# ЧТО ТАКОЕ «НЕ ВЕЗЁТ» И КАК С НИМ БОРОТЬСЯ, ИЛИ КАК ОБЕСПЕЧИТЬ НАДЁЖНОСТЬ РЭС ПРИ РАЗРАБОТКЕ

**Часть 2**

**Ковалёв Владимир Викторович**
*директор ООО "Новус-Лаб" /"Novus-Lab" Ltd./*

## ***КОНСТРУКЦИОННЫЕ АСПЕКТЫ***

Спектр конструкционных вопросов очень широк. Полностью ответить на них в рамках журнальной статьи практически невозможно. Затрону лишь наиболее «популярные».

«Мостиком» от электрической схемы к конструктиву можно считать печатную плату, которая являет собой некий «сплав» электрической схемы и механической конструкции. Именно печатная плата определяет механическую прочность электронного узла, смонтированного на ней. В ряде случаев она служит одним из несущих элементов конструкции всего устройства. Во многом, именно печатная плата ответственна за ЭМС, терморежимы РЭ. В общем, печатная плата является крайне важным звеном разрабатываемого изделия, и многие конструкционные вопросы надёжности прямо или косвенно с ней связаны.

Одним из определяющих элементов надёжности смонтированной печатной платы служат посадочные места РЭ. Правильно разработанное (с учётом технологии пайки и монтажа, разброса размеров РЭ) посадочное место обеспечит минимум брака при сборке изделия. В своей практике, при проектировании посадочных мест, я предпочитаю руководствоваться стандартом IPC-7351. Неплохим подспорьем в этом является freeware-программа «PCB Matrix LP Viewer» (http://www.pcblibraries.com). В любом случае, этот вопрос желательно согласовывать с конструкторами и технологами сборочного производства, которые дадут необходимые рекомендации по допускам и зазорам (например, для операции автоматического монтажа). Не следует упускать из виду и вопрос грамотной маркировки посадочного места. Необходимая информация в слое маркировки, значительно упростит процесс монтажа, ремонта и обслуживания. Например, информация о контуре РЭ, цоколёвка (маркировка ключевых выводов), позиционное обозначение, поясняющие надписи у разъёмов, контрольных точек и подстроечных элементов.

При разработке посадочного места массивных компонентов не стоит забывать о необходимости их дополнительного крепления. Это значительно повышает надёжность печатного узла при механических перегрузках (удары и вибрации). Нередко разработчики считают дополнительные крепления крупногабаритных деталей необязательными, несмотря на рекомендации производителя РЭ. Мне, например, не раз приходилось видеть буквально высыпавшиеся из плат РЭ в устройствах, работающих в условиях сильных вибрационных нагрузок. В этой связи хочу сказать, что выбор точек закрепления печатной платы также важен. Редко какой инженер делает прочностной (в т.ч. и модальный) анализ, несмотря на то, что современные CAD и FEM пакеты позволяют проводить такой анализ (хотя бы оценочный) даже инженеру средней квалификации. Во многих случаях достаточно даже провести компоновку с учётом возможных ударных и вибронагрузок руководствуясь опытом, прикидочными расчётами и здравым смыслом, хотя конечно, субъективные критерии иногда бывают далеки от объективных. Кстати, для аппаратуры, работающей в условиях значительных механических перегрузок, возможно стоит пересмотреть элементную базу, отдав предпочтение, например, РЭ с лучшими массогабаритными показателями (обладающие естественно и меньшими моментами инерции, что благоприятно скажется на вибростойкости электронного узла).

С процессом проектирования печатной платы, неразрывно связан и вопрос компоновки, связанный также с дизайном изделия. Часто можно встретить такие огрехи компоновки, как неудачное расположение РЭ, теплоотводов, подсоединительных терминалов и интерфейсных

разъёмов, затрудняющие монтажные или ремонтные работы, нарушающие технологические зазоры (например, для высоковольтных цепей), создающие неблагоприятный терморежим уже собранного в корпусе устройства и т.д.

Особое внимание следует уделить вопросам обеспечения нормальных терморежимов РЭ в смонтированном устройстве. Оптимальное размещение элементов на печатной плате позволяет избежать многих неприятных эффектов, например, градиента температур (особенно важно для прецизионных узлов). В приведённом на рис. 10 примере показано как градиент температур, вызванный неудачной компоновкой может привести к ухудшения характеристик схемы. Разница между элементами R3 и R4 достигает 20°C, что для резисторов общего назначения значит уход сопротивлений от номинала на единицы процентов.

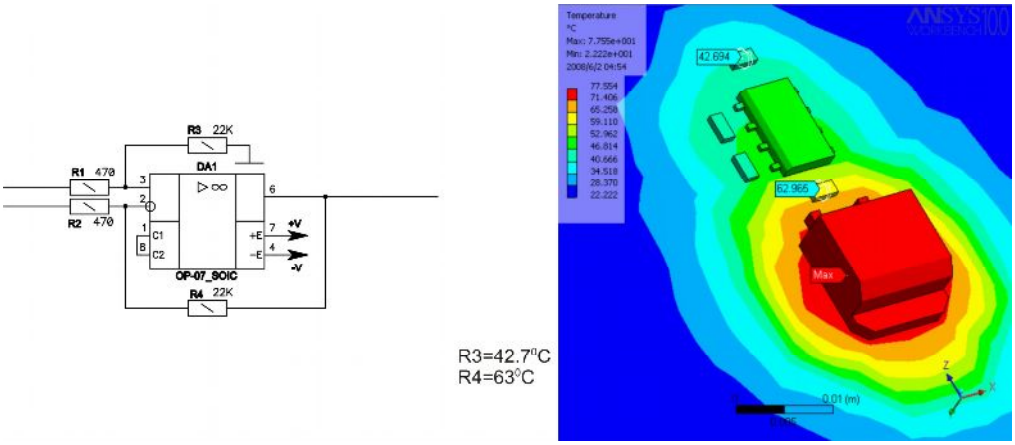


Рисунок 10

На рис. 11 показан эффект «локального перегрева», когда близко расположенные на плате тепловыделяющие элементы «подогревая друг друга», обеспечивают локальную концентрацию большой тепловой мощности при неудовлетворительном её отводе. Причём для выхода из данной ситуации достаточно обеспечить более равномерное распределение тепла, например «растащив» на большие расстояния тепловыделяющие элементы.

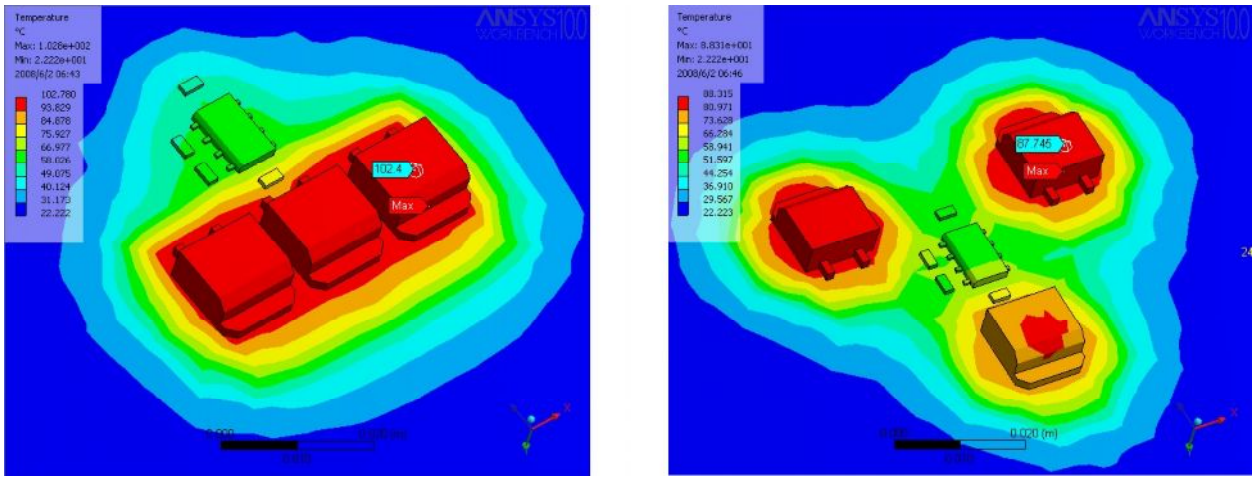


Рисунок 11

Показанный на рис. 12 «тепличный» эффект гарантирует перегрев зоны «внутри» радиатора за счёт поглощения теплового излучения, расположенными там РЭ, да еще и за счёт изоляции конвекционных потоков.

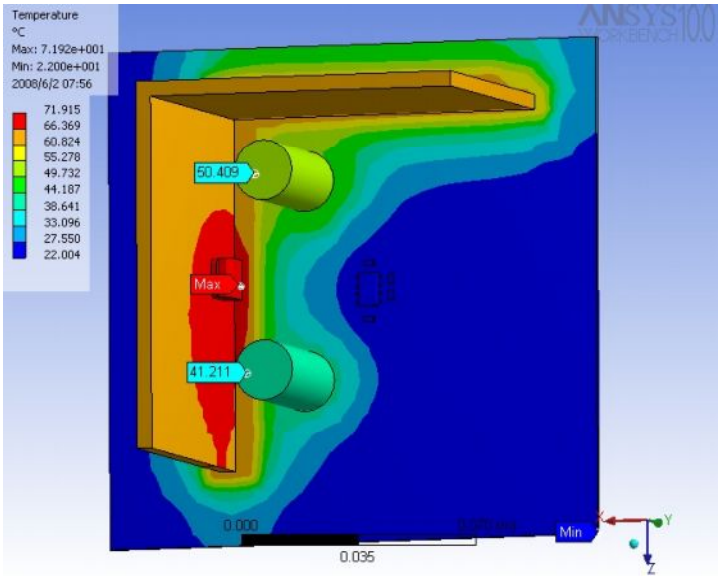


Рисунок 12

Можно привести ещё десятки неудачных, с точки зрения обеспечения терморежимов, примеров компоновок. Это показывает лишь, насколько индивидуальным может быть каждое конструкторское решение, и что универсальных рекомендаций тут не существует. Нужно лишь помнить о возможных «граблях» и внимательно относиться к вопросу обеспечения терморежимов конструкции.

Не менее важным, чем обеспечение терморежимов, является и вопрос электромагнитной совместимости (ЭМС). Здесь как нельзя лучше подходит выражение, что «*профилактика лучше лечения*». Лучше не бороться с помехами, а не создавать благоприятных условий для их возникновения. Наилучшим способом борьбы с импульсными и ВЧ помехами, является локализация цепей, генерирующих помеху. Как правило, это цепи, где протекают импульсные и ВЧ токи значительной величины («значительный» – понятие сугубо субъективное и индивидуальное для каждой конкретной разработки). Хорошим методом является минимизация трасс на печатной плате и соединительных проводов, для цепей с протекающими импульсными и ВЧ токами, а так же шунтирование емкостями по питанию тех узлов схемы, которые потребляют большие импульсные мощности. Если же источник помех локализовать не удалось, то нужно защищать подверженные помехам цепи и узлы схемы. Касательно слаботочных и прецизионных цепей есть ряд методов, описанных в литературе. Ещё одним действенным методом защиты от помех является экранирование. Экранировать можно как источник помех, так и подверженные влиянию помех чувствительные узлы схемы. Экранирование может осуществляться как экранами (электромагнитными и электростатическими), так и элементами конструкции, а также полигонами и специальными слоями на печатной плате.

Следующим важным моментом, особенно для сильноточных, импульсных, прецизионных схем, и в самом тяжёлом случае их сочетанием, является разводка «питания» и «земли». Часто разводке этих цепей не уделяется должного внимания, в результате резко снижается надёжность схем, повышается вероятность сбоев в работе, ухудшаются характеристики, вплоть до полной неработоспособности удачно работающего «в макете» изделия.

Для обеспечения разводки цепей «земли» и «питания» необходимо выделить функциональные блоки схемы, потребляющие большие токи, а также блоки, чувствительные к помехам по питанию. Выделив такие участки схемы, необходимо развести питающие и «земляные» цепи таким образом, чтобы получились независимые контуры протекания питающих

токов для каждого блока. При этом «земли» всех блоков, а также общего источника питания, должны быть эквипотенциальны, причём как для постоянного тока, так и для ВЧ (для импульсных схем можно ориентироваться на частоту 3-й, а лучше 5-й гармоники самого короткого импульсного сигнала). Конструктивно, эквипотенциальность обычно достигается за счёт разводки «земли» трассами наименьшей длины и наибольшего возможного сечения. Для примера рассмотрим топологию, показанную на рис. 13.

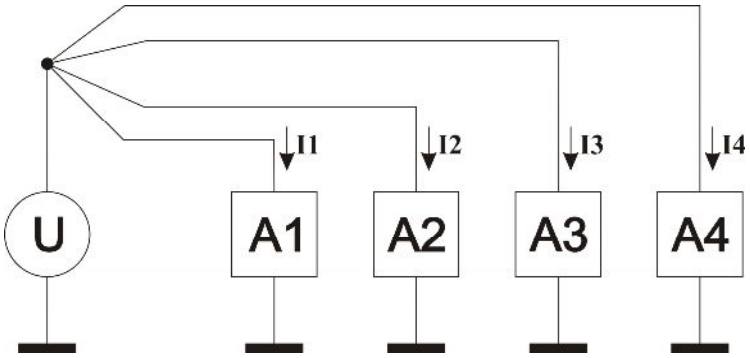


Рисунок 13

Если «земли» всех блоков A1…A4 и источника питания U, с пренебрежимо малым внутренним сопротивлением, эквипотенциальны, то токи I1…I4 в контурах питания, никоим образом не будут влиять друг на друга. На практике реализация данной топологии может быть затруднительна, в частности, проблематично обеспечить эквипотенциальность «земель» всех блоков. Как правило, реальная схема выглядит примерно так, как показано на рис. 14, т.е. присутствует некая «земляная шина» с ненулевым (возможно даже довольно большим) импедансом. Задача разработчика – свести её импеданс к минимуму.

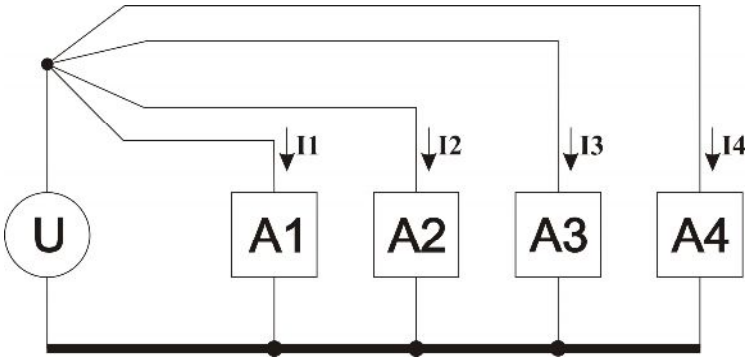


Рисунок 14

Кроме снижения индуктивности и омического сопротивления проводников и печатных трасс, посредством вариации их «геометрии», довольно эффективным известным средством снижения импеданса по ВЧ является применение шунтирующих конденсаторов по питанию, которые позволяют нивелировать эти паразитные параметры. Исключив протекание импульсных токов по трассам питания и «земли», мы тем самым не допустим работы этих проводников в качестве антенн, передающих помехи. Достигается это, как я сказал выше, с помощью грамотной топологии и применения шунтирующих (в данном случае можно сказать накопительных) емкостей по цепям питания. Термин «накопительный» уместнее применять для емкостей, стоящих в непосредственной близости от узлов, потребляющих большие импульсные токи, при этом питание узла происходит именно за счёт энергии накопленной в конденсаторе, исключая, таким образом, протекание импульсных токов по длинным подводящим трассам питания и «земли».

Если блоки связаны аналоговыми сигналами относительно общей «земли», то вариант с эквипотенциальными «землями» из возможных в реализации, является единственно простым. Как альтернативный вариант – возможно построение схемы с гальванической развязкой по

управляющим сигналам, но для аналоговых систем это ведёт к серьёзному усложнению схемотехники. Для цифровых систем, в большинстве случаев, гальваноразвязка делается гораздо проще.

Если блоки независимы, или зависимы некоторые из них, то задача значительно упрощается. В этом случае можно, как показано на рис. 15, сделать эквипотенциальными только «земли» связанных (зависимых) блоков А2 и А3, либо разделить контуры питания, снабдив каждый блок шунтирующей (накопительной) ёмкостью, а зависимые блоки соединить гальванически развязанным каналом, как, например, показано на рис. 16.

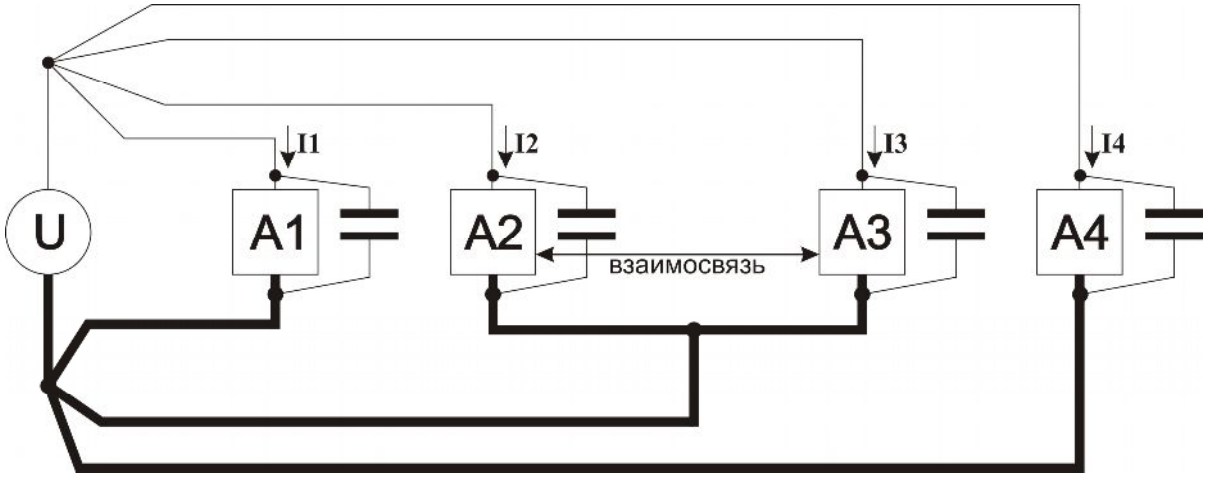


Рисунок 15

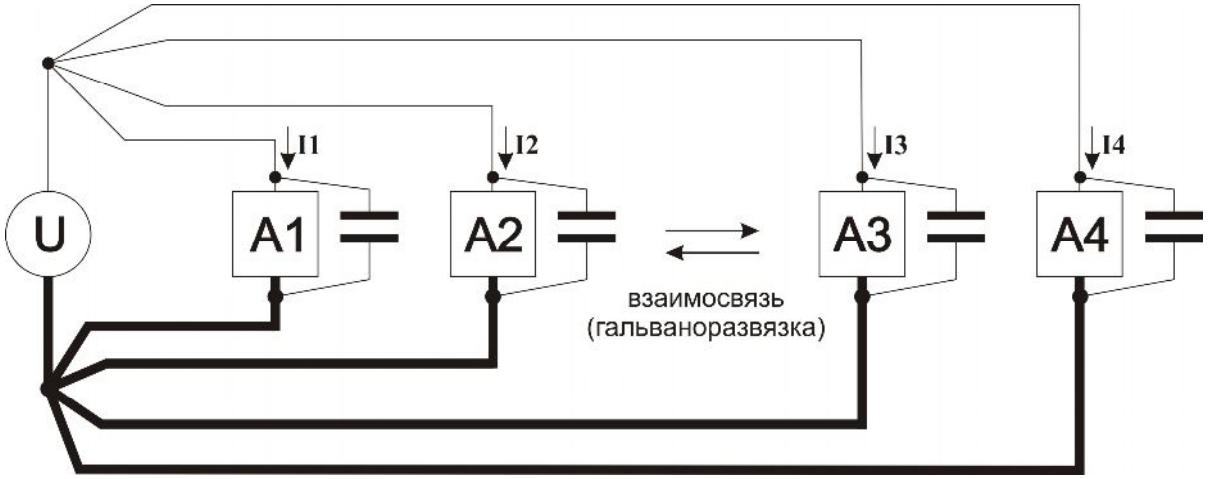


Рисунок 16

Другой путь решения проблемы развязки по питанию – применение ФНЧ, как показано на рис. 17, более сложен в реализации, но иногда и более эффективен, позволяя в некоторых случаях упростить разводку питания (чаще применяется в ВЧ схемах), правда в этом случае также встаёт проблема обеспечения эквипотенциальности «земель». На практике часто приходится применять оба метода (истина, как обычно лежит где-то посередине).

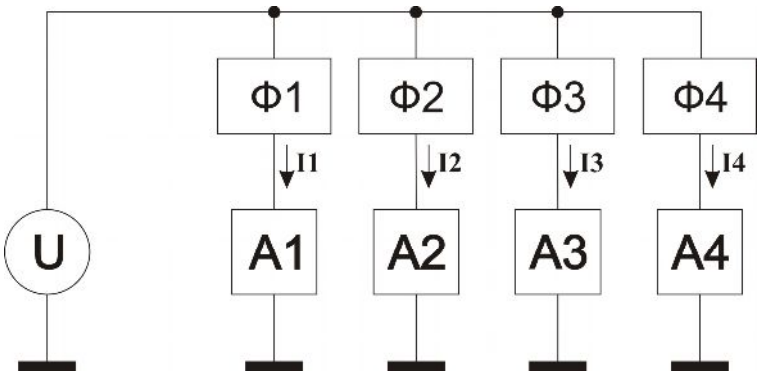


Рисунок 17

Вообще говоря, практически все меры по улучшению ЭМС направлены на минимизацию длин трасс протекания импульсных и ВЧ токов. Можно также сказать, что оптимальным решением будет локализация этих токов в контурах минимальных размеров, образованными компонентами блока и элементами монтажа.

Резюмируя вышесказанное, можно посоветовать делать хотя бы приблизительные инженерные расчеты. Посчитайте максимальные возможные перегрузки элементов, проанализируйте компоновку печатной платы с точки зрения ЭМС и обеспечения терморежимов силовых РЭ и прецизионных узлов. Проверьте разводку «земли», силовых, прецизионных и высокоимпедансных цепей. Необходимо оценить подверженность прецизионных и высокоимпедансных цепей наведённой помехе. Прецизионными считаем цепи, величина напряжений или токов в которых, на шесть и более порядков меньше их абсолютных величин в данной схеме. Высокоимпедансными считаем цепи с модулем полного сопротивления десятки килоом и выше (как правило, это входы аналоговых каскадов, ОУ, АЦП и т.п.). Нужно проанализировать схему на стойкость к кондуктивным помехам по входам/выходам и другим цепям внешних связей. Также нелишним будет проверить стойкость изделия к ударам и вибрациям, в случае сомнений, провести хотя бы прикидочный статический и модальный анализ, естественно, если есть соответствующие требования ТЗ. Не стоит так же упускать из виду вопросы, связанные с монтажом (верификация посадочных мест, маркировка печатной платы, закрепление массивных РЭ), а так же вопросы связывающие компоновку изделия и дизайн.

### ***ПРОГРАММНЫЕ ПРОБЛЕМЫ***

Ввиду того, что программируемых компонентов превеликое множество, невозможно дать универсальные рекомендации, подходящие во всех случаях и ко всем программируемым компонентам, зато можно организовать свою работу таким образом, чтобы максимально упростить отладку, поиск ошибок и модификацию программ в будущем. Лично я не помню ни одного случая, чтобы написанная однажды программа никогда бы не улучшалась и не модифицировалась, ведь как известно (из тех же незабвенных законов Мерфи) «*любая полезная программа содержит ошибки*».

Не претендуя на истину в последней инстанции, «как нужно работать программисту», приведу ниже правила, не раз помогавшие мне эффективно решать и предотвращать проблемы, возникающие при разработке, отладке и модификации ПО.

В программировании существует такое мнение – «лучший комментарий к программе на малоизвестном языке – программа на известном языке». Так вот – язык, которым в совершенстве владеет каждый из нас – это естественный язык человеческого общения. **Поэтому первое и наиглавнейшее правило при программировании – всегда не лениться комментировать исходные тексты.** Причём никогда не откладывать это «на потом», ибо многие идеи

воплощённые в коде, как идеи, очень быстро забываются, остаётся голый код, и понять потом суть программы, бывает очень непросто. Таким образом, комментарий должен быть неотъемлемой частью программы и писаться должен одновременно с ней. Чтобы написание комментария не воспринималось как ненужная рутина, выскажу пару мыслей по поводу того, как я это обычно делаю.

1) Всегда необходимо в заголовке программы (процедуры или функции) поставить блочный комментарий и детально описать суть её работы, синтаксис (входные/выходные параметры), правила применения. Словесно описать общий алгоритм работы, именно общий, не вдаваясь в подробности, разве что указав некоторые ключевые моменты. А также всегда указывать версию и дату последней модификации.

2) Снабжать исходный текст строчными и блочными комментариями. Причём комментарий не должен повторять исходный текст – он должен пояснять его. Рассмотрим примитивный пример:

```
mov   A,B   ; копируем регистр B в A
rol   A     ; сдвигаем A влево на 1 разряд
add   A,B   ; складываем A и B
```

Такой комментарий абсолютно бесполезен, т.к. его информационная ценность равна нулю, он просто дублирует машинную программу, причём на самом низком уровне. Я бы даже сказал, что комментарий – это программа на идеальном языке программирования высокого уровня – естественном. Он должен пояснять суть программы, её алгоритм, дополнять исходный текст.

```
; -- Увеличим время работы XXX втрое.
mov   A,B   ; В регистре «B» исходное время
rol   A     ;
add   B,A   ; Теперь в регистре «B» утроенное время.
```

Согласитесь, что такой комментарий (даже вырванный из общего контекста программы) кое-что значит. Он совсем не дублирует математические действия, определяемые операторами машинного языка, он поясняет программу на более высоком, алгоритмическом уровне. Не всегда необходимо писать подробный строчный комментарий, иногда комментарий может пояснять целый блок кода, но краткие строчные комментарии, как правило, бывают полезны.

Очень хороший способ проверить правильно ли всё прокомментировано – это попробовать «ухватить» суть программы только по одним комментариям.

Следующий «кит» на которого опирается «искусство программирования» это несомненно блок-схема. Можно долго спорить: нужна ли она или нет. Моё твердое убеждение: блок схема абсолютно необходима. Бытует мнение, что блок-схемы, называемые также структурными и функциональными (см. аналогию в разделе «Схемотехнические аспекты»), бесполезны для современных систем и кросс-средств разработки, где используются объектно-ориентированные языки программирования, всевозможные шаблоны и «визарды». Сами же встроенные системы имеют многозадачные операционные среды «на борту», самые модные «фишки», вплоть до Java-машин, TCP/IP и прочих «причиндалов «больших братьев» (ПК). И что-де блок-схема – жуткий атавизм времён Фортрана-IV.

Я не согласен с таким мнением. С помощью блок-схемы возможно описание практически любой программы, в том числе и многопоточных программ и программ для систем с прерываниями. Да, алгоритмические описания стали более сложными (даже визуально они всё больше отходят от классических алгоритмов), сложнее стало и взаимодействие с операционной средой, и это тем более веский довод в их пользу. К тому же работа на уровне алгоритма – ключ к созданию эффективного и надёжного ПО. До сих пор не угасают споры (уже не один десяток лет!) о преимуществах и недостатках разных языков и методов программирования, но все равно неизменной остаётся одна истина: лучшая оптимизация – алгоритмическая. Т.е. косвенно подтверждается почти забытый факт, что любая программа, на любом языке, начинается не с

объявления переменных, функций и определений, а именно с алгоритма. По моему глубокому убеждению, настоящий программист (не путать с «кодером»), реализуя свои идеи, оперирует именно алгоритмами, и среда, в которой он сегодня работает, – это всего лишь инструмент (именно на сегодня). Если разработчик серьёзно собирается заниматься soft-поддержкой своего изделия, то наличие блок-схемы значительно упростит процесс модификации ПО, не говоря уже про поиск и исправление ошибок. К тому же вопрос трансляции встроенного ПО на другие платформы не такой простой как может показаться. Большинство встроенных систем по-своему уникальны и зачастую их ПО жёстко привязано к аппаратным ресурсам. Действительно сложную программу просто перенести на другую платформу только если она функционирует под управлением какой-нибудь известной стандартной ОС. А если это «вещь в себе», то неизбежно возникают проблемы переносимости, какими бы высокоуровневыми языками не пользовался программист. Наличие в этом случае грамотной блок-схемы значительно упрощает ситуацию.

Очередным непростым этапом разработки надёжного встроенного программного обеспечения, является тестирование. Это очень ответственный и зачастую долгий процесс. И тут как нельзя кстати будет наличие под рукой блок-схемы. По своему опыту могу сказать, что подавляющее большинство ошибок находится именно при анализе алгоритма, т.е. блок-схемы. Методики тестирования ПО у каждого программиста свои. Кому-то нравится многократно прогонять программу в симуляторе, кто-то пользуется аппаратными интерфейсами отладки, кто-то вооружившись приборами отлаживает программу в реальном времени. Все эти методы по своему хороши и обладают каждый своими плюсами и минусами, но главное, чтобы этому процессу было уделено самое пристальное внимание. Лучше всего начинать тестирование в процессе написания программы, что при применении подхода структурного программирования не составляет большого труда, так как любой программный кирпичик, будь то функция, подпрограмма или даже целая ветвь алгоритма, может быть автономно проверен и «вылизан до блеска».

Ну и пожалуй последняя рекомендация – общая культура программирования. Повторюсь, что изложенные здесь принципы – лично мое мнение, но я неоднократно убеждался на практике, насколько сильно их влияние на написание качественного и действительно надёжного ПО.

Итак, культура программирования. Принципов здесь не так много и важнейшие из них следующие:

**1) При проектировании ПО, с успехом можно применять подход к проектированию сложных систем, когда на уровне более общей структурной схемы определяются принципиальные моменты взаимодействия программных и аппаратных ресурсов.**

**2) Необходимо стремиться, по возможности, к структурному программированию (если нет крайне жёстких ограничений по аппаратно-программным ресурсам). Каждая подпрограмма и функция в идеале должна быть автономной законченной и отлаженной единицей, этаким «чёрным ящиком», дали задание – получили результат.**

**3) Из п. 2 вытекает естественное желание разделения функциональности по специализированным программным и аппаратным модулям. Как правило, опытный программист встроенных систем эффективно распределяет аппаратные ресурсы программируемого компонента в соответствии с общим алгоритмом работы всего устройства (опять блок схема).**

**4) Из п. 3 вытекает естественная потребность оформления исходных текстов в виде относительно небольших и относительно самостоятельных программ, подпрограмм и процедур, хранящихся в отдельных файлах. Написание всей программы в виде одной большой «простыни» – признак «плохого тона». К тому же с такой программой, как правило, крайне неудобно работать.**

**5) Комментарии!!! Как общие алгоритмические, так и применительно к входным/выходным параметрам подпрограмм и функций (см. выше).**

**6) Все глобальные константы и переменные, необходимые для работы программы должны быть описаны в одном месте (файле). Как правило, большинство систем программирования поддерживают многофайловые проекты (когда исходный тест программы – совокупность файлов – программных модулей и процедур). Так вот именно в одном файле должны быть декларированы и подробно описаны глобальные константы и переменные и соответственно изменяться они должны только в этом одном файле.**

**7) Никогда не использовать в исходных текстах числовые константы. Все числовые константы должны быть именованы и описаны в соответствии с п. 6.**

**8) Использовать только подходящие по смыслу имена подпрограмм, переменных и адресные метки. Абракадабра в метках и именах, или безликие идентификаторы (типа «label1», «label2», «file1» и т.п.) крайне затрудняют понимание исходного текста программы.**

**9) Головная программа проекта, или файл описания переменных должны содержать краткое функциональное описание всех подпрограмм, процедур и функций.**

**10) Снабдить головную программу или файл описания пояснениями принципиальных особенностей проекта (например, какие должны быть ключи компиляции и наименование среды разработки, какова должна быть конфигурация программируемого компонента, если она задаётся дополнительными средствами, и т.п.). А также описать в общих словах суть и алгоритм работы всей программы.**

**11) Визуально структурировать исходный текст, пользуясь отступами, выравниванием и т.п. Искусно форматированный исходный текст естественным образом позволяет «читать» программу функциональными блоками.**

**12) Всегда делать внятные и в меру подробные описания блок-схем, переменных и констант, программных текстов и т.п. Желательно исходить из того, что «исходник» должен понять (конечно, приложив немного умственных усилий), абсолютно сторонний программист. Такой подход позволит в дальнейшем, даже по прошествии некоторого времени, легко вспомнить суть программы и подправить её в случае необходимости. А уж о корпоративной работе (когда проект может «кочевать» от разработчика к разработчику) я вообще молчу, здесь это условие просто обязательно.**

**13) Имена переменных, констант, подпрограмм и меток в блок-схеме и исходном программном тексте должны совпадать.**

**14) Блок-схема должна быть всегда актуальной, т.е. всегда должно быть соответствие между блок-схемой и программным текстом. Сначала необходимо вносить изменения в блок-схему, и только потом в программный текст. Если действовать наоборот, то есть большой риск внести в программу ошибки (получится как в расхожей хохме: «в новой версии программы исправлены старые ошибки и добавлены новые» :)**

## ***ПРОБЛЕМЫ ДИЗАЙНА***

Вопрос настолько широк, что обсуждать его в рамках данной статьи не представляется возможным. Позволю себе лишь самые общие высказывания касательно данной области. Одной из основных точек соприкосновения проблем дизайна и обеспечения надёжности является эргономика. Помимо внешнего вида изделия, как воплощения полёта мысли дизайнера, существует еще и эргономика, как направление создания удобных для человека вещей. Частенько именно эргономика страдает. За множеством модных интерфейсов и «лампочек» часто не найти нужных органов управления, которые, казалось бы, должны быть на первом месте. Излишне броская или не в меру яркая индикация (как впрочем и излишне блёклая и незаметная) часто мешает восприятию необходимых данных. Как первейший элемент ответственный за эффективность взаимодействия человека с техникой, именно эргономика самым важным образом влияет на надёжность этого взаимодействия. Не нужно забывать, что в соответствии с теми же законами Мерфи «*любая система, зависящая от человеческой надёжности – ненадёжна*», и

задача разработчика свести влияние человеческого фактора к минимуму. На втором месте стоит сопряжение дизайнерских решений с конструкционными и технологическими решениями проектируемого изделия. Такие вещи как материаловедение, результаты конструкционных расчётов (тепловых, прочностных, ЭМС и т.п.), технологии производства и сборки должны учитываться при создании дизайн-концепции изделия.

### ***ЗАДАЧИ ТЕСТИРОВАНИЯ***

На самом деле тестирование технических решений идёт всегда параллельно разработке (аналогично рекомендации начинать тестирование ПО уже в процессе его написания). В принципе, разработка устройства, в некотором роде и есть мысленное тестирование на возможность работы в разных условиях. Помимо традиционного макетирования и программного моделирования всегда полезно проверять работу узлов схемы в условиях, приближенным к рабочим. Гораздо меньше сюрпризов преподносит система, построенная из отлаженных и испытанных ранее узлов. Это конечно не отменяет финальных испытаний устройства на соответствие требуемых в ТЗ характеристик. Для облегчения процесса тестирования, как в последующем и процесса обслуживания или ремонта, необходимо предусмотреть на схемах и печатных платах необходимый набор контрольных точек, диагностических разъёмов и т.п. При массовом производстве устройства, конечно необходимо обеспечить процедуру выходного контроля изделий по ряду важнейших параметров, и естественно разработать грамотную методику тестирования. В некоторых случаях бывает нелишне подумать и о входном контроле качества компонентов. Качественная проверка, как и разработка вообще, немыслима без парка хороших измерительных приборов, которые далеко не всегда есть в наличии. Видимо ввиду их дороговизны разработчики часто пользуются весьма ограниченным набором приборов. По правде сказать, многие задачи действительно решаются методами косвенных измерений. Тут главное грамотно представлять, что хотим измерить, с какой точностью, и какими способами этого можно достичь. Контролировать приборами результат разработки необходимо всегда, не надеясь на характеристики комплектующих (так как кроме характеристик есть еще схемотехника, физика, банальный брак РЭ и многое другое). Ну и напоследок выскажу мысль, что изготовленный удачно один образец еще ничего не значит. Статистика и еще раз статистика. Чтобы дать заключение о работоспособности изделия (а в особо ответственных случаях даже отдельного узла) требуются испытания ряда аналогичных изделий. Часто, к сожалению, таким испытанием служит выход устройства в серию. Чтобы избежать проблем с серийным изделием процесс тестирования необходимо проводить тщательно и ответственно.

### ***ОФОРМЛЕНИЕ КД и ПД***

Тут особо распространяться не буду. Скажу лишь, что созданием качественной конструкторской и пользовательской документации сегодня пренебрегает подавляющее большинство разработчиков. А ведь хорошая конструкторская документация это и дальнейшая техподдержка изделия и его модификация и анализ отказов, и многое, многое другое. Создание грамотной конструкторской документации сопряжено с трудностями использования САПР и библиотеками УГО в отечественных стандартах, но эти трудности преодолимы (есть и коммерческие библиотеки, да и самому можно «нарисовать» необходимый минимум в соответствии с ГОСТ). Игнорирование ГОСТ при создании конечной КД, на мой взгляд, крайне нежелательно, так как, во-первых, вносит путаницу (и ошибки) в документацию, а во-вторых, значительно затрудняет техническое сопровождение изделия (особенно разными специалистами).

По поводу пользовательской документации скажу, что она совершенно необходима во избежание проблем, связанных с неправильной эксплуатацией изделия. Из практики можно сказать, что пользовательскую документацию почти всегда читают только при возникновении каких-либо проблем. Как всегда подтверждается один из законов Мерфи: «*Если ничто другое не помогает, прочтите, наконец, инструкцию*». В свете вышесказанного можно рекомендовать, во-

первых, разработку устройства с дружественным пользователю интерфейсом, всевозможными «защитами от дурака», а во-вторых, создание краткой памятки пользователя (наряду с полным руководством, которое также необходимо). Написание приличного руководства пользователя доступно далеко не всем «технарям», тут можно рекомендовать руководствоваться примерами, помощью «гуманитариев» и здравым смыслом. Как бы то ни было, написание ПД абсолютно необходимая часть работы.

Напоследок могу лишь добавить, что разработка не считается законченной, если она не документирована.

## ***ЗАКЛЮЧЕНИЕ***

Подводя итог всему вышесказанному, хочется сказать, что поднятые проблемы проектирования хоть и затрагивают вопрос надёжности разрабатываемой аппаратуры, но больше имеют отношение к квалификации инженера-разработчика. В этой, надеюсь полезной, статье я хотел поделиться некоторыми методами повышения качества работы инженера-разработчика РЭС. Ключевые моменты, многие из которых ускользают от внимания разработчика, и на которые стоит обратить внимание, отображены на рисунке-схеме (рис.18).

Статья конечно не охватывает всех проблем, возникающих при разработке РЭС и влияющих на её надёжность. Не были, например, затронуты специальные конструкционные и схемотехнические методы обеспечения надёжности, такие как различные способы резервирования, проектирование по результатам анализа критичности отказов, разработка устойчивых к сбоям программных решений и т.д. «За кадром» (или затронутыми вскользь) остались также многие конструкционные методы повышения надёжности, такие как компоновка изделия, вопросы защиты от ударов и вибраций, экранирование, проектирование печатных плат, пыле- и влагозащита и много чего ещё. Это специальные вопросы, о которых при проектировании РЭС не стоит забывать, и ответы на которые заинтересованный читатель может найти в специальной литературе. Но, тем не менее, наиболее популярные общие проблемы обеспечения надёжности РЭС при разработке я постарался осветить. В некоторых случаях даны конкретные рекомендации, в иных, просто указано на проблему, и тут уже задача разработчика не упустить её из виду и проработать методы её решения. В любом случае это не набор инструкций «как нужно делать», а скажем некий «сборник рецептов». Воспринимать их или нет – личное дело каждого. Скажу лишь, что эти рекомендации родились не за один день, а являются результатом работы над многими проектами. Их соблюдение не раз облегчало мне мою инженерную деятельность. Предвижу возражения: «Но ведь разработка будет длиться долго, и постоянно будет «спотыкаться» об эти правила!» Да, не всегда всё просто, но когда соблюдение подобных правил войдёт в систему, Вы сами уже не сможете работать по-другому, потому что вознаграждением будет безотказная работа спроектированной аппаратуры и минимум проблем с заказчиком, ведь как известно лучше «медленно запрягать, но быстро ехать». К тому же такой подход обеспечивает регламент ведения работ и чёткое понимание того что, когда, зачем и как нужно делать. Ведь согласитесь, никому не захотелось бы покупать автомобиль, изготовленный на заводе, где правит принцип «сделаем как-нибудь». Так почему же при проектировании РЭС мы, разработчики, такое себе частенько позволяем?

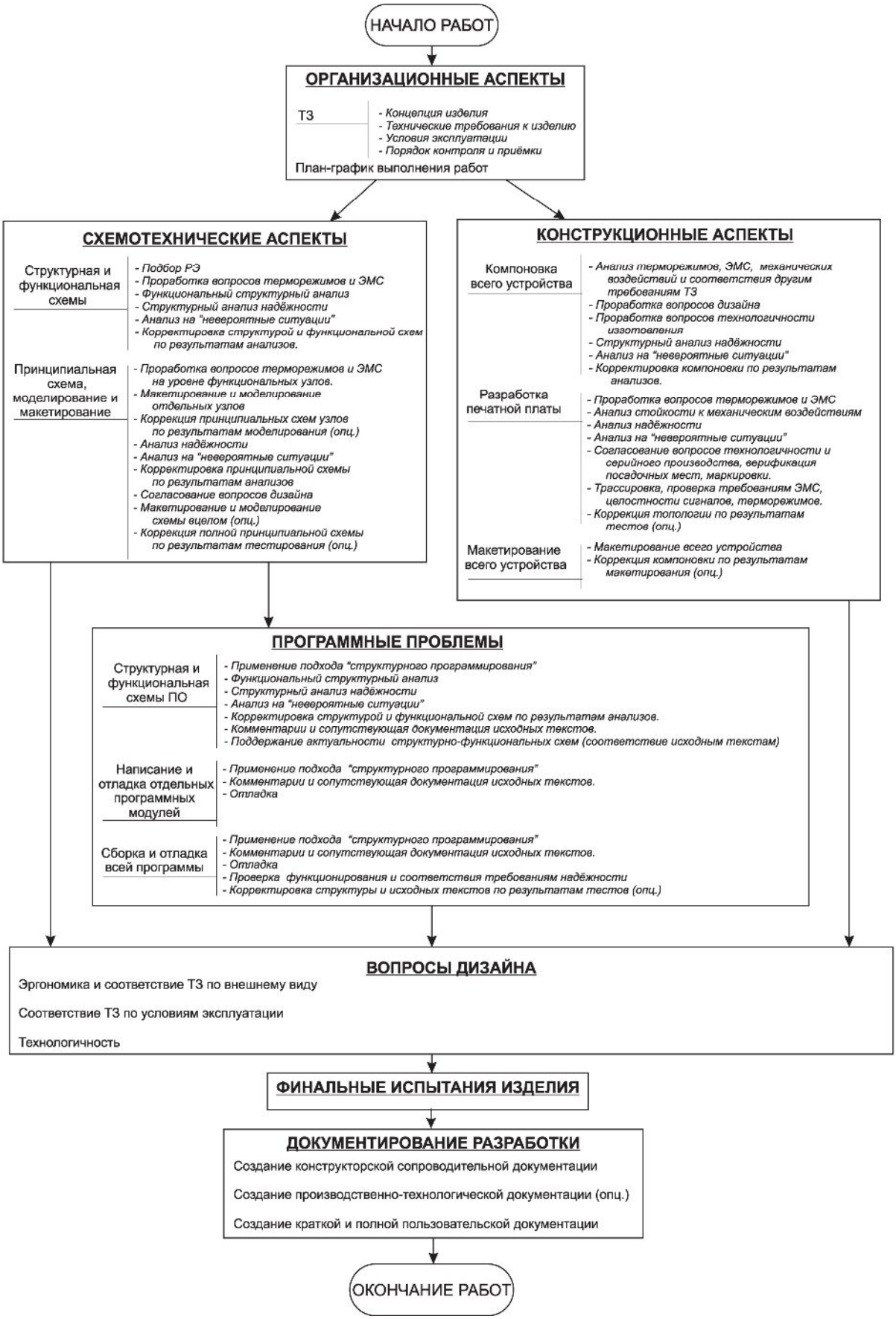
НАЧАЛО РАБОТ
ОРГАНИЗАЦИОННЫЕ АСПЕКТЫ
ТЗ
- Концепция изделия
- Технические требования к изделию
- Условия эксплуатации
- Порядок контроля и приёмки
План-график выполнения работ
СХЕМОТЕХНИЧЕСКИЕ АСПЕКТЫ
Структурная и функциональная схемы
- Подбор РЭ
- Проработка вопросов терморежимов и ЭМС
- Функциональный структурный анализ
- Структурный анализ надёжности
- Анализ на "невероятные ситуации"
- Корректировка структурой и функциональной схем по результатам анализов.
Принципиальная схема, моделирование и макетирование
- Проработка вопросов терморежимов и ЭМС на уровне функциональных узлов.
- Макетирование и моделирование отдельных узлов
- Коррекция принципиальных схем узлов по результатам моделирования (опц.)
- Анализ надёжности
- Анализ на "невероятные ситуации"
- Корректировка принципиальной схемы по результатам анализов
- Согласование вопросов дизайна
- Макетирование и моделирование схемы вцелом (опц.)
- Коррекция полной принципиальной схемы по результатам тестирования (опц.)
КОНСТРУКЦИОННЫЕ АСПЕКТЫ
Компоновка всего устройства
- Анализ терморежимов, ЭМС, механических воздействий и соответствия другим требованиям ТЗ
- Проработка вопросов дизайна
- Проработка вопросов технологичности изготовления
- Структурный анализ надёжности
- Анализ на "невероятные ситуации"
- Корректировка компоновки по результатам анализов.
Разработка печатной платы
- Проработка вопросов терморежимов и ЭМС
- Анализ стойкости к механическим воздействиям
- Анализ надёжности
- Анализ на "невероятные ситуации"
- Согласование вопросов технологичности и серийного производства, верификация посадочных мест, маркировки.
- Трассировка, проверка требованиям ЭМС, целостности сигналов, терморежимов.
- Коррекция топологии по результатам тестов (опц.)
Макетирование всего устройства
- Макетирование всего устройства
- Коррекция компоновки по результатам макетирования (опц.)
ПРОГРАММНЫЕ ПРОБЛЕМЫ
Структурная и функциональная схемы ПО
- Применение подхода "структурного программирования"
- Функциональный структурный анализ
- Структурный анализ надёжности
- Анализ на "невероятные ситуации"
- Корректировка структурой и функциональной схем по результатам анализов.
- Комментарии и сопутствующая документация исходных текстов.
- Поддержание актуальности структурно-функциональных схем (соответствие исходным текстам)
Написание и отладка отдельных программных модулей
- Применение подхода "структурного программирования"
- Комментарии и сопутствующая документация исходных текстов.
- Отладка
Сборка и отладка всей программы
- Применение подхода "структурного программирования"
- Комментарии и сопутствующая документация исходных текстов.
- Отладка
- Проверка функционирования и соответствия требованиям надёжности
- Корректировка структуры и исходных текстов по результатам тестов (опц.)
ВОПРОСЫ ДИЗАЙНА
Эргономика и соответствие ТЗ по внешнему виду
Соответствие ТЗ по условиям эксплуатации
Технологичность
ФИНАЛЬНЫЕ ИСПЫТАНИЯ ИЗДЕЛИЯ
ДОКУМЕНТИРОВАНИЕ РАЗРАБОТКИ
Создание конструкторской сопроводительной документации
Создание производственно-технологической документации (опц.)
Создание краткой и полной пользовательской документации
ОКОНЧАНИЕ РАБОТ

Рисунок 18

В заключение пожелаю коллегам-инженерам побольше успешных проектов полезных и надёжных устройств. Все-таки хочется, чтобы техника работала на нас, а не мы на неё.

Литература

1) *«Влагозащита печатных узлов» Уразаев В.Г. 2006*

2) *«Выполнение электрических схем по ЕСКД» Усатенко С.Т., Каченюк Т.К., Терехова М.В. 1989*

3) *«Искусство схемотехники 1-3» Хоровиц П., Хилл У. 1993*

4) *«Как читать DATA SHEET на микросхемы» Воробьёв Е. журнал «Электронные компоненты» №3 2008 стр.17-20*

5) *«Механические воздействия и защита радиоэлектронной аппаратуры» Маквецов Е.Н., Тартаковский А.М. 1993*

6) *«Полупроводниковые приборы» Тугов Н.М., Глебов Б.А., Чарыков Н.А. 1990*

7) *«Силовые полупроводниковые ключи» Воронин П.А. 2001*

8) *«Справочник конструктора РЭА: Общие принципы конструирования» под ред. Варламова Р.Г 1980*

9) *«Схемотехника аналоговых электронных устройств» Павлов В.Н., Ногин В.Н. 1997*

10) *«Теория надёжности радиотехнических систем» Левин Б.Р. 1978*