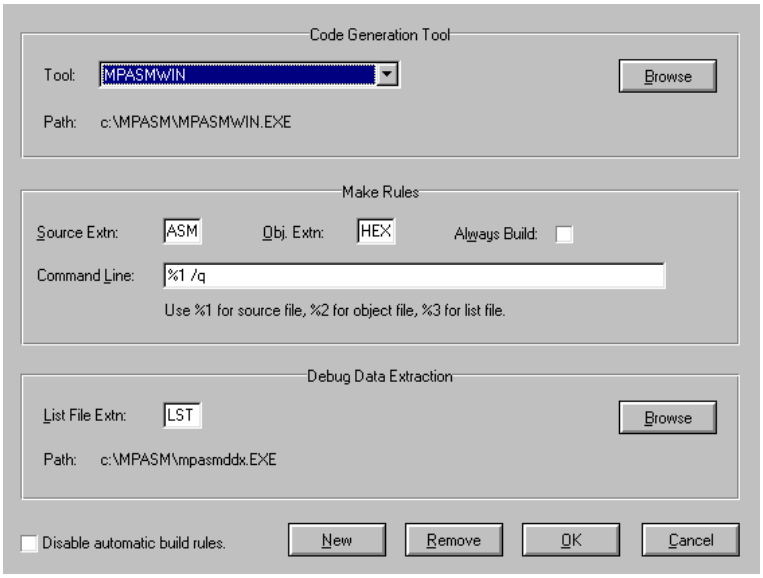


Рис. 4.9. Диалоговое окно настроек компилятора ассемблера

Кроме этих изменений лучше, хотя это может получаться только у меня, но лучше перенести файл проекта и все файлы исходного текста в свою папку, например, /home/vladimir/asm.

Теперь можно открыть и поправить текст, сохранить его, оттранслировать его, включить файл ассемблерного текста для отладки и загрузить после компиляции (**Build All**) hex-файл в микроконтроллер. Работает отладка, работает контроллер. Остались проблемы с окном наблюдения, оно отказывается работать, но это не самое ужасное.

Повторюсь еще раз, все это чистейшей воды спекуляции, но кто сказал, что этого нельзя делать, если программа работает, а меня больше устраивает работать с ней в Linux? Будут серьезные проблемы, последуют «оргвыводы», не возникнет серьезных проблем, отчего бы нет? Главным остается то, что система проектирования Proteus имеет все необходимые средства работы и микроконтроллерами!

Как работать с линиями?

Пользуясь авторским правом творить произвол, я решил сам задать такой вопрос.

Сегодня любителей интересуют, порой, такие разработки, за которые я и не стал бы браться, слишком много документации предстоит прочитать, прежде, чем выполнишь работу. Вместе с тем многие простые вопросы оказываются за пределами внимания любителей, и они больше склонны доверять мнению профессионалов, чем собственному.

Линии используются очень часто. Это и линии питания, соединяющие разные модули конструкции, это и коаксиальный кабель для передачи изображения, и витая пара, например, компьютерной линии. Proteus предлагает много интересных интерфейсов, таких как USB и сетевая карта, для проведения экспериментов с симуляцией в ISIS. Меня же интересует более простой вопрос, нельзя ли в Proteus увидеть, как ведет себя линия в разных, не свойственных ей ситуациях?

В библиотеке компонентов есть линия с потерями, элемент **LOSSYLINE** в разделе **Modeling Primitives**. Как любая линия с распределенными параметрами она должна обладать рядом характеристик: сопротивление на метр, погонная индуктивность и емкость. Для начала воспользуемся поиском в Интернете, чтобы найти следующий вариант: 9.4 Ом на 100 м, 5.6 нФ на 100 м, индуктивность отсутствует, поэтому решим, что это 1 мкГн на 100 м.

Эти параметры можно добавить в свойства линии, используя редактирование этих свойств. Там же есть возможность задать длину линии. Собственно, я использую два отрезка линии, каждый по 50 м, между которыми добавляю сопротивление 50 Ом. Схема для проведения эксперимента содержит импульсный генератор с короткими и редкими импульсами, сопротивлением 50 Ом, как бы внутреннее сопротивление генератора, и нагрузка всей линии тоже 50 Ом. Я не знаю волновое сопротивление получившейся линии, и выбранные мною резисторы достаточно случайны. Однако первый эксперимент в программе Proteus я бы назвал ознакомительным.

Мне приходилось порой решать простой вопрос, все ли будет хорошо, если линия, а из четырех проводов по двум передавалось питание, имеет сопротивление около 2 Ом. Это небольшое сопротивление, но когда устройство, подключаемое к этой линии потребляет ток порядка ампера на проводах может падать напряжение около 2 В. Если таких устройств не одно, а два или три, то забыть о сопротивлении линии значит впоследствии ломать голову, что не так с системой? Устройства могут работать при напряжении питания ниже номинального, как правило разработчики предусматривают некоторые дополнительные меры по блокированию этого, но пониженное питание по меньшей мере может приводить к неустойчивой работе, зависящей от «настроения» устройства. Бывает обидно обнаружить такую простую ошибку, как неучтенное падение напряжения на проводах. Но не так уж часто случается подобное, чтобы концентрировать на этом внимание.

Однако другие особенности линий, если исследование проблем происходит не в лабораторных условиях, могут доставить большие неприятности. Далеко не всегда в вашем распоряжении есть все необходимое оборудование для проверки линии, да и проверка желательна при отключенном оборудовании, что возможно тоже далеко не всегда, поэтому мне представляется очень удобной возможность в программах САПР провести ряд экспериментов, позволяющих наблюдать, что происходит в реальной линии в тех или иных нештатных ситуациях. На рисунке ниже дополнительное сопротивление R2 в 50 Ом между двумя отрезками линии должно изображать плохой контакт в разъеме.

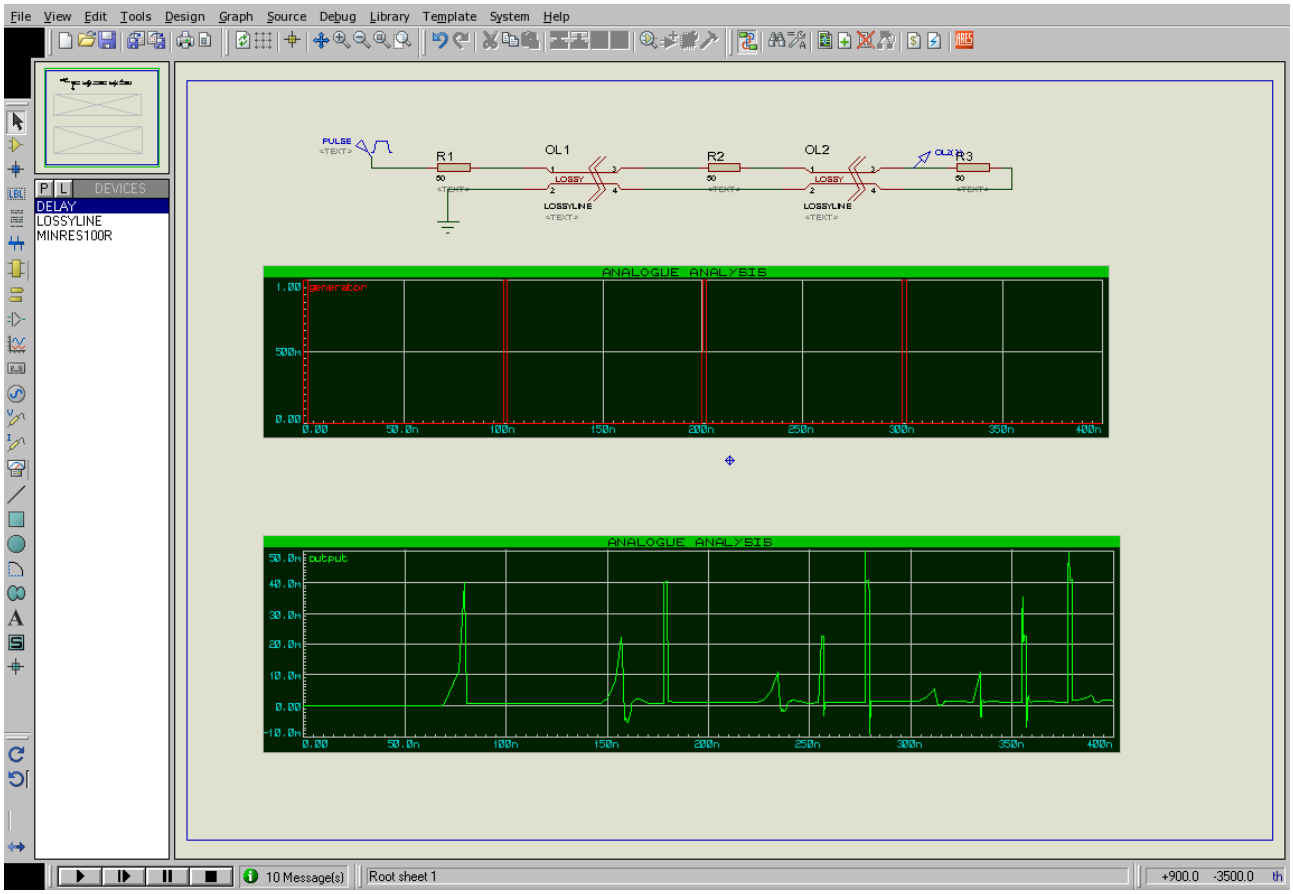


Рис. 5.1. Эксперименты с линией в Proteus

Верхний график — это импульсы от генератора, редкие и красивые прямоугольные импульсы. Многие информационные сигналы имеют такую же природу и переносят сигналы от устройства к устройству по линиям.

Нижняя диаграмма показывает, что первый импульс приходит к концу линии с запозданием в 70 нС, его форма искажена, но это не самое неприятное. Следом за ним, или между двумя дошедшими до конца линии импульсами, есть еще один импульс (150 нС на диаграмме). Этот импульс появляется из-за отражения первого импульса, он опережает второй импульс, но запаздывает относительно первого. В дальнейшем появляются отражения и первого и второго импульсов и т.д. С каждым последующим отражением амплитуда отраженных импульсов уменьшается, но они могут вызывать помехи и при амплитуде меньшей, чем у «правильного» импульса. Чтобы проверить себя я уменьшу паразитное сопротивление R2 с 50 до 1 Ом.

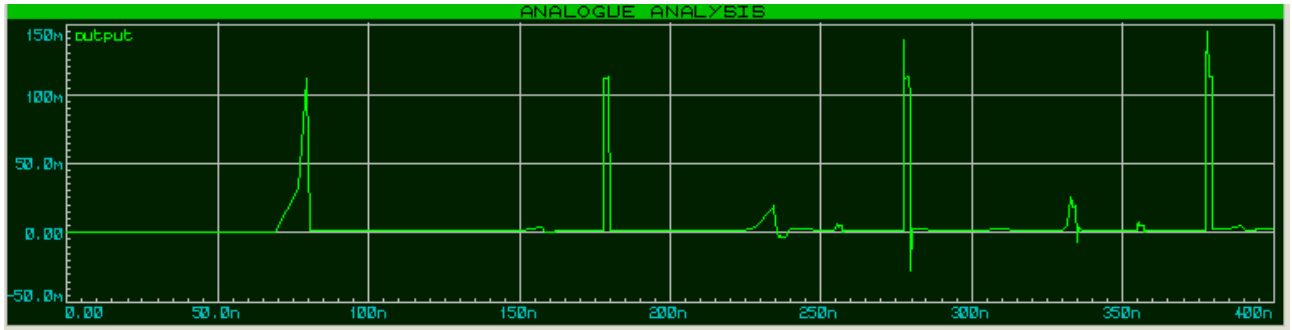


Рис. 5.2. Осциллограмма сигнала в линии при улучшении ее параметров

Паразитный сигнал явно уменьшился. Ликвидация такой неприятности займет одну-две секунды, достаточно вытащить патч-корд и вернуть его на место. Соединение может после этого проработать долгие годы без проблем. Но сколько времени уйдет на поиск этого злополучного места соединения? И очень повезет, если вы начнете решение проблемы не с разборки устройства на другом конце линии.

Пока я рисовал линию, мне вспомнился еще один случай, который мне интересно было бы перенести из давней оказии в проверку с помощью программы. Все знают, что интерфейс RS232 (точнее COM-порт) вполне успешно может работать на расстоянии в 10-15 метров. Реальное расстояние было вдвое меньше, но соединение работало «через пень-колоду». Как выяснилось, из-за монтажной ошибки в одном месте компьютерный кабель UTP-8 переходил в коротенький участок кабеля другой природы. С точки зрения электрического соединения это не должно сказываться на работе, да и на амплитуде импульсов тоже. Но любые неоднородности линии, тем не менее, сказываются на работе сети. Посмотрим, что покажет Proteus, если второй участок линии OL2, короче первого в 10 раз будет иметь другие погонные сопротивление и емкость.

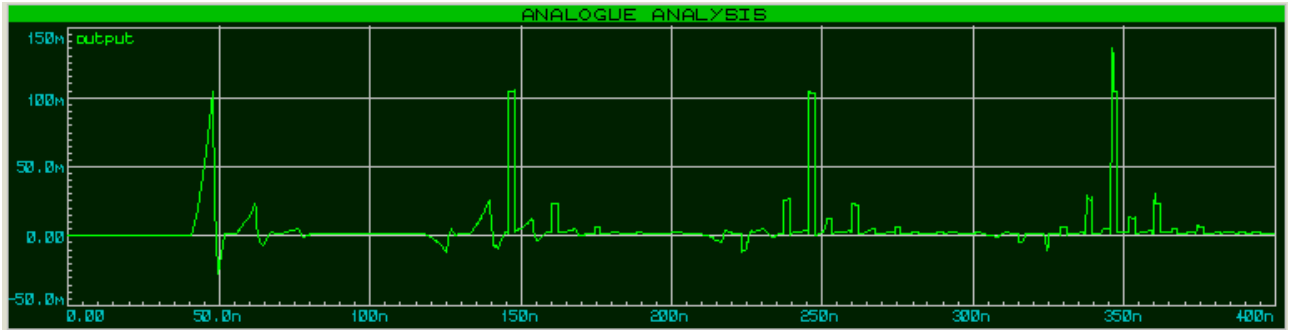


Рис. 5.3. Осциллограмма импульсного сигнала при разнородных участках линии

Неоднородность линии, видимо, вызывает отражения, как и при повреждении соединения. Не скажу, что это для меня откровение, я знаю о существовании приборов для отыскания повреждений в линиях, работающие на принципе отражения сигнала от любых неоднородностей линии. Но прибор, который мне довелось видеть, показывает условное изображение повреждения и позволяет прочитать по ручкам настройки расстояние до места повреждения. Сами сигналы он не позволяет увидеть.

Я не знаю, насколько необходимо в любительской практике исследовать линии, но знать о возможных проблемах полезно, полезно и увидеть поведение импульсных сигналов в линии при наличии разных «несоответствий» в ней. Имея представление о происходящих процессах, легче принять решение о том, что нужно сделать, если, соединив видео-проигрыватель с телевизором случайным кабелем, вместо четкого изображения получаешь дополнительные контуры. Можно заменить кабель, но не всегда. И в этом случае, причина скорее всего в том, что волновое сопротивление кабеля отличается от штатного, можно постараться осуществить согласование с помощью дополнительных устройств.

Все осциллограммы, приведенные выше, соответствуют частоте генератора 10 МГц, но многие линии работают на более высоких частотах. Что произойдет в последнем случае, если увеличить частоту генератора до 100 МГц? Если не забыть уменьшить в свойствах графика время наблюдения в 10 раз, то симуляция проходит быстро, иначе, на моем компьютере, она идет слишком долго, но результат представляет определенный интерес.

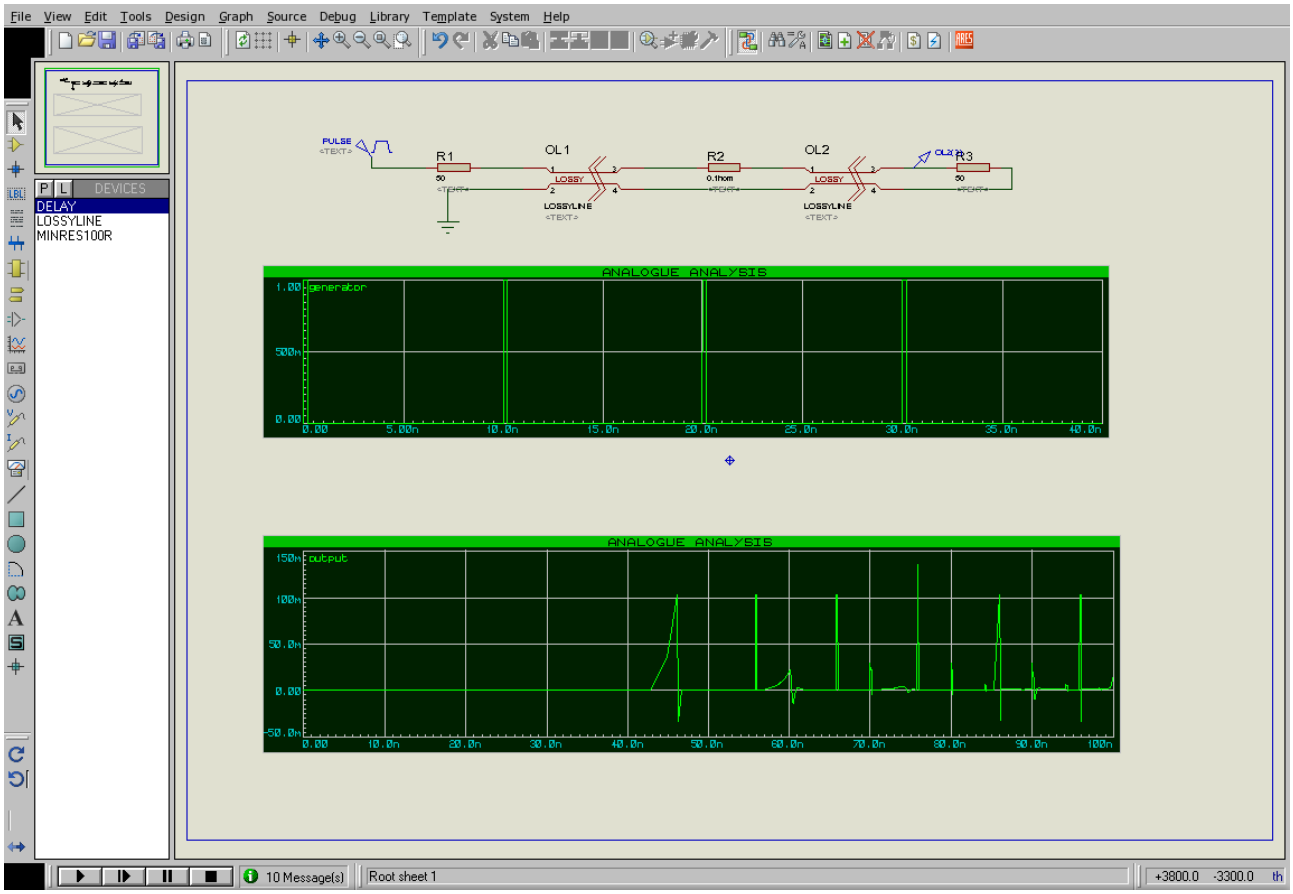


Рис. 5.5. Осциллограмма предыдущего эксперимента при увеличении частоты

Нет оснований углубляться в эту тему, но тот факт, что система проектирования Proteus поддерживает симуляцию и таких процессов, делает заявление о полезности изучения электроники за компьютером отнюдь не голословным.

Помогите найти схему внешнего генератора импульсов 4-8 MHZ (например на 555ЛН1).

Схем тактовых генераторов, просто генераторов и сигнал-генераторов разного назначения и построения можно найти много. Сам вопрос не кажется интересным даже для начинающего любителя, если у него есть несколько подходящих книг по электронике. Но в применении к программам САПР он не столь очевиден. Далеко не все программы, да и далеко не всегда помогают разобраться с генераторами.

Посмотрим, чем может помочь Proteus тем, кому интересно посмотреть работу генераторов в деталях.

Начнем с примера, приведенного в заголовке. Используем микросхему ТТЛ с несколькими инверторами и соберем самую простую схему.

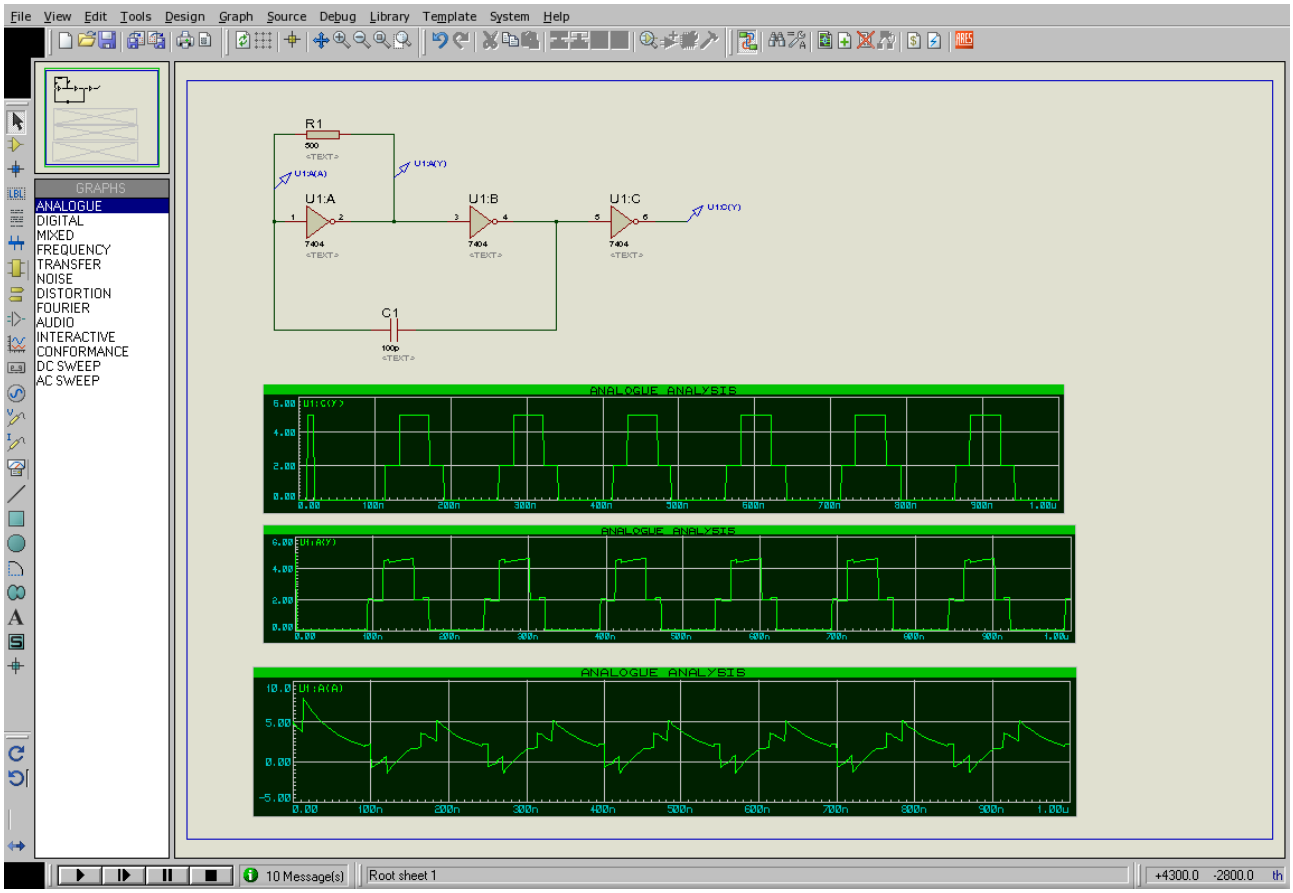


Рис. 6.1. Простой генератор на цифровых микросхемах

Осциллограммы позволяют увидеть сигналы в разных точках схемы, что, в свою очередь, позволяет понять процессы, происходящие в цепи. Иногда при работе с цифровыми элементами в программах САПР у пользователя складывается мнение, что программа работает только на логическом уровне, но не на уровне схемы. Едва ли нижний график можно отнести к логическому уровню.

Уменьшение сопротивления до 250 Ом, а конденсатора до 50 пФ приводит не только к увеличению частоты, но позволяет увидеть, что импульсы (в верхней части) укорачиваются весьма сильно, приближая момент, когда генерация станет неустойчива или совсем прекратится.

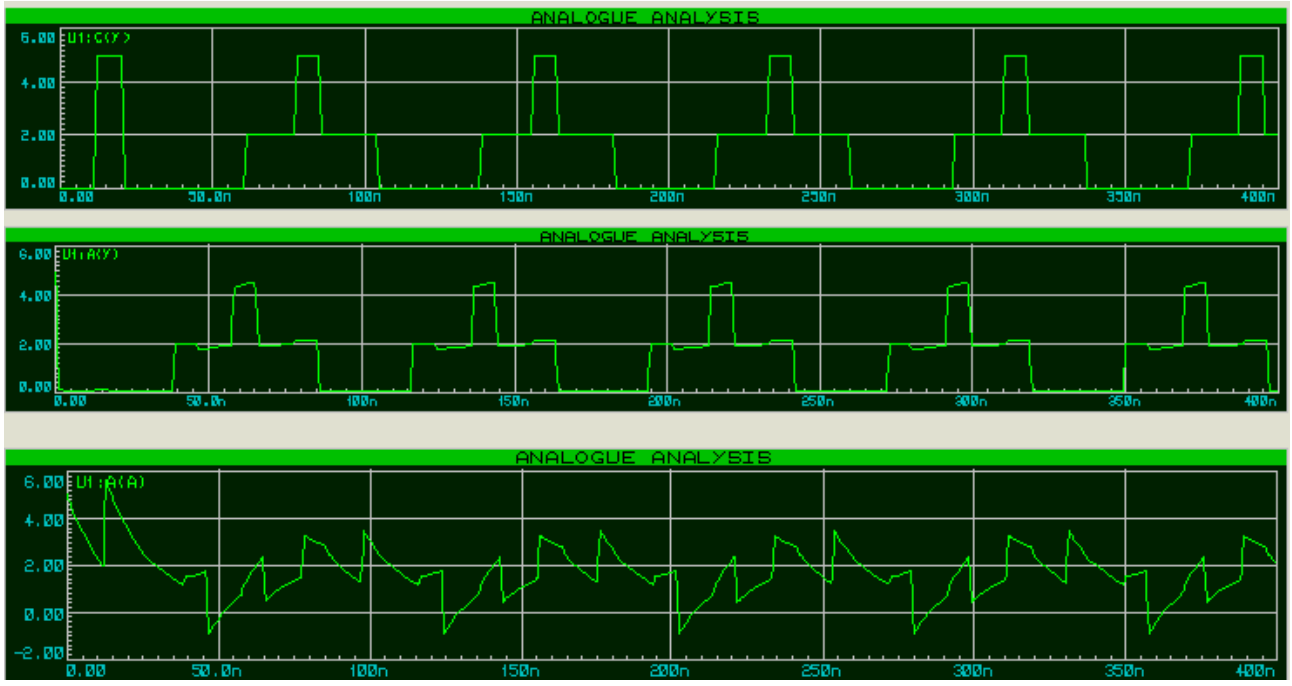


Рис. 6.2. Изменение частоты с помощью значений резистора и конденсатора

Существуют более симметричные схемы генераторов, а для улучшения вида импульсов на выходе, если это требуется, можно использовать D-триггер.

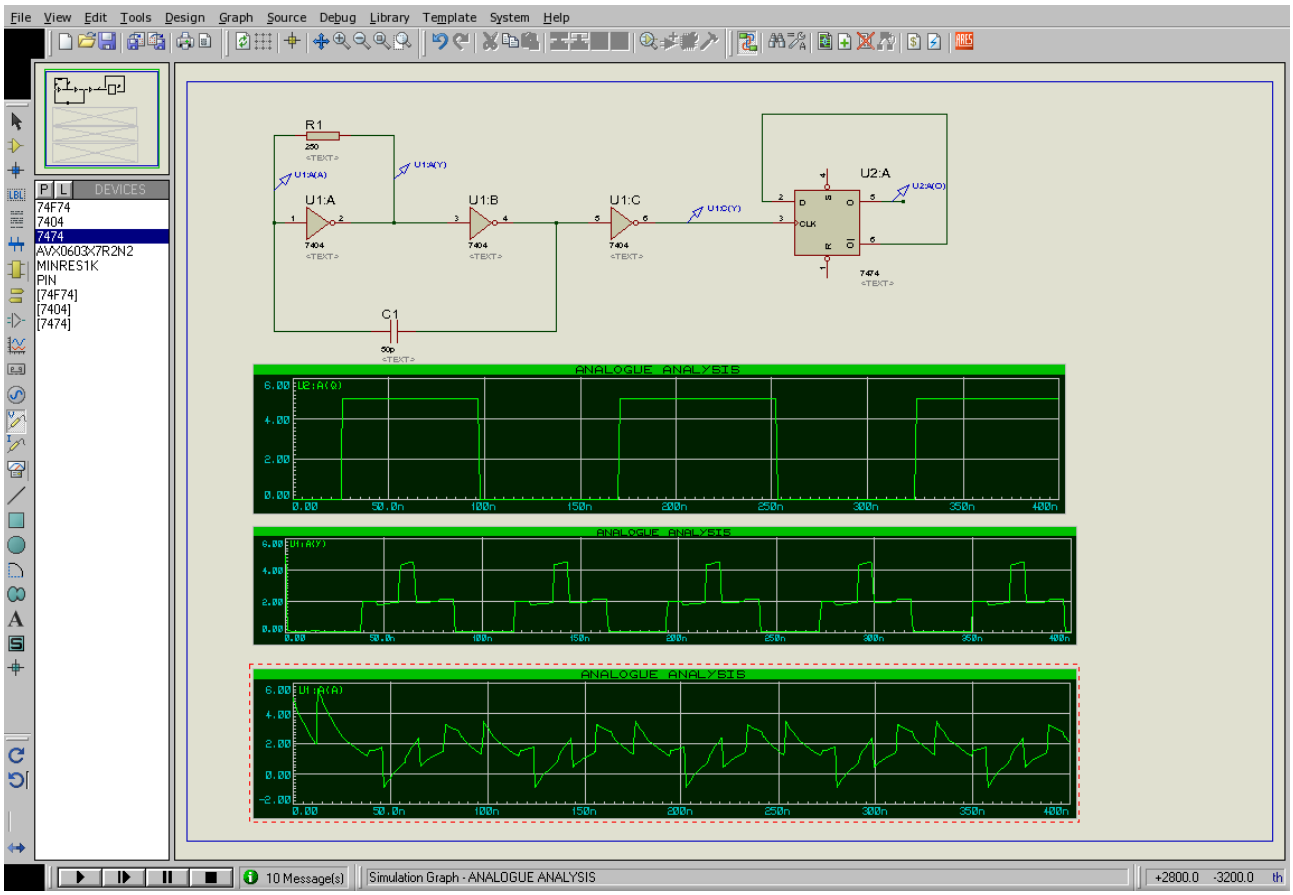


Рис. 6.3. Улучшение формы импульсов с помощью D-триггера

Частота выходных импульсов при таком решении уменьшается вдвое, но форма импульсов заметно лучше. Ряд интересных возможностей представляют цифровые микросхемы для создания генераторов импульсов. Импульсные сигналы, наряду с синусоидальными, наиболее часто используются при наладке и проверке схем. В некоторых случаях хватает «меандра» на выходе генератора, но иногда предпочтительнее короткие импульсы (или длинные). Есть простые схемные решения, позволяющие регулировать скважность импульсов на выходе генератора. При использовании микросхемы серии 4049 схема генератора с элементами R1 = 10 кОм, C1 = 0.5 нФ, RV1 = 1 МОм выглядит следующим образом:

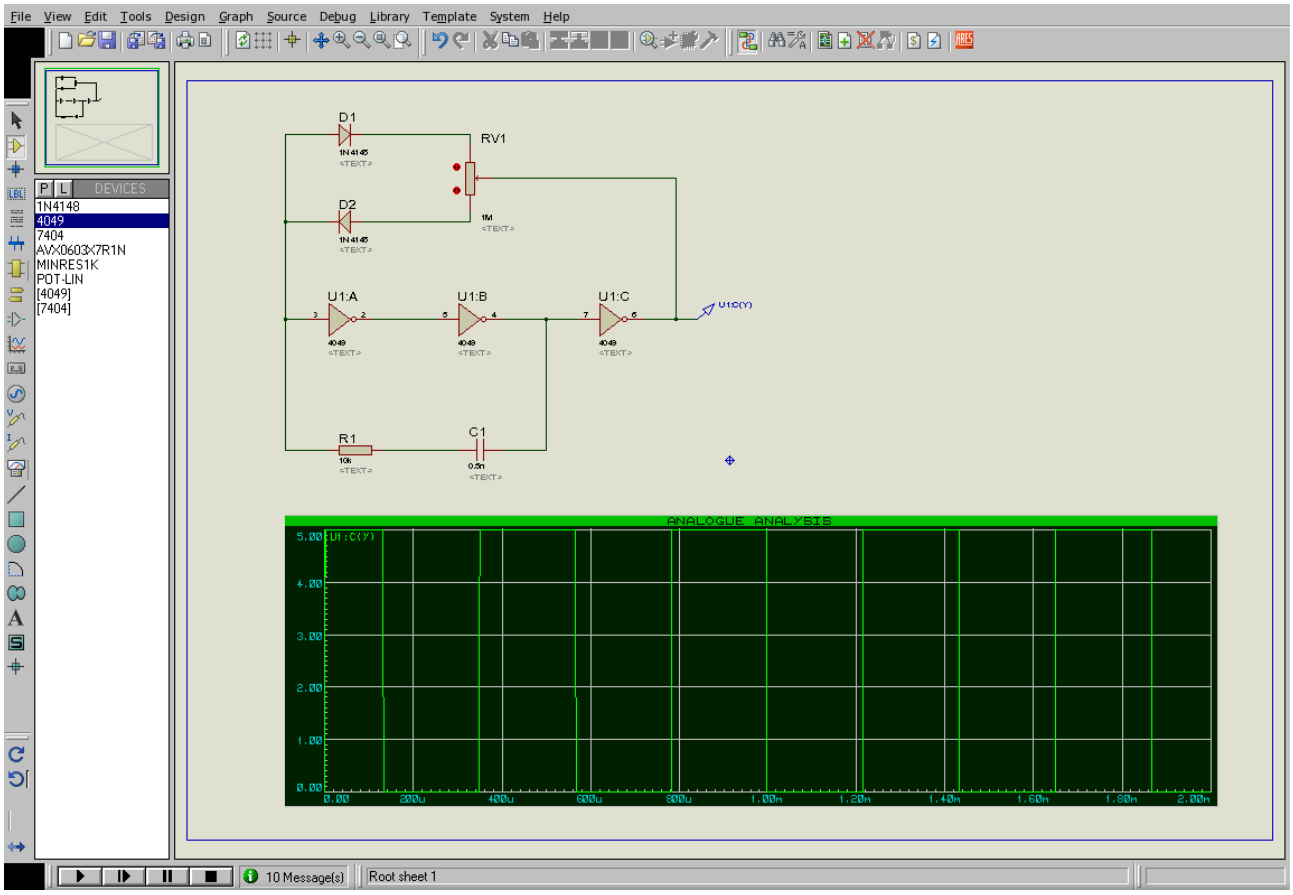


Рис. 6.4. Генератор с регулируемой скважностью импульсов

При смещении движка потенциометра вверх диаграмма получается следующей:

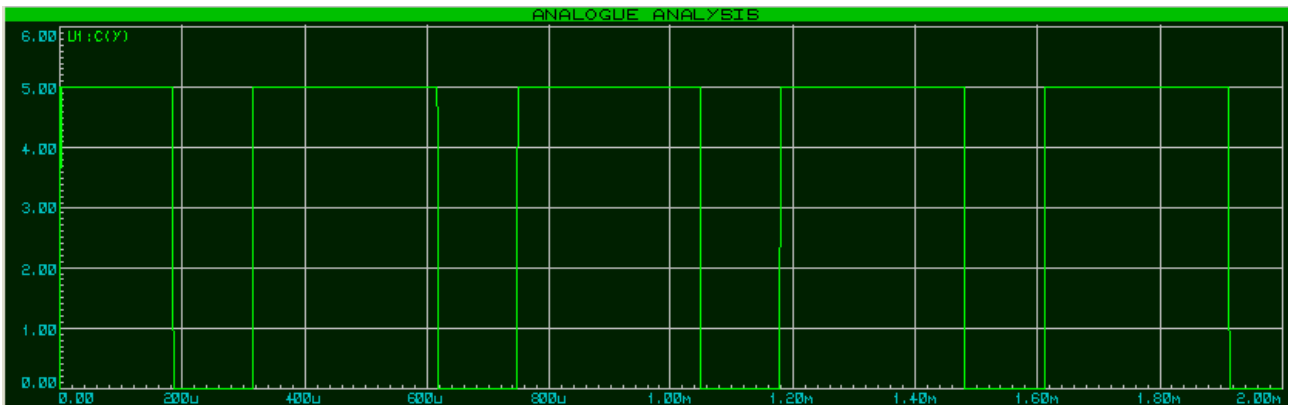


Рис. 6.5. Осциллограмма импульсов при изменении положения регулятора

Иногда возникает интерес к микросхемам таймера 555. И не всегда программа САПР легко справляется с этим аналого-цифровым устройством. Хотя из поставки Proteus можно выбрать красивый интерактивный пример работы микросхемы, я хочу привести схему из книги «Радиолюбительские схемы на ИС типа 555».

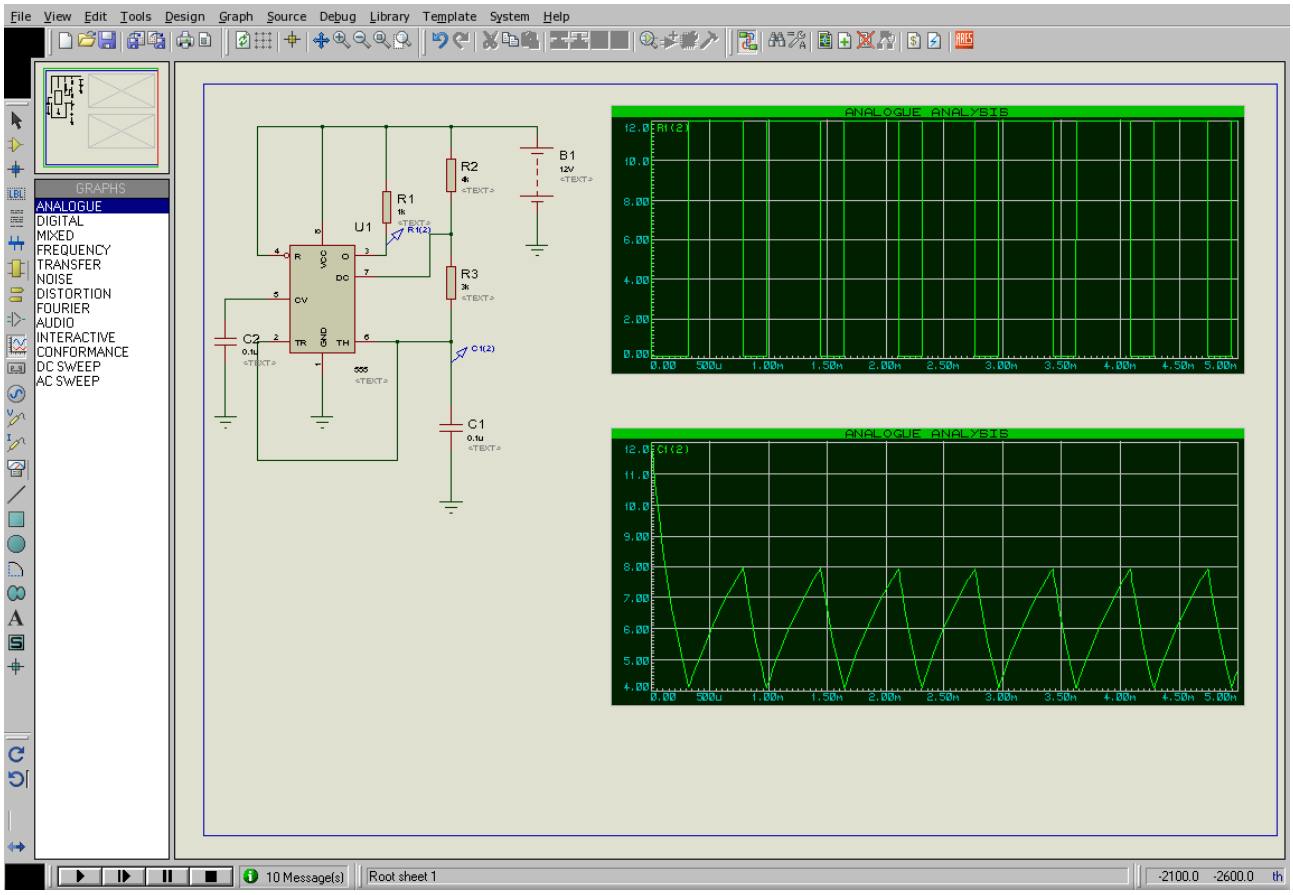


Рис. 6.6. Мультивибратор на микросхеме таймера 555

Желающие могут сравнить диаграммы этого рисунка с теми, что приведены в книге. Меня это сравнение вполне убеждает в возможности работать в Proteus не ожидая особых подвохов со стороны программы.

Мне давно хотелось посмотреть работу одной из схем в программе. Схемы бывают разные, иногда попадаются красивые решения. Каждая из схем по-своему красива, одна оригинальна, другая надежна, третья привлекательна простотой. В работе чаще используешь типовые решения, искать что-то сверх того приходится редко, но еще реже удается найти нечто попадающее под категорию «красивого» решения. Эти решения запоминаются надолго. Некогда мне понадобилось быстро собрать функциональный генератор, нужны были и прямоугольные импульсы, и треугольные и синусоида. Полистав, уж не помню что: журнал ли, книгу, — я нашел решение, которое с тех времен рекомендую всем любителям. Оно запомнилось своей простотой и функциональностью. Я пытался несколько раз воспроизвести его с помощью EDA программ, но безуспешно. Попробую еще раз в Proteus.

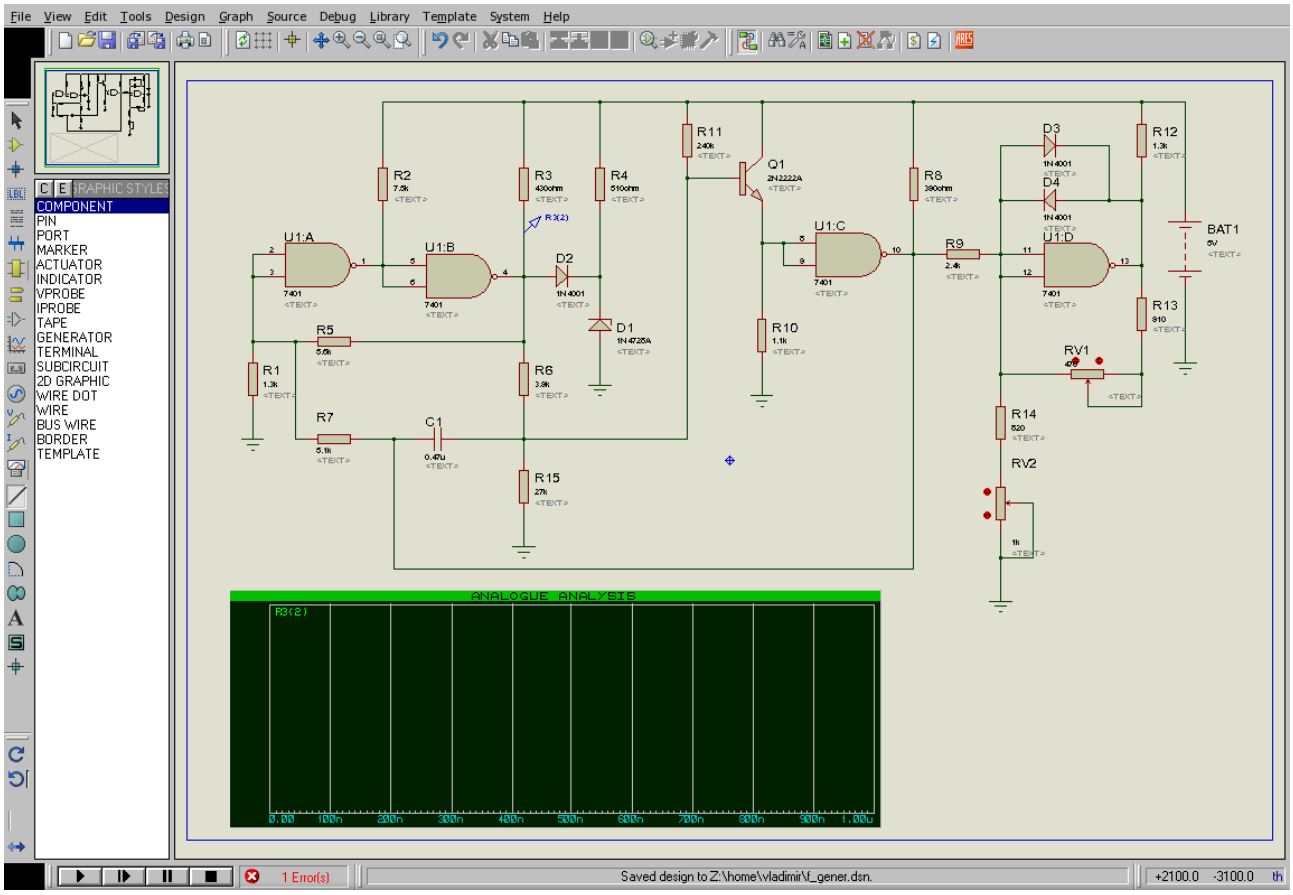


Рис. 6.7. Функциональный генератор на цифровой микросхеме

Симуляция схемы не проходит, появляются ошибки. То, что схема работает, я помню, но ее налаживания я, конечно, не помню. Обратимся вначале к описанию схемы и попробуем разобраться по частям.

> *На элементах U1.A и U1.B выполнен компаратор. На транзисторе Q1, элементе U1.C и конденсаторе C1 построен интегратор, а на элементе U1.D и диодах D3 и D4 преобразователь треугольного напряжения в синусоидальное.*
>
> *Как возникают колебания в генераторе?*
>
> *Положим, на выходе U1.B высокий уровень напряжения, конденсатор C1 начинает заряжаться через резистор R6 напряжением с выхода U1.B, а напряжение на выходе интегратора начнет линейно спадать. Это напряжение через резистор R7 поступает на вход компаратора, и как только оно достигнет величины 0.5В (определяется сопротивлением R5) компаратор переключится в другое устойчивое состояние с низким уровнем на выходе U1.B. Но напряжение на базе транзистора Q1 выше этого напряжения, что заставит конденсатор C1 разряжаться через резистор R6 и выходное сопротивление элемента U1.B. Напряжение же на выходе интегратора при этом будет линейно нарастать, пока не достигнет величины в 3.7В, при которой компаратор вновь вернется в исходное состояние с высоким уровнем на выходе U1.B, а напряжение на выходе интегратора начнет опять линейно спадать.*

Меня несколько смущает упоминание компаратора. Начнем проверку с этой части схемы. Чтобы проверить работу «компаратора», если схема не захочет самостоятельно работать, я хочу использовать генератор треугольных импульсов. Кстати, я не видел такого генератора в

средствах симуляции, так что полезно будет разобраться и с этим.

Для начала я удалю из схемы преобразователь треугольных импульсов в синусоидальные, сохраню проект под другим именем и добавлю внешний генератор. Для интерактивного наблюдения Proteus предлагает сигнал-генератор. Но мне хотелось бы использовать графические возможности, что позволяет легче разобраться в происходящем. По этой причине я выбираю в качестве источника сигнала **PWLIN**.

После добавления источника и соединения его со входом U1.A следует настроить сам сигнал. Обращаемся к свойствам этого генератора.

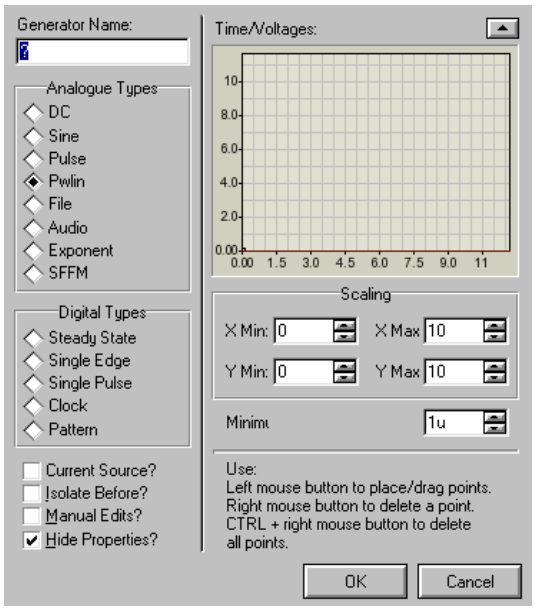


Рис. 6.8. Диалоговое окно генератора PWLIN в Proteus

В правой части окна под надписью **Scaling** есть возможность изменить значения масштаба по оси x и y. В окошке **XMax** я задаю 10m (10 мС), после щелчка в окошке **YMax** графический вид **Time/Voltages** меняется, его шкала x теперь имеет предел 11 мС. Осталось поменять вид сигнала: щелчок левой клавиши мышки в графическом поле в точке с координатами 4.5 мС, 5 В, второй щелчок в точке 9 мС, 0 В, — и я получаю то, что мне нужно. На всякий случай я меняю значение YMax на 5 и щелкаю в окне Minimt. Результат выглядит следующим образом:

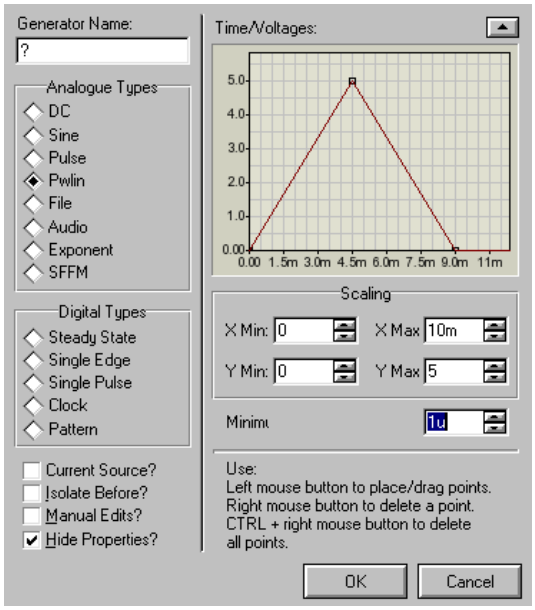


Рис. 6.9. Задание треугольного импульса в окне диалога PWLIN

В описании схемы сказано, что компаратор переключается при напряжении 0.5 В, что определяется величиной резистора R5. Я задаю несколько точек наблюдения, соответственно им рисую несколько графических окон, и намерен проверить, будет ли зависеть напряжение переключения от величины R5. В реальной схеме это возможно имеет место, но симуляция, даже очень удачная, может вести себя иначе.

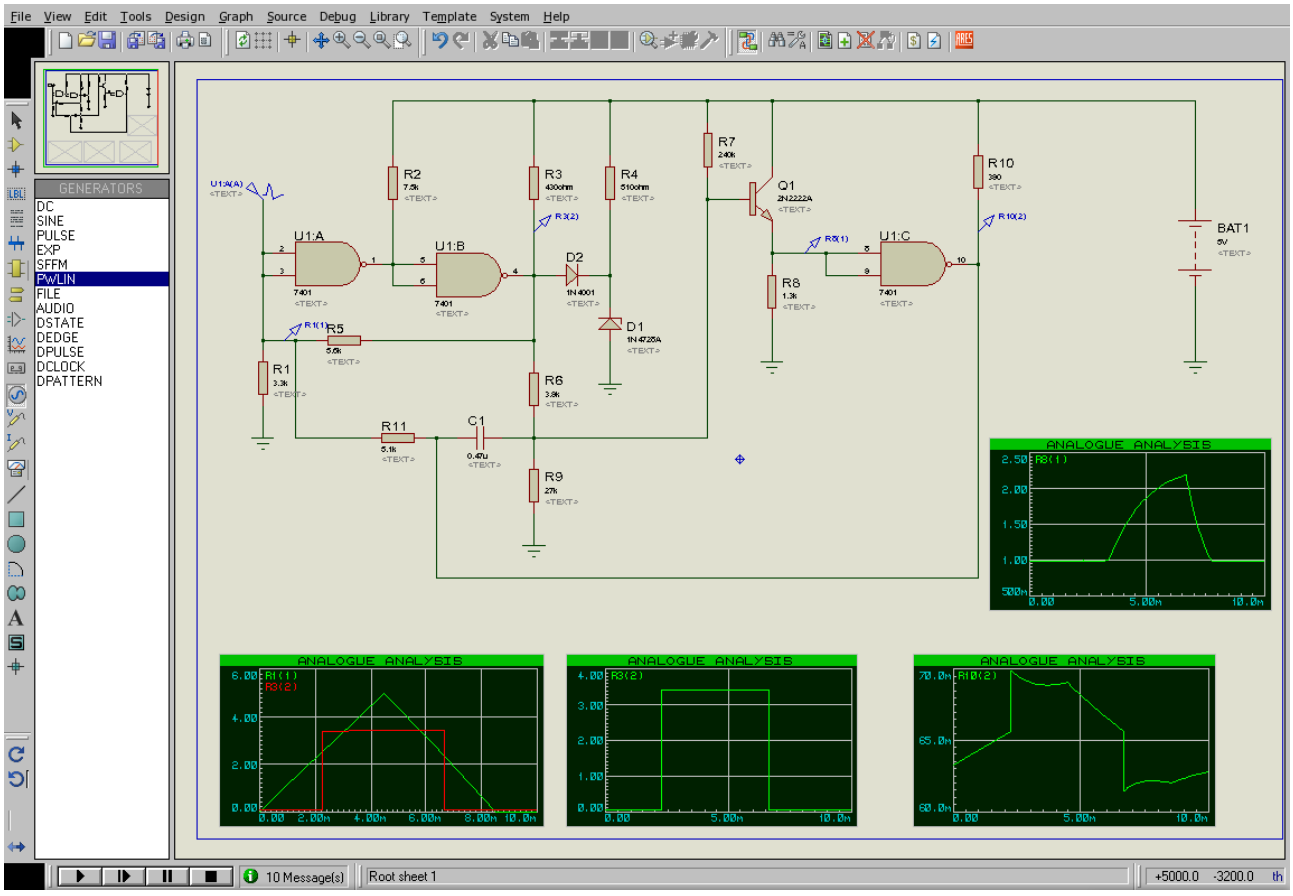


Рис. 6.10. Проверка части функционального генератора на К155ЛА8

Первый график отображает треугольный импульс на входе и выходной импульс компаратора. Получается, что переключение происходит при 2.5 В. Если добавить сопротивление, имитирующее внутреннее сопротивление генератора, то, манипулируя резисторами схемы, можно несколько изменить напряжения переключения, но не в столь широких пределах, как хотелось бы. В равной мере манипуляция с резисторами без дополнительного генератора позволяет добиться некоторого подобия однократного переключения схемы, но это далеко не то, что, по моему мнению, происходит в реальной схеме.

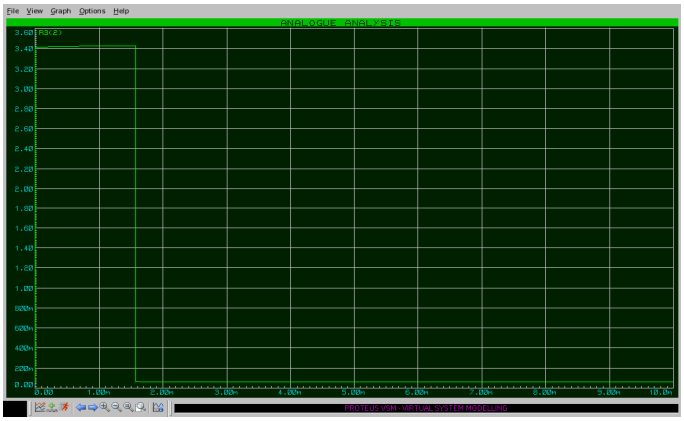


Рис. 6.11. Переключение в схеме при изменении величины резисторов

Я полагаю, что модели цифровых элементов имеют такой параметр, как напряжение переключения или схожий. Эта величина фиксирована для ускорения работы цифровых устройств, а использование цифровых микросхем вне штатных условий, основанное на их усилительных свойствах и возможное в реальных условиях, приводило бы только к затягиванию времени симуляции. Для проверки подобных предположений следовало бы создать свою модель вентильного цифрового элемента на транзисторах, но это выходит за пределы рассказа о системе Proteus.

Что ж, мне и в этот раз не удалось наблюдать работу схемы функционального генератора на цифровой микросхеме, но я познакомился со свойствами Proteus, о которых раньше не знал. Время потрачено не зря.

Собрал одно, собрал другое — не работает. Что делать?

Достаточно часто можно услышать подобный крик души начинающего любителя. Особенно обидно, когда, доверившись солидному изданию, повторяешь схему на самостоятельно изготовленной печатной плате, а схема не работает. Бывалые справедливо советуют читать книги и осваивать теорию. Совет правильный, но трудно сказать, что больше может огорчить в подобной ситуации, то, что схема не работает, или то, что нужно выбросить свою работу и начать читать что-то. Однако можно найти компромисс. В первую очередь, если есть навыки, можно проверить работоспособность активных элементов: микросхем, транзисторов, тиристоров. Если проверить их жизнеспособность не получается, лучше пожертвовать ими, спасая печатную плату, в которую вложено много труда. Микросхемы «выкусывают», располагая кусачки как можно ближе к корпусу, вывод за выводом — эти выводы позже можно легко выпаять. Как правило, микросхемы легко купить, а повторить изготовление платы много сложнее. Аналогично можно удалить транзисторы. Оставшиеся элементы следует проверить, затем проверить правильность монтажа по электрической схеме устройства, исправляя ошибки, если они обнаружатся. И теперь можно перейти к работе с макетной платой, используя все доступные средства для выявления причин неудачи: книги, приборы, программы и советы более опытных товарищей по увлечению. В конечном счете первая неудача быстро превратится в обычную работу с любой электрической схемой, и обязательно пополнит багаж знаний.

Схемы для повторения могут отличаться капризным характером. Одни требуют буквального повторения всей элементной базы и работают только в таком виде, другие требуют тщательной наладки, без которой их нельзя «оживить», третьи могут содержать ошибки, не замеченные при публикации. Хорошо повторяются схемы, использующие типовые решения, наихудшим образом могут повторяться «оригинальные» решения. Иногда складывается впечатление, что автор этого оригинального решения сам не понимает, как работает схема. Читая статью в каком-нибудь издании, где приводятся формулы призванные обосновать выбор решения, обычно не затрудняешь себя повторением всех выкладок, но, вооружившись бумагой и карандашом, можешь к собственному удивлению обнаружить, что приводимые формулы, собственно, никакого отношения к схеме не имеют. Бывает и такое. Но схема работает, во всяком случае, работала у автора, и он повторял ее несколько раз. Ваша попытка повторить схему может оказаться неудачной. Бывает и такое.

Но вернемся к вопросу, заданному в начале этой истории. Что же делать? В первую очередь не следует спешить в принятии решения. Вы несколько раз проверили правильность монтажа и не нашли ошибок. Это еще не значит, что их нет. Отложите плату на время и попробуйте понять, как работает схема. Разбейте ее на функциональные узлы и разберите работу этих узлов, даже если полная работа схемы основана на взаимосвязях этих узлов, каждый из них выполняет часть работы, которую можно проверить. Как это сделать, зависит от ваших предпочтений: на бумаге, где вы рисуете этот узел, на макетной плате, где вы собираете этот узел или в программе EDA, где есть возможность легко менять и схему, и элементы схемы, и режимы работы. Не забывайте только, что не всегда программа может моделировать все, что угодно, и не всегда то, что вы ожидаете от программы, соответствует тому, что в программе происходит. Чтобы не быть голословным, я хочу привести несколько экспериментов со схемой, работу которой, честно говоря, не понимаю. Схема очень проста, но тем не менее.

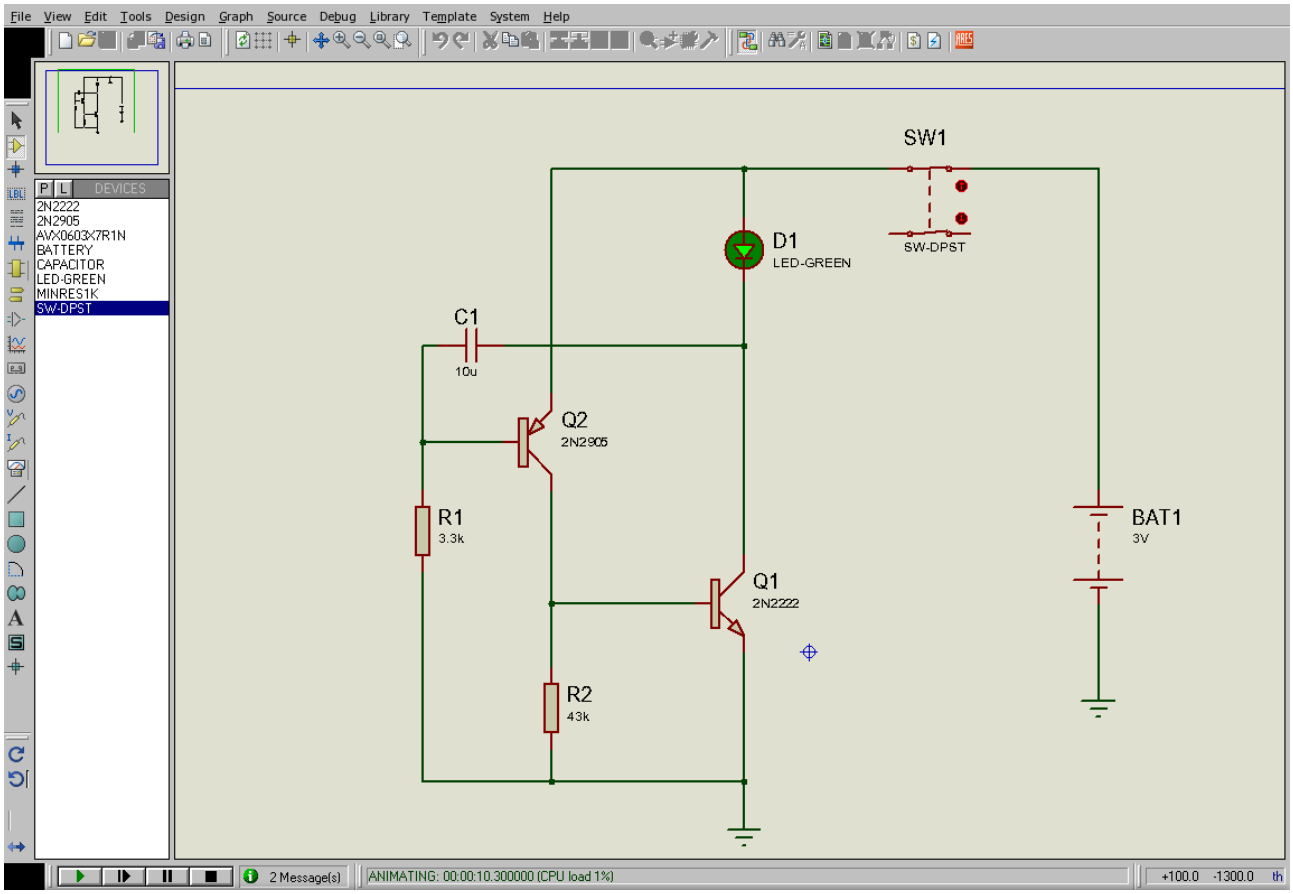


Рис. 7.1. Схема генератора импульсов

Светодиод D1 должен давать кратковременные вспышки с некоторой частотой. Мне понятно, что в процессе образования автоколебаний принимает активное участие конденсатор C1, но как он это делает, мне не понятно. Попытка запустить симуляцию схемы дает тот самый эффект, что и нужен. Схема «молчит». Найти причину этого без понимания работы схемы я, увы, не в состоянии. Поэтому первое, что я собираюсь сделать, это проверить работу программы в части заряда конденсатора — я хочу понять, отрабатывает ли программа этот процесс. Удалим из схемы все кроме резистора, конденсатора и батарейки. Нарисуем график зависимости напряжения на конденсаторе от времени. Ведь, если программа игнорирует это, она не сможет правильно симулировать работу схемы, а я потрачу много времени впустую, пытаясь «оживить» схему.

Из практики я знаю, что конденсаторы большой емкости можно проверить с помощью тестера, включив его в режим измерения сопротивления и присоединив его щупы к выводам конденсатора. В начальный момент стрелка прибора отклонится, показывая некоторое сопротивление, затем это сопротивление начинает расти, а ток через микроамперметр прибора уменьшаться. Таким образом, при подключении конденсатора через сопротивление ток через него некоторое время протекает. Посмотрим, что происходит в программе.

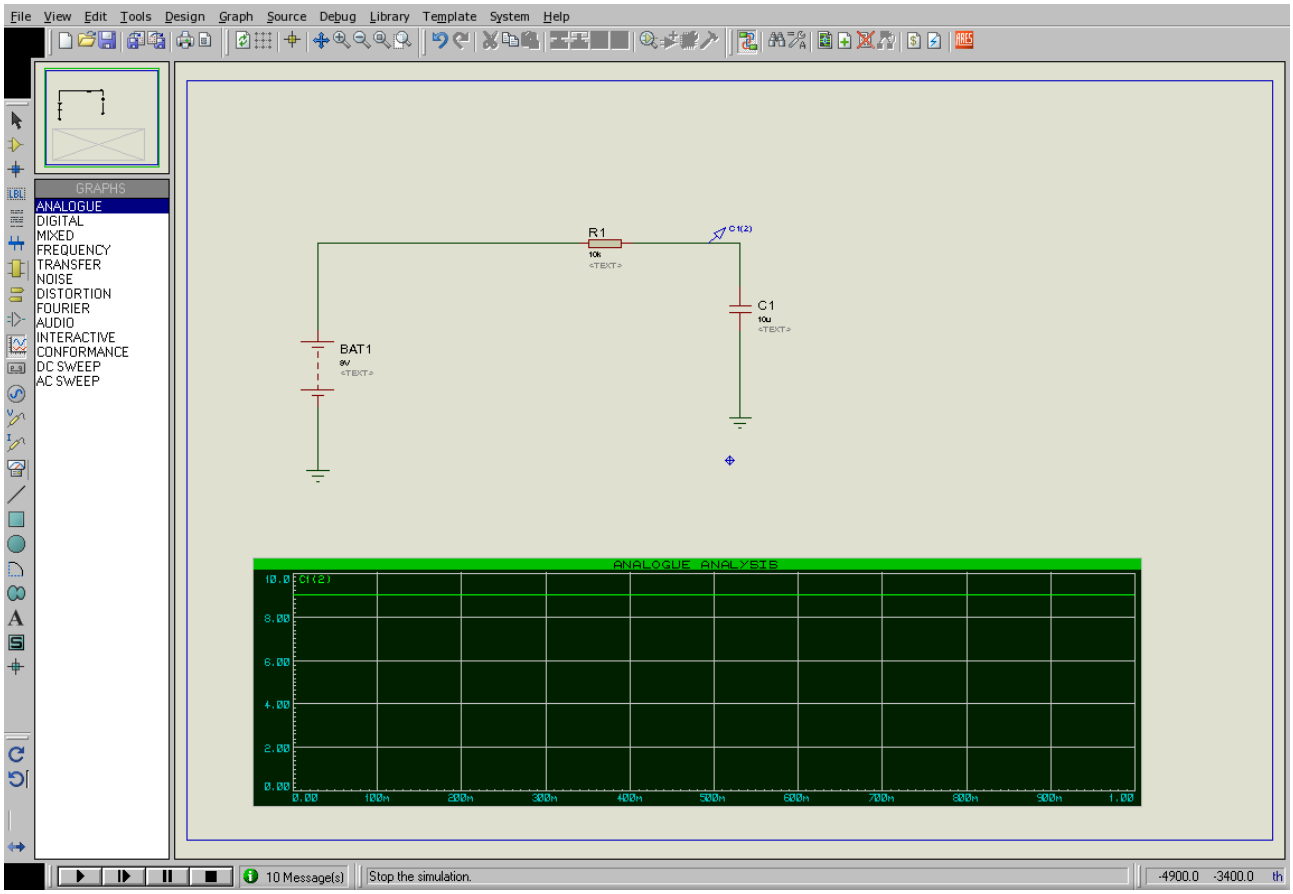


Рис. 7.2. Заряд конденсатора через сопротивление

На графике я не вижу каких либо изменений напряжения на конденсаторе. Напряжение в точности равно напряжению на батарейке 9 В. Означает ли это, что программа на работает или не отображает заряд конденсатора?

Я не учел, когда рисовал схему, что в оригинальной схеме есть выключатель. Поскольку я пользуюсь графическим отображением результата симуляции, я не буду добавлять выключатель, а заменю батарейку генератором ступенчатого напряжения. В Proteus есть генератор PWLINE, который я использую и настрою, как показано ниже.

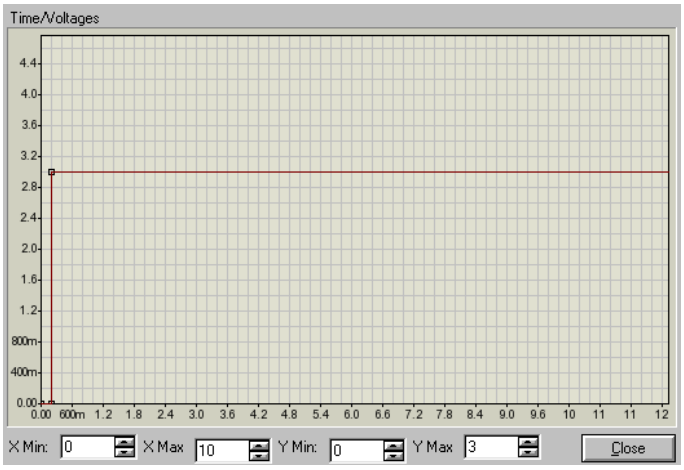


Рис. 7.3. Настройка генератора ступенчатого напряжения

В начальный момент времени напряжение на выходе равно нулю и сохраняется до 200 мС таким, а затем сразу возрастает до 3 В. Это должно соответствовать замыканию контактов выключателя. Посмотрим, что происходит с напряжением на конденсаторе в этом случае.

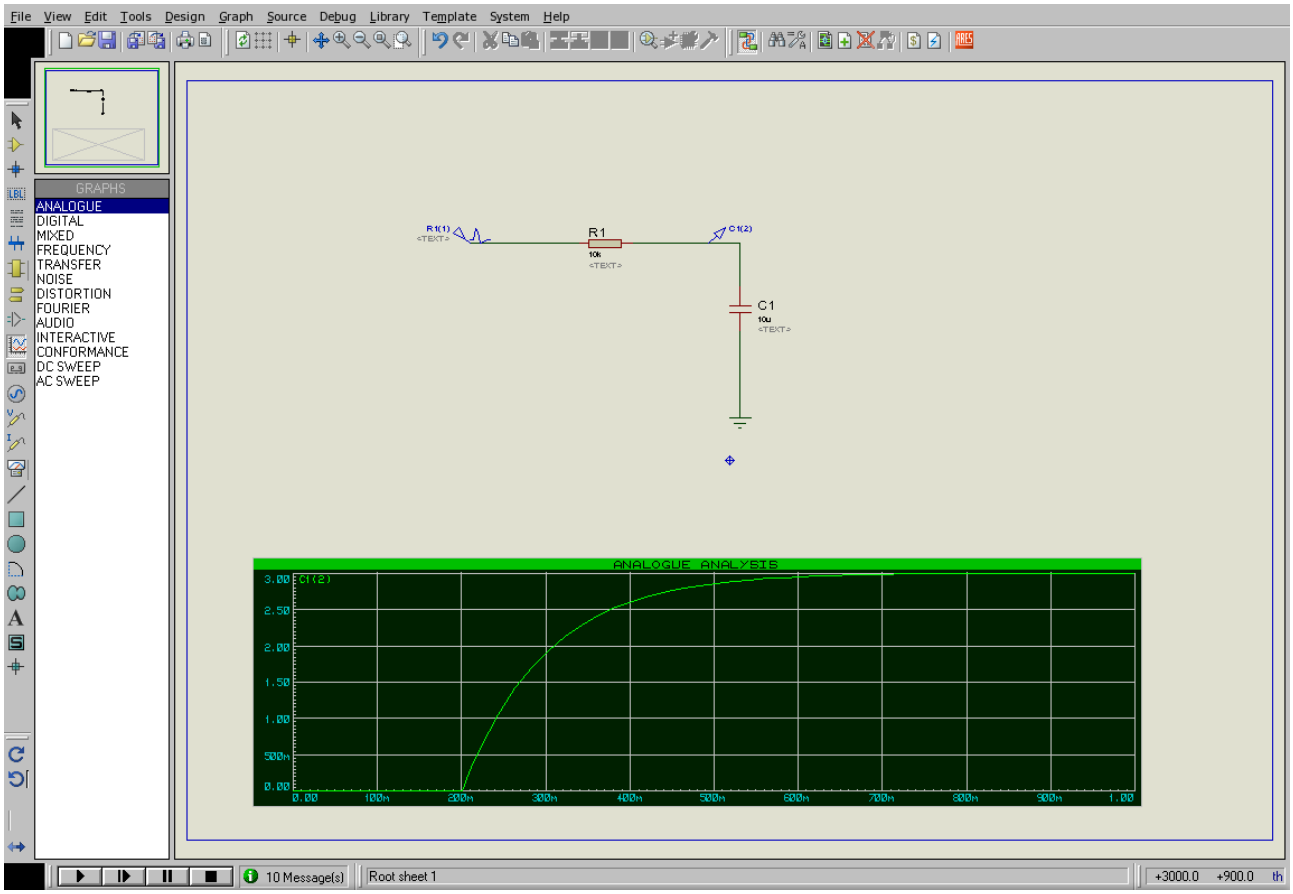


Рис. 7.4. Заряд конденсатора через резистор при использовании генератора «ступеньки»

Теперь конденсатор ведет себя так, как я и представляю заряд конденсатора, напряжение на нем увеличивается по экспоненте.

Но замена батарейки генератором ступенчатого напряжения в основной схеме все-таки не приводит к успеху, как и замена светодиода из набора для анимации на основной тип.

Что можно еще сделать? Можно перейти к макетной плате, но мне интересно понять работу схемы, а понимания я пока не достиг. Итак. Вернемся к моменту, когда напряжение изменяется от нуля к 3 В, напряжению батарейки. Для этого момента времени конденсатор можно заменить резистором малой величины, а идеальный конденсатор даст короткое замыкание между выводом резистора R1, подключенным к базе транзистора Q2 и катодом светодиода, соединенным с коллектором транзистора Q1. В этом случае через базу транзистора Q2 должен протекать ток, открывающий транзистор, что откроет и транзистор Q1, светодиод будет гореть. Обычно на светодиоде, включенном в прямом направлении, падает 1.5-2 вольта (хотя не всегда так).

Произведя необходимые изменения в схеме я не вижу ожидаемых результатов, что вызывает у меня желание изменить тактику и просто проверить все токи в цепях. Удалив графики я расставляю амперметры, используя диалог свойств, меняю их на миллиамперметры и наблюдаю нечто меня удивляющее — токи в цепях соответствуют моим ожиданиям, кроме тока через светодиод, который, похоже, не протекает.

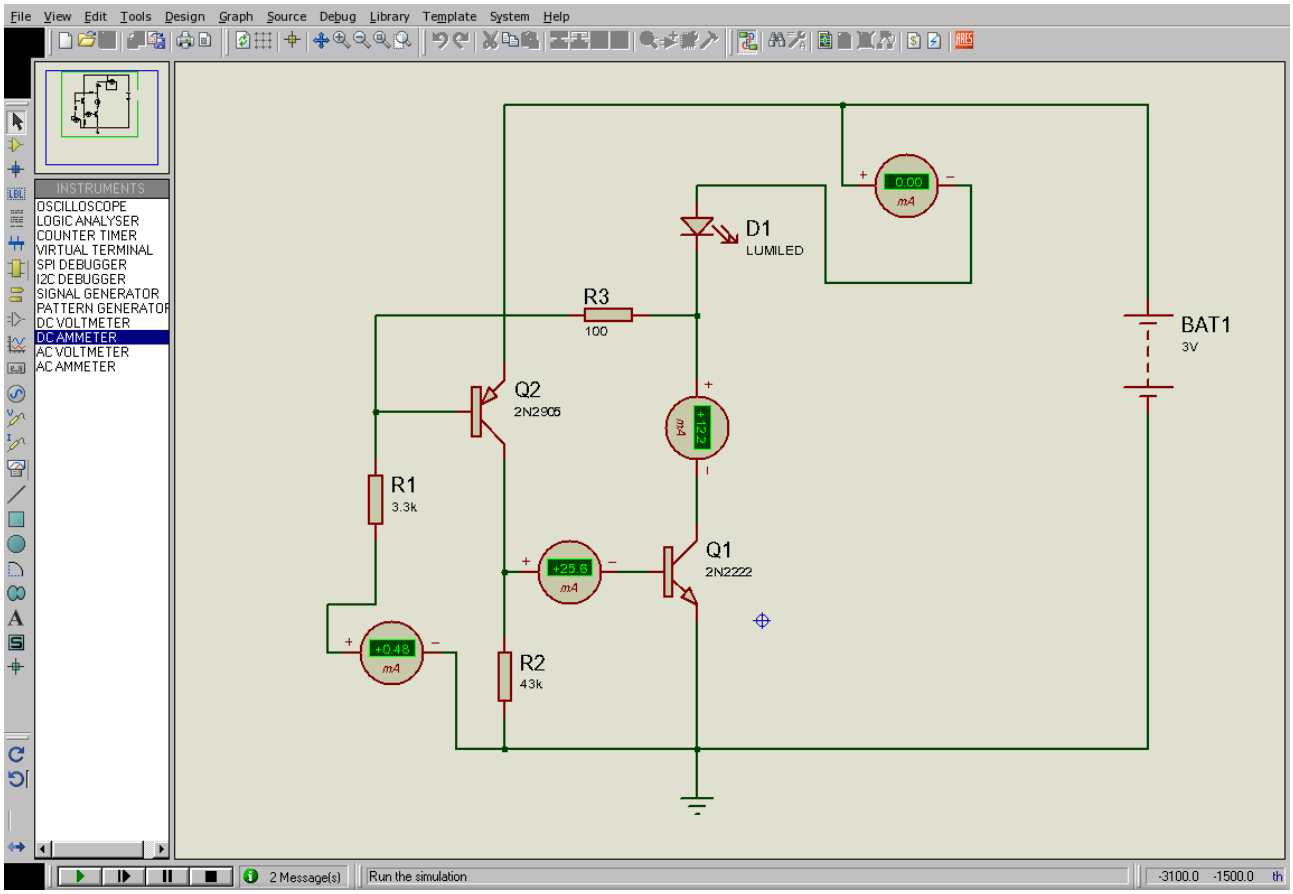


Рис. 7.5. Измерение токов в ветвях схемы

Что ж, не будет большой беды, если светодиод в статическом режиме заменить эквивалентным сопротивлением. Обычный ток через обычный светодиод можно выбрать равным 10 мА при падении напряжения на нем 2 В. Тогда эквивалентное сопротивление получается равным 200 Ом. Меняем. И получаем ток и в этой ветви.

Теперь меня интересует поведение светодиода. Оставляем в схеме только светодиод, вместо батарейки включаем генератор PWLINE со следующими настройками (первоначально я задал максимальное напряжение 3 В, затем изменил его на 10 В).

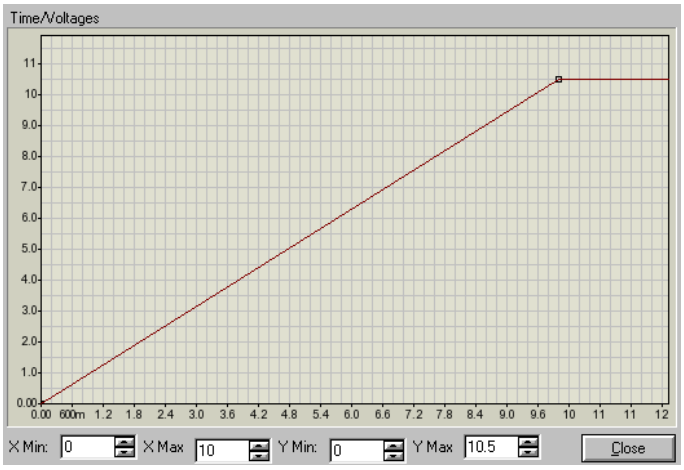


Рис. 7.6. Настройки генератора PWLIN для проверки светодиода

К катоду светодиода я добавлю пробник тока, который с помощью режима вращений (и отображений) поверну так, чтобы направление тока удобно отображалось на графике. График, где я наблюдаю изменение напряжения на светодиоде и тока через него, выглядит так.

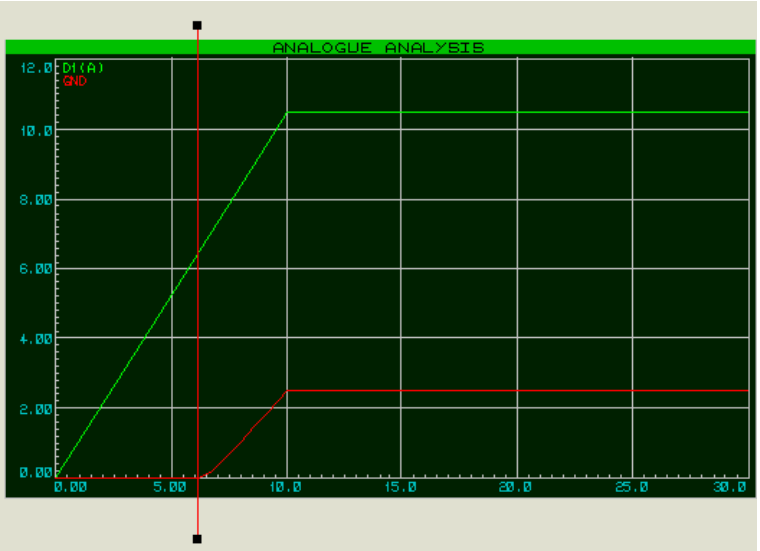


Рис. 7.7. График изменения напряжения и тока через светодиод

Теперь понятна причина неудачи при измерении токов — светодиод открывается при напряжении больше 6 В. Для сравнения я повторяю этот эксперимент с другим светодиодом из компонент Proteus.

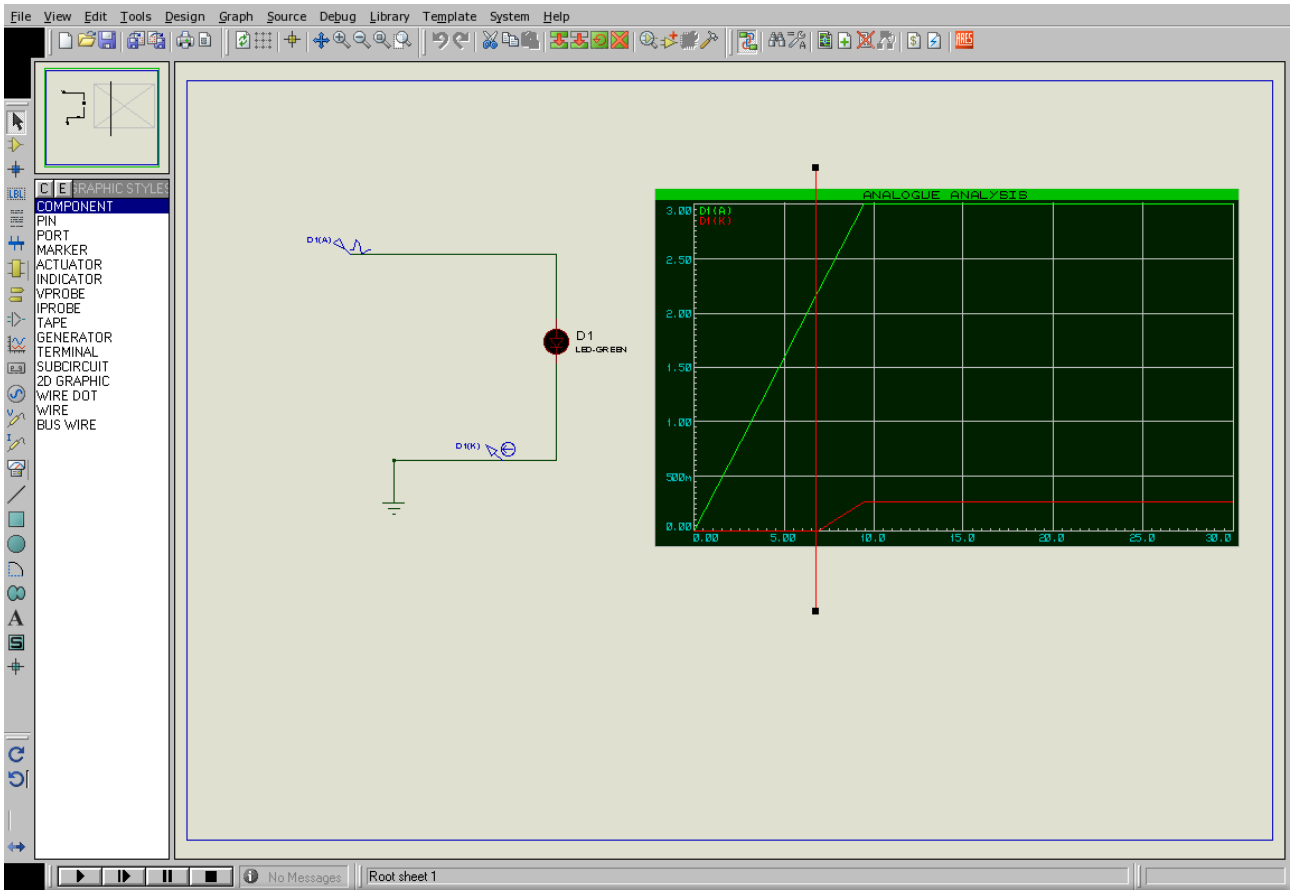


Рис. 7.8. Второй эксперимент с другим светодиодом

Этот светодиод открывается при напряжении порядка 2 В. Если теперь повторить схему рисунка 7.5, заменив светодиод, то ток в цепи светодиода протекает, что и следовало ожидать, если не делать ошибок, особенно, учитывая, что я сам замечал, что не для всех светодиодов падение напряжения на них в рабочем режиме около 2 вольт.

Теперь, когда отдельные вопросы включения и переключения несколько яснее, мне интересно посмотреть на схему в другом ключе, на эту мысль наводит меня график происходящего на резисторе R1.

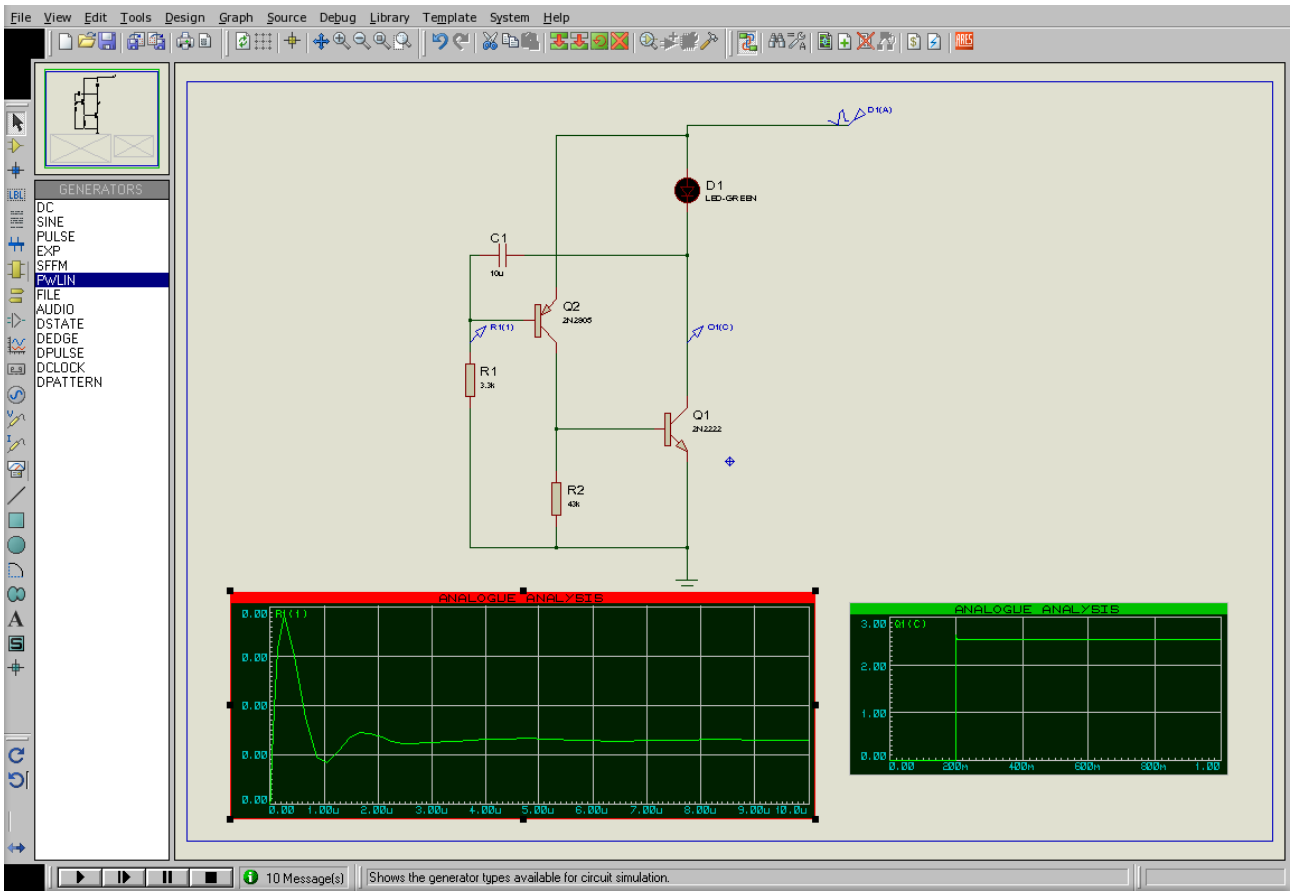


Рис. 7.9. Повторный эксперимент со схемой при увеличении напряжения до 5 В

В сущности схема, если удалить конденсатор C1 — это двухкаскадный усилитель. Манипулируя величиной резистора R1 можно задать рабочую точку такой, чтобы на коллекторе транзистора Q1 было напряжение равное половине питающего напряжения, при этом светодиод лучше заменить эквивалентным резистором, а генератор «ступеньки» на батарейку. На входе усилителя можно поставить генератор синусоидального напряжения с частотой, положим, 10 кГц, и посмотреть сигналы на входе и выходе.

Несколько попыток с измерением постоянного напряжения на выходе и подстройкой напряжения генератора заканчиваются тем, что можно увидеть графики напряжений.

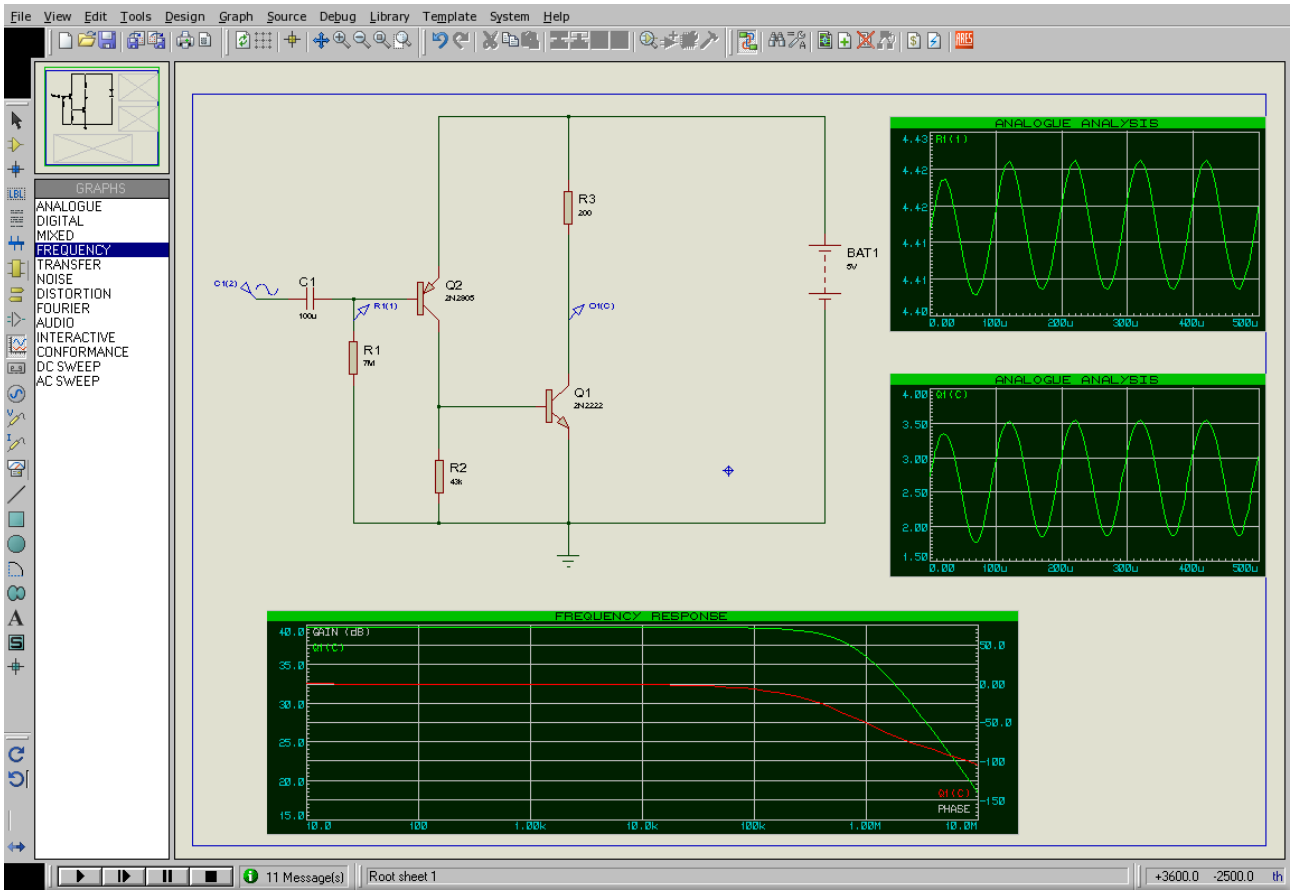


Рис. 7.10. Испытания схемы, как двухкаскадного усилителя

В первую очередь меня интересует соотношение сигналов на входе и на выходе, но не по величине, а по фазе. Сигналы по фазе совпадают. Таким образом, включая конденсатор C1, как на рисунке 7.9, мы вводим положительную обратную связь. Положительная обратная связь должна приводить к автогенерации. Следовательно, хотя бы этот механизм позволяет мне понять, как должна работать схема.

Осталось выяснить, виновата ли в отсутствии симуляции программа Proteus, виновата ли схема, или виноват я, использовав случайные транзисторы вместо рекомендованных. Положительная обратная связь должна приводить к самовозбуждению, но я использовал транзисторы, которые могут иметь очень большой коэффициент усиления, что, в свою очередь, может приводить к тому, что транзистор Q1, работающий в режиме насыщения, входит в столь глубокое насыщение, что срывает генерацию. Попробуем регулировать рабочий режим транзистора Q1, меняя величину резистора R1. Это изменит частоту генерации (при удачном исходе), но с частотой можно разобраться либо позже, либо на макетной плате. Контролировать выбор резистора мне помогает вид сигнала на резисторе R1.

После нескольких попыток появляется сигнал на выходе, который мне представляется достаточно правомерным для этой схемы, исключая частоту повторения импульсов.

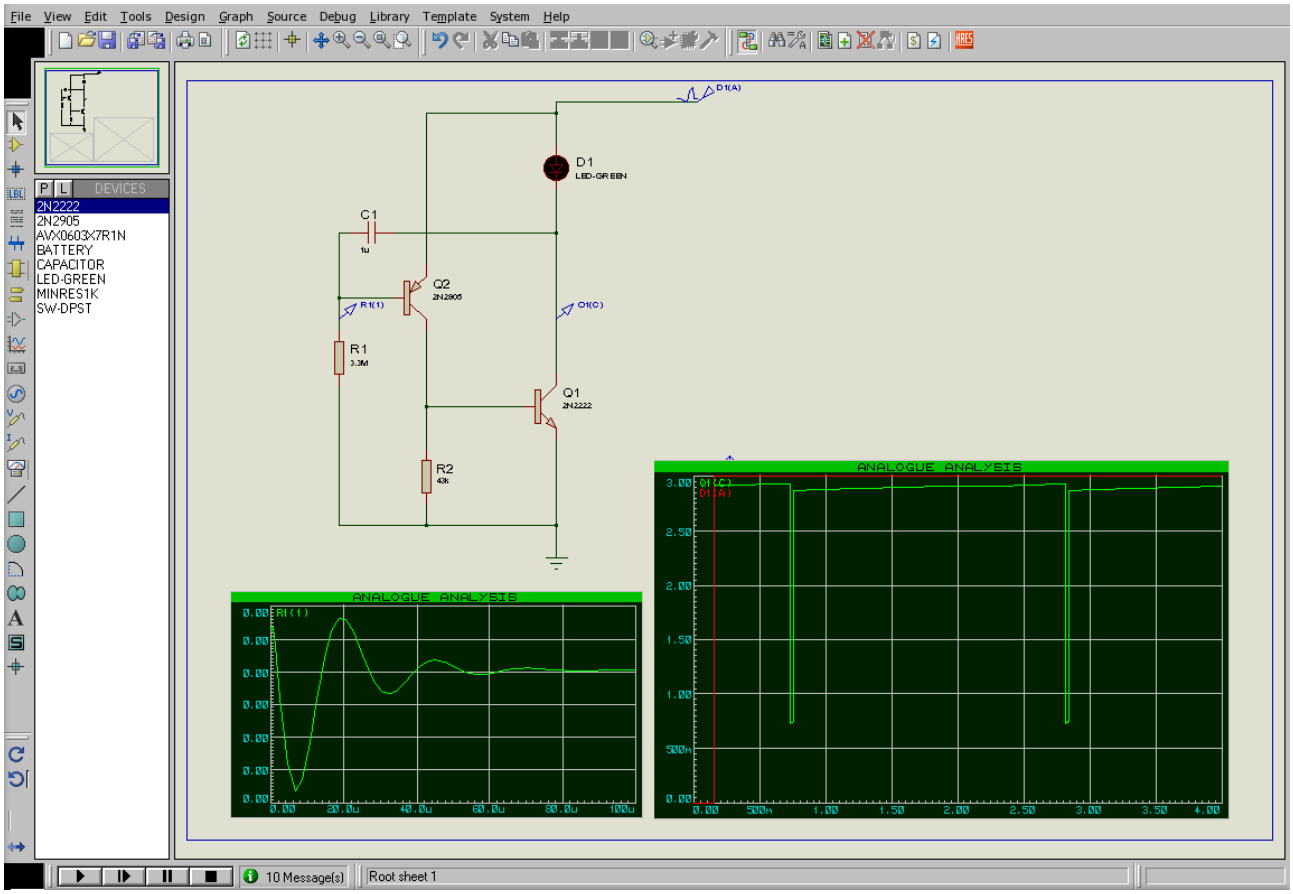


Рис. 7.11. Работа схемы после ее настройки

Подобрать частоту повторения, если это важно, можно изменением величины конденсатора C1.

После проведенных экспериментов работа схемы мне представляется следующей. В начальный момент транзисторы закрыты, пока не завершится заряд конденсатора C1 и транзисторы не откроются. Через полностью открытый транзистор Q1 конденсатор быстро разряжается (собственно, цепь разряда, видимо, переход база-коллектор Q2 и база-эмиттер Q1), что приводит к новому циклу заряда конденсатора при закрытых транзисторах.

В итоге, возвращаясь к началу рассказа, я могу ответить на вопрос, что бы я делал, если бы схема не захотела работать. Я проделал бы все, что проделал, повторил это на макетной плате, постаравшись использовать рекомендованные компоненты схемы: транзисторы и светодиод. А после того, как схема заработает, аккуратно удалил бы активные компоненты и те, что изменились в результате наладки, с печатной платы и перенес бы компоненты с макетной платы. Думаю, устройство заработало бы.

Почему я работаю с AVR?

Такой вопрос я не задаю себе, но часто встречаю в несколько иной форме: «Что лучше, PIC или AVR?». Затрудняюсь сказать что-либо вразумительное, но ответ, с которым я вполне согласен, я нашел на сайте http://avr.nikolaew.org, ссылку на который обнаружил на www.radio-portal.ru в недавно открывшемся разделе форума по микроконтроллерам. С микроконтроллерами Atmel, вернее контроллером 8051, я знакомился слишком давно, но упоминание о Proteus на форумах чаще связывалось с AVR микроконтроллерами, и любопытство взяло верх.

У меня нет таких задач, которые заставили бы меня сравнивать микроконтроллеры разных производителей, а те небольшие опыты, которые я проводил, вполне вписывались в работу с PIC-контроллерами. Вдобавок мои предпочтения, касающиеся Linux, в полной мере укладывались в то, с чем я имел дело в части микроконтроллеров. Как я выяснил, Proteus вполне счастливо работает с PIC-контроллерами, отчего эту главу с чистой совестью я мог бы опустить. Однако отчего бы не познакомиться с AVR-контроллерами? В первую очередь в Linux и Proteus.

В поставках Proteus есть много примеров работы со всеми мыслимыми устройствами, по крайности в любительской практике. Есть примеры работы с AVR-контроллерами.

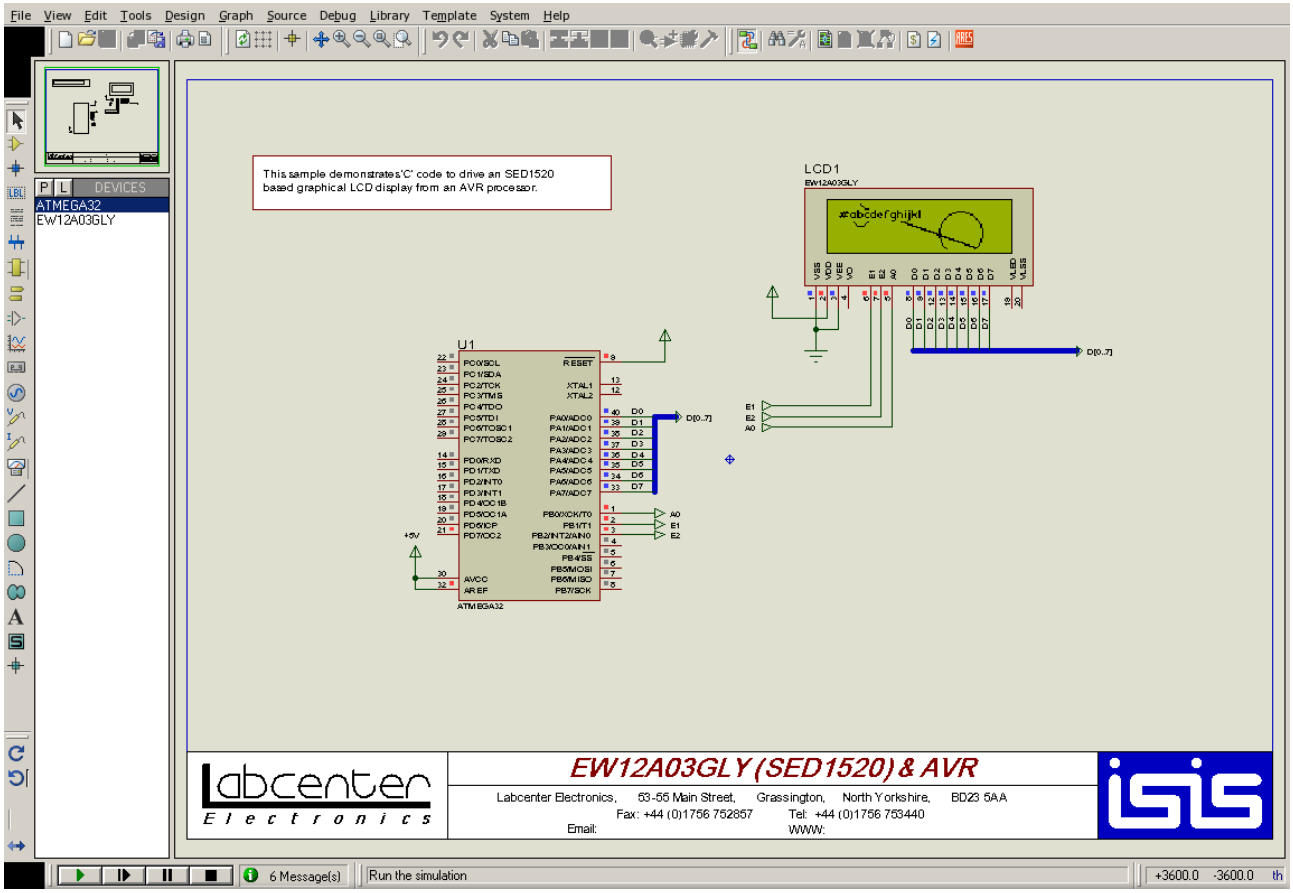


Рис. 8.1. Пример работы AVR-контроллера в Proteus

Предпочитая работать с языком высокого уровня в удобной графической среде, я ищу, что предлагает Linux для работы с AVR. В моем дистрибутиве Fedora 8 есть некоторые средства программирования этих микроконтроллеров. Загружаю я очень быстро, но также быстро убеждаюсь, что это только транслятор и полезные утилиты, но не привычная графическая

среда программирования. Поиск по запросу «avr linux» в Интернете приводит меня к программе **Kontrollerlab**, которая существует в стабильной версии для Fedora Core 5 и в следующей альфа версии для Fedora 6. Начну со стабильной версии. Надеюсь, я загрузил для своего дистрибутива и компилятор «С», и что-то полезное для программатора, хотя не собираюсь пока собирать и использовать программатор.

Первый запуск программы **Kontrollerlab** показывает, что графический интерфейс очень похож на многие интерфейсы программирования в Linux. На всякий случай я создаю новый проект через раздел **Project** основного меню, где есть пункт **New Project**. Обычная процедура указания места расположения проекта и его названия, следом через раздел File и New, где можно выбрать, будет ли это файл на языке С или ассемблерный, я создаю исходный файл. Остается написать код пробной программы.

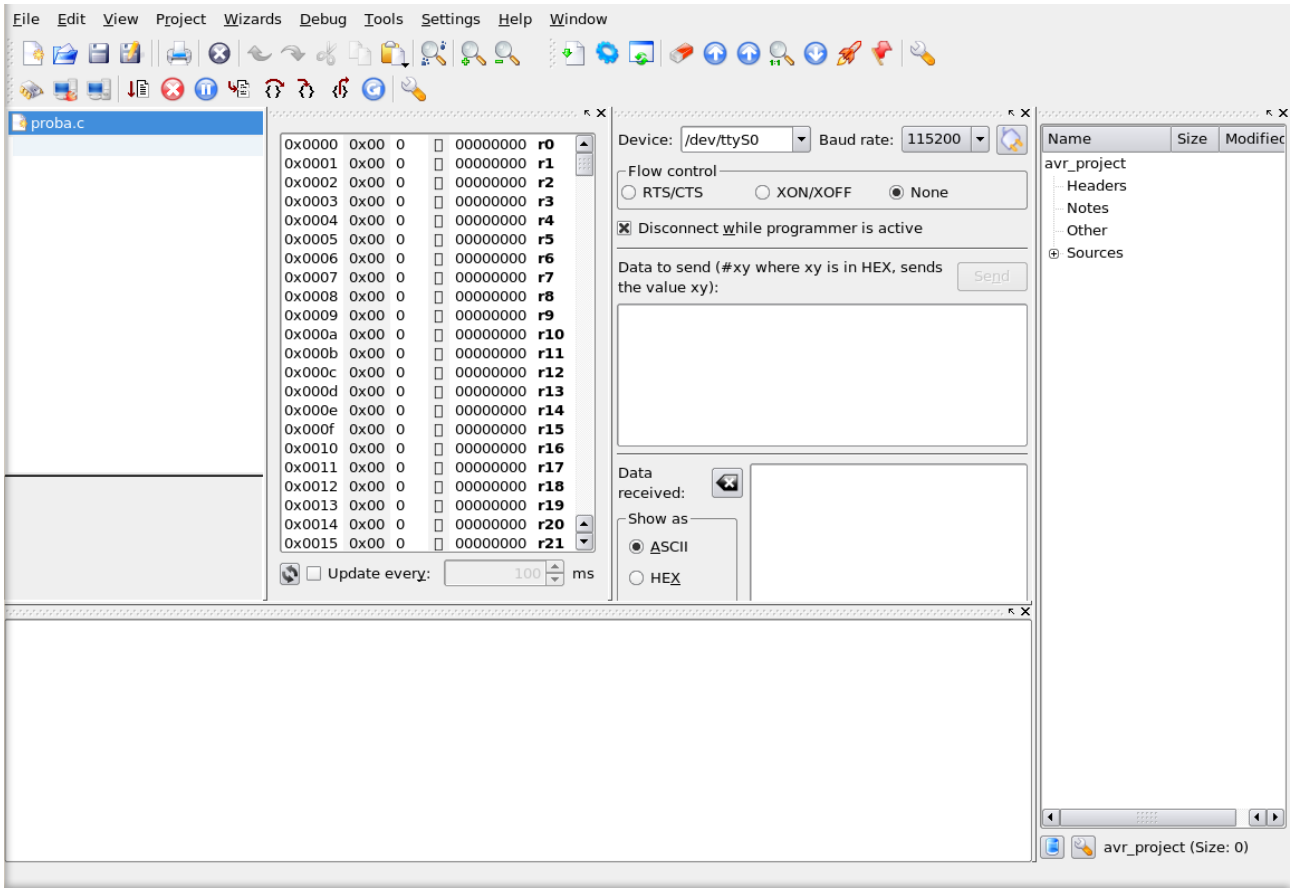


Рис. 8.2. Запуск программы Kontrollerlab

Я не видел программы AVRStudio, но работа много лет назад с программой для программирования и отладки контроллера 8051 оставила очень хорошие впечатления, а за прошедшие годы, я уверен, среда программирования контроллеров Atmel стала только лучше. Но меня интересует среда программирования для Linix. Итак.

Небольшой текст на языке С после нескольких поправок транслируется, полученный hex-файл даже работает в Proteus, но попытки запустить отладку в программе Kontrolltrlab не дают видимого эффекта. Не помогает и установка последней версии, то есть, отладка как-то идет, работают команды пошаговые, что-то происходит с регистрами микроконтроллера, но я бы не назвал это удобным отладочным средством. Хуже всего, что сайте проекта эти разделы документации еще не созданы. Конечно, можно транслировать исходный текст в этой программе, а присоединять дополнительные элементы и симулировать работу в Proteus. Что

можно проверить работу микроконтроллера таким образом, я знал и до этого, стоило ли начинать эксперименты?

Для отладки программ AVR-контроллеров есть штатная программа AVRStudio. Но, хотя установка проходит без видимых осложнений, работать программа в Linux не желает. Выяснять, отчего это так, особого желания нет, достаточно простого ответа — используй программу с той операционной системой, для которой она предназначена. Но есть и еще рекомендуемые бывалыми средства работы с AVR-контроллерами: VMLab и WinAVR. VMLab даже обозначена, как работающая в Linux (под Wine), что радует.

И действительно, программа устанавливается, запускается. Некоторые сомнения одолевают при попытке создать новый проект с помощью раздела основного меню **Project-New Project**.

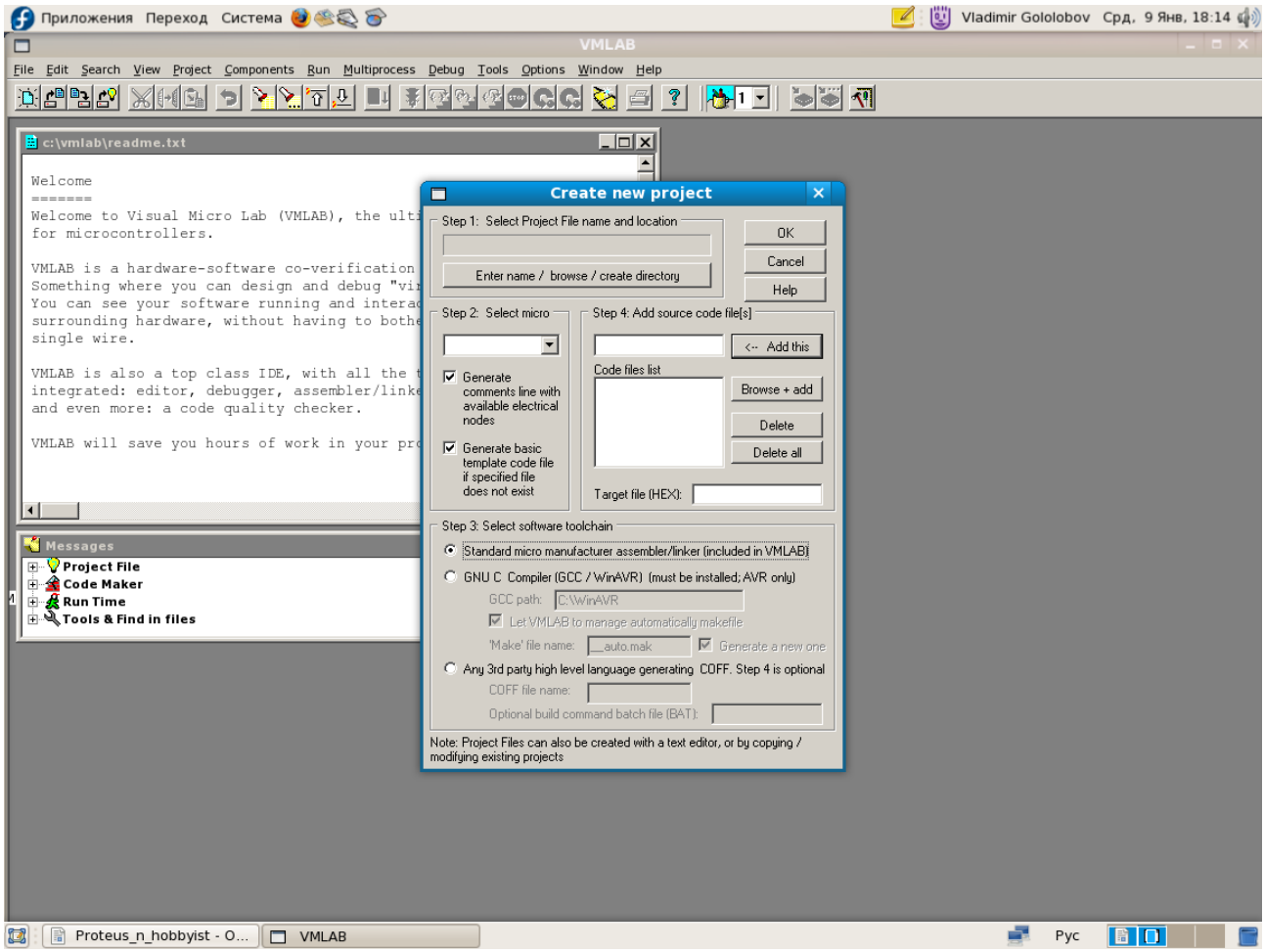


Рис. 8.3. Программа VMLab в Linux

Как всегда есть несколько путей к достижению цели: найти документацию и прочитать ее, обратиться к опыту тех, кто уже работал с программой, можно посмотреть демонстрационные проекты, чтобы попытаться понять, как следует создавать проект. И есть еще один путь — открыть обучающий проект, расположенный в папке *tutorial*. Пожалуй, с этого я начну. Проекты обозначены достаточно ясно *Step01.prj* и т.д.

В окне сообщений «*Messages*» в разделе **Code Maker** появляется предупреждение, что следует запустить пересборку проекта для получения hex-файла. В разделе основного меню **Project** есть пункт **Rebuild-all**.

Мне жаль, но после запуска на пересборку проекта программа «виснет», и я не понимаю, отчего это происходит. Дело явно не в программе, а в ее работе в Linux. Может быть, программа совсем не работает?

Это легко проверить, с программой приходит множество примеров, скажем, в папке *AVR_demo*. Мне приглянулся проект *lcd.prj*, который можно открыть с помощью **Project-Open project** разделов основного меню. Проект открывается, появляется сообщение, что hex-файл обновлен и нет необходимости пересобирать проект (скорее всего, я пытался запустить его раньше), но я не вижу «оживших» клавиш отладки, поэтому запускаю сборку проекта, после которой оживает клавиша запуска с иконкой светофора с зеленым светом. Если нажать на эту клавишу, то в окне дисплея появляется предложение ввести что-нибудь с клавиатуры в окно терминала. Я ввожу слово *proba*.

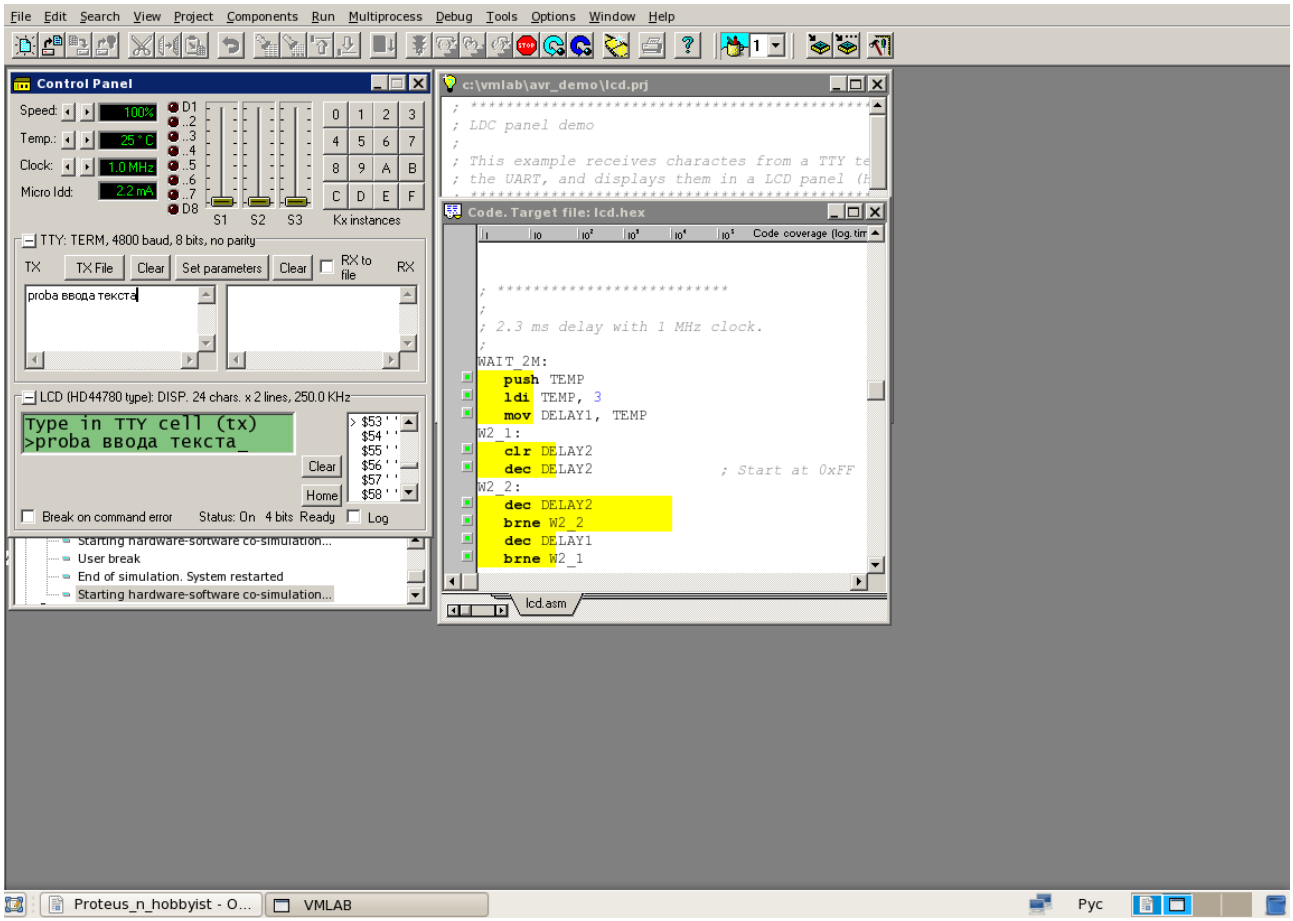


Рис. 8.4. Демонстрационный пример в VMLab

Я не удержался и переключил клавиатуру на ввод русского текста, что не помешало его правильному отображению на дисплее. Если в разделе основного меню **Run** отметить **Animate code**, то включается анимация прохождения кода. Становится интересно, в чем же причина того, что две демонстрационные программы ведут себя столь различно. Пустое любопытство, согласен, в Windows нет проблем, но — интересно.

Первый эксперимент. В месте установки *VMLab* есть папка *Work*. Перенесем туда текст на ассемблере *lcd.asm* (это текст предыдущего проекта) из папки с демонстрационными программами и попробуем создать свой проект. В окне диалога (рис. 8.3) щелчком по клавише **Enter name / browse/ create directory** (это отмечено как Step 1, первый шаг) можно указать директорию и задать имя проекта, скажем, *test*. В программе *Wine*, которая поддерживает работу Windows программ, в диалоговом окне выбора директории есть возможность выбрать либо диск C:, виртуальный логический диск Windows структуры, либо

диск Z:, логический диск реального расположения ОС Linux, если нажать на клавишу с иконкой домика и выбрать «Мой компьютер» в окне выбора. Я выбираю диск C:, нахожу программу *VMLab*, расположенную в корневой папке, и выбираю, наконец, папку *Work*. Проект по умолчанию назван *my_idea.prj*, и достаточно исправить название проекта. Нажав на клавишу **Save**, сохраняем эти настройки и обращаемся к шагу 2, где следует выбрать модель микроконтроллера. Здесь, чтобы не искать ненужных приключений, я подсматриваю в файле *lcd.prj*, исходном файле проекта, который открывается в тестовом редакторе, какова исходная модель — AT90S8515_64K. Такая модель есть, а приставка 64K означает наличие внешней памяти. Возможно это не обязательно. Попробуем выбрать базовую модель. На третьем шаге я не знаю, что выбрать и оставляю настройки «по умолчанию», а на четвертом шаге в окне Code files list мне приходится выбирать *lcd.asm* с помощью клавиши **Browse +add**, иначе из диалога не получается «плавный» выход, но исходный текст *test.asm*, появившийся после задания проекта, в окошке выше исчезает. Пусть так. Клавиша **OK** закрывает окно диалога и в проекте появляется текст проекта, исходный ассемблерный текст и окно сообщений рекомендует пересобрать проект. В пункте **Windows** основного меню есть возможность с помощью **Title** расположить рабочие окна удобным для меня образом. И, если до этой операции подменю **Components** было «серо» (не активно), то теперь оно активизировалось. Я понимаю, что мне нужно добавить терминал и дисплей. Теперь такая возможность появилась (это TTY и LCD module). После этого выбора в файле проекта появляются две строки:

```
  X[inst_name]    TTY(baud_rate  [n_bits]  [parity]  [odd_parity]  [n_stop_bits]
[rx_display_as]) node_tx node_rx
  X[inst_name]  LCD(chars lines oscil_freq) RS RW E D7 D6 D5 D4 D3 D2 D1 D0
```

Конечно, следует задать параметры терминала и дисплея, задать их привязку к контроллеру. Самый простой способ это осуществить, вставить эти строки из готового проекта:

```
  Xterm TTY(4800 8) PD0 PD1
  Xdisp LCD(24 2 250K) PD2  PB0  PD3   PD7 PD6 PD5 PD4   nc3 nc2 nc1 nc0
```

К данному моменту я полагаю, что все необходимое для сборки проекта есть, и пришло время нажать на **Build** в меню **Project** (или использовать соответствующую клавишу инструментального меню). Сборка (с трансляцией исходного текста) проходит гладко, о чем свидетельствует запуск программы.

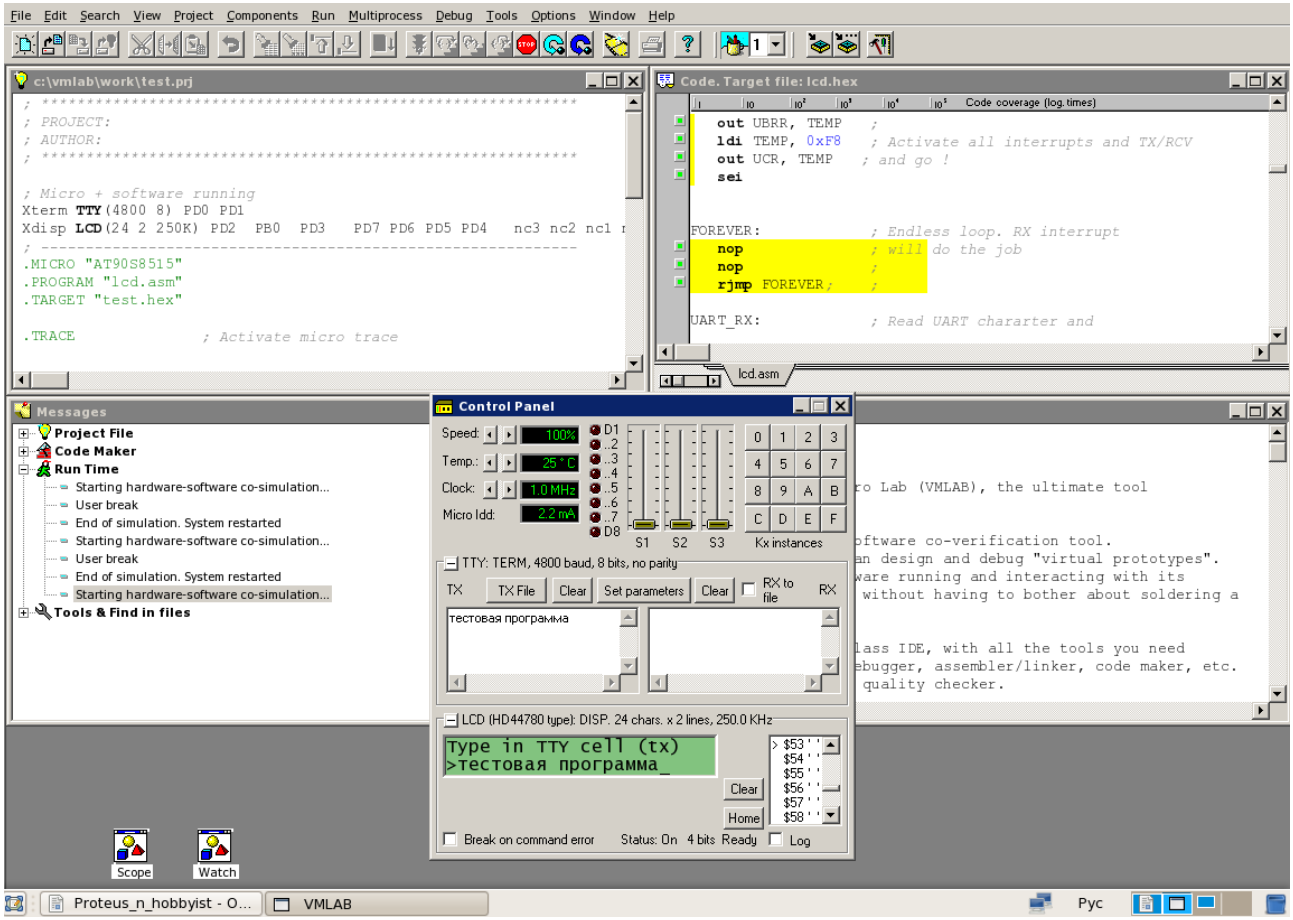


Рис. 8.5. Первый эксперимент по созданию проекта

Проблем не возникает, или я их не вижу, но выход из программы и повторное открывание созданного проекта не вызывают сомнений. Даже пропущенная мною в директивах проекта строка:

```
.plot v(pd0) v(pd1)
```

После ее добавления в проект позволяет наблюдать на экране виртуального осциллографа прохождение сигналов передачи.

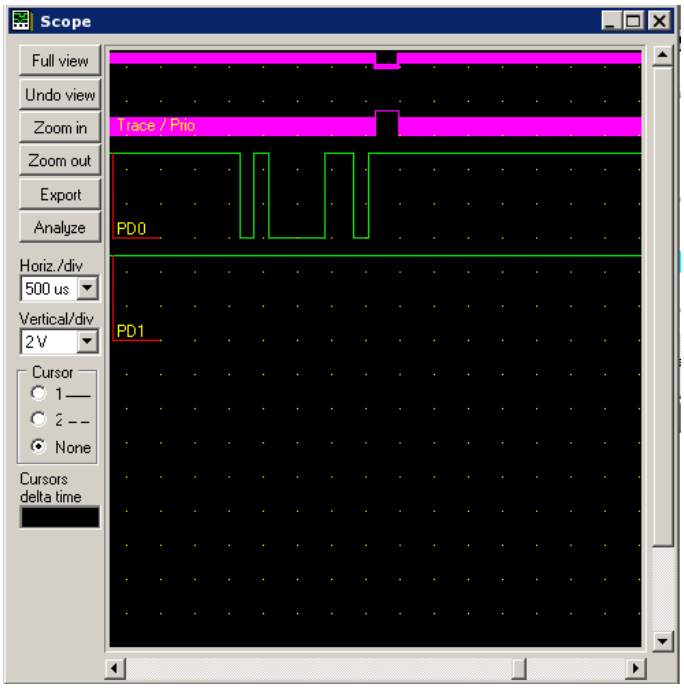


Рис. 8.6. Встроенный осциллограф в VMLab

Второй эксперимент. Опять создадим новый проект, но теперь я не буду использовать готовый файл ассемблера, а в список файлов, нажав на клавишу **<-- Add this** рядом с появившимся файлом *test2.asm*, по названию проекта, добавлю его в список файлов. И выберу модель AT90S8515, с которой все работало. Текст программы я просто скопирую из исходного файла, который открывается текстовым редактором. После сборки проекта программа не виснет, но появляется сообщение о том, что не получается запуск ассемблера и совет проверить, есть ли у меня права на работу с этой папкой. Попытка создать новый пустой проект и оттранслировать его приводит к прежнему результату.

Полная переустановка программы VMLab снимает эту проблему. Теперь пустой проект для модели AT90S8515 транслируется без проблем. Но попытка запустить пустой проект, созданный из шаблона (собственно, в руководстве с проектом Step_01 тоже пустой файл исходного текста) для ST6210 приводит к зависанию программы, и появляется необходимость полной переустановки. Интересно, свойство это моего дистрибутива или свойство Linux запуска программы под Wine? Я могу это проверить в другом дистрибутиве Linux — Ubuntu 7.10...

Нет. Смена дистрибутива не меняет поведения программы. Не любит Linux компонента ST6210. Используя время, проведенное в Ubuntu, я решил посмотреть, что с Proteus в этом дистрибутиве? И не зря. Первая загрузка демонстрационной программы выявила одну особенность — текст примечаний, штампа и все надписи компонент электрической схемы исчезли, как бы их и не было совсем. Немного странно, поскольку другие программы, работающие под Wine, этой проблемы не обнаруживают, иначе можно было бы «грешить» на отсутствие нужных шрифтов. В средствах изменения стиля, где есть настройка шрифтов с демонстрацией образцов, виден только векторный шрифт, и только этот шрифт работает при вводе текста в программе. Потратив минут десять на разные эксперименты, я к своему удивлению обнаруживаю, что если в конфигурации Wine выбрать WinME или Win98, то проблема шрифтов «остается за бортом». Советов в Интернете по этому вопросу я не нахожу, но обнаруживаю обсуждение схемы выполненной на микроконтроллере ATTiny2313. Автор предоставил на обсуждение и схему, и исходный текст, и hex-файл проекта. Я не собираюсь

повторять схему в «железе», но хочу попробовать, как она работает в Proteus.

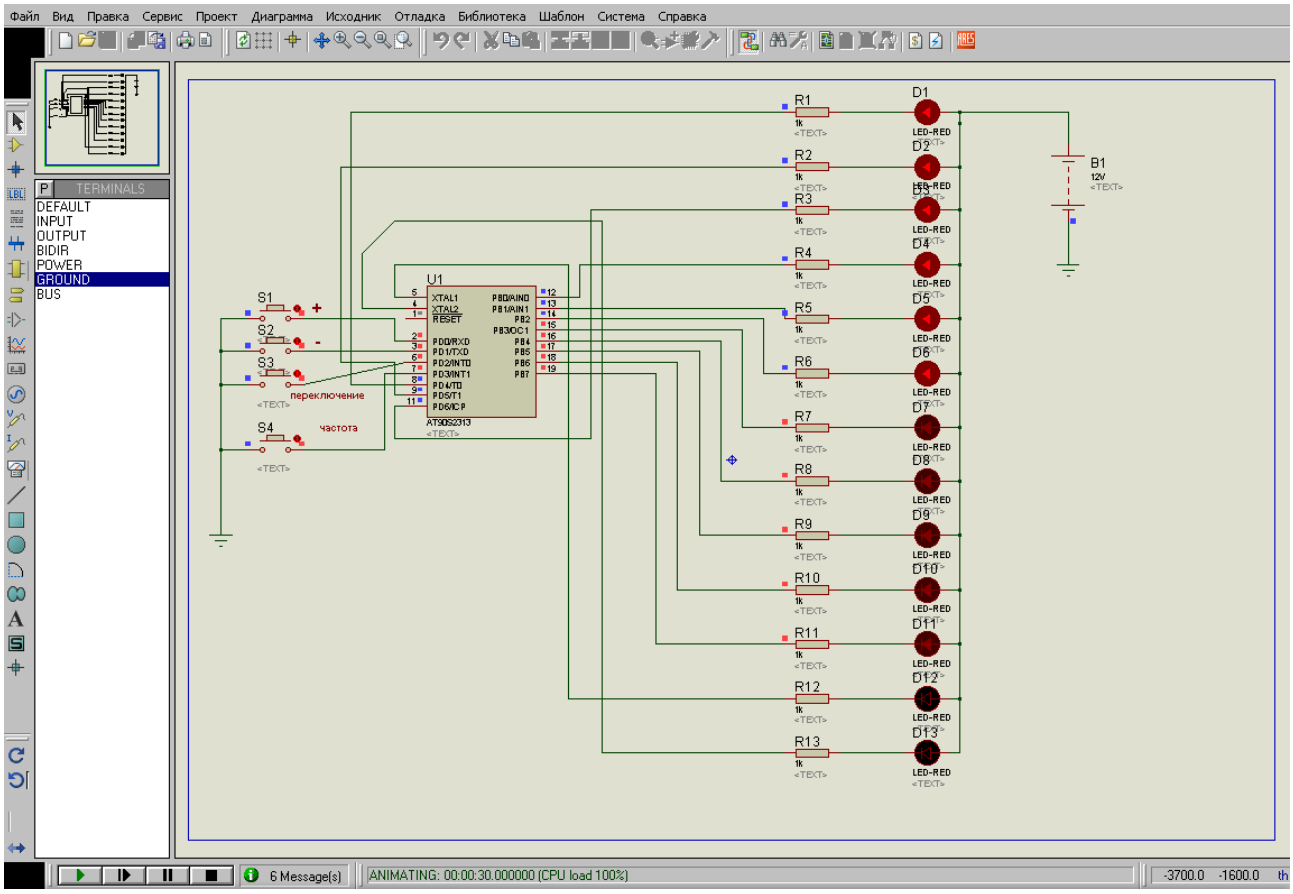


Рис. 8.7. Работа в Proteus схемы с сайта cxem.net

В Proteus я не нахожу нужного контроллера, но использую AT90S2313. Схема работает, реагирует на переключение вида работы гирлянды. В программе не работают светодиоды D12 и D13, несколько медленно происходит переключение. Посмотрим, что говорят на форуме те, кто собирал схему:

> *Я собрал схему, все заработало! Только вот очень медленно работает! Я раз 50 нажимал на увеличение скорости работы программы, но такой скорости, как на видео-ролике у меня не получается! Да, обещанного стабилизированного свечения вроде бы не наблюдалось! Максимум, еле заметное мигание! Что делать?*
>
> *Мигает быстро, а когда на кнопки нажимаешь реагирует примерно через 5 сек (хотя смотря на какую кнопку нажимать). Это нормально? По моему должно быть быстрее, но я не знаю как должно быть.*
>
> *Не понравилась скорость переключения, маленькая. Надо бы побыстрее, так как эффекта бегущей тени или огня при такой скорости не получается, слишком медленно. Можно изменить пределы регулирования? Кроме того нельзя добиться ровного, постоянного свечения светодиодов, они все равно мигают, быстро конечно, но мигают.*
>
> *...сделал как ты сказал! Теперь правда, намного быстрее стало работать, только перестали гореть первые три светодиода, т.е. нет ответа с 4,5,8 ножек МК! В чем может быт проблема?*

Я не знаю, есть ли в Proteus возможность задать слово конфигурации микроконтроллера,

как справедливо отмечают на форуме проблемы могут крыться в этом. Обсуждение на форуме продолжается.

А я хочу завершить беглый обзор возможности работы в Linux со штатными средствами программирования AVR-контроллеров. Если бы мне пришлось начать с ними работу, я попробовал бы сделать это в Linux, несмотря на ограничения, конечно, до тех пор, пока эти ограничения не помешали бы работе. Что же до любительской практики, то, возвращаясь к началу разговора о выборе контроллера, я готов повториться, что если мне понятно, когда есть претензии к организации памяти, прерываний и работе со стеком у профессионала, но любителю, сдается, такие тонкости понадобятся не скоро. Proteus же работает и с PIC-контроллерами и AVR, что я, собственно, и хотел выяснить.