**РЕФЕРАТ**

Пояснювальна записка містить: сторінок, розділів, ілюстрацій, таблиць, додатків, джерел.

Об'єкт проектування - цифровий генератор-вимірювач низькочастотних сигналів.

Мета роботи - проектування двухпортового цифрового пристрою для генерації, та вимірювання параметрів, низькочастотних сигналів, в діапазоні 0 В ... +5 В довільної форми, на основі мікроконтролера.

Метод проектування — розрахунок та конструювання окремих вузлів генератора-вимірювача. У роботі розглянуті існуючі варіанти виконання як аналогових, так і цифрових генераторів-вимірювачів. Виділені їх  
переваги і недоліки. Проведено аналіз основних технічних рішень, на основі яких розроблена структурна і функціональна схеми цифрового генератора-вимірювача. Зроблено вибір елементної бази та розрахунок окремих вузлів цифрового генератора-вимірювача. Написано і виконано налагодження програмного забезпечення мікропроцесора генератора-вимірювача.

Виготовлено дослідний зразок спроектованого пристрою.  
Пропонується використовувати результати проекту у навчальному  
процесі для оснащення лабораторій, що займаються обробкою і аналізом сигналів і процесів в радіоелектронних пристроях.

ЦИФРОВИЙ, АНАЛОГОВИЙ, ГЕНЕРАТОР, СИГНАЛ, МІКРОКОНТРОЛЕР, КЛАВІАТУРА, РІДКОКРИСТАЛІЧНИЙ ІНДИКАТОР, ДРУКОВАНА ПЛАТА.

**РЕФЕРАТ**

Пояснительная записка содержит: страниц, разделов, иллюстраций, таблиц, приложений, источников.

Объект проектирования – цифровой генератор-измеритель низкочастотных сигналов.

Цель работы – проектирование двух портового цифрового устройства для генерации, и измерении параметров, низкочастотных сигналов, в диапазоне 0 В … +5 В произвольной формы, на основе микроконтроллера.

Метод проектирования – расчет и конструирование отдельных узлов генератора-измерителя. В работе рассмотрены существующие варианты исполнения как аналоговых, так и цифровых генераторов-измерителей. Выделены их достоинства и недостатки. Проведен анализ основных технических решений, на основе которого разработана структурная и функциональная схемы цифрового генератора-измерителя. Произведен выбор элементной базы и расчет отдельных узлов цифрового генератора-измерителя . Написано и выполнена отладка программного обеспечения микропроцессорного устройства генератора-измерителя . Изготовлен опытный образец спроектированного устройства.

Предлагается использовать результаты проекта в учебном процессе для оснащения лабораторий, занимающихся обработкой и анализом сигналов и процессов в радиоэлектронных устройствах.

ЦИФРОВОЙ, АНАЛОГОВЫЙ, ГЕНЕРАТОР, СИГНАЛ, МИКРОКОНТРОЛЛЕР, КЛАВИАТУРА, ЖИДКОКРИСАЛЛИЧЕСКИЙ ИНДИКАТОР, ПЕЧАТНАЯ ПЛАТА.

**ABSTRACT**

Explanatory note contains: pages, sections, illustrations, tables, applications, sources.

Object design - digital low-frequency measuring-generator.

Objective - designing the dual port digital device for generating, and measuring parameters of, low-frequency signals in the range of 0 V ... +5V, arbitrary form, based on a microcontroller.

Design method – calculation and design of individual nodes measuring-generator. The paper discusses the current versions of both analog and digital measuring-generator. Highlighted their advantages and disadvantages. Designing of the basis of structural and functional digital circuit measuring-generator has been done based on analysis of basic technical solutions. Material selection element base and calculation of individual nodes of digital measuring-generator have been done. Software of microprocessor unit of measuring-generator has been written and debugged. The prototype of the designed device has been made.

It is proposed to use the results of the project in the educational process for laboratories engaged of processing and signal analysis and processes in electronic devices.

DIGITAL, ANALOG, GENERATOR, SIGNAL, MICROCONTROLLERS, KEYBOARD, LCD, PRINTED CIRCUIT BOARD.

содержание

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 9](#_Toc378606155)

[ВВЕДЕНИЕ 10](#_Toc378606156)

[1 ОБЗОР МЕТОДОВ И УСТРОЙСТВ ГЕНЕРАЦИИ СИГНАЛОВ 11](#_Toc378606157)

[1.1 Основные виды аналоговых генераторов периодических сигналов 11](#_Toc378606158)

[1.1.1 RC-генераторы аналоговых сигналов 11](#_Toc378606159)

[1.1.2 LC-генераторы аналоговых сигналов 13](#_Toc378606160)

[1.1.3 Кварцевые генераторы 16](#_Toc378606161)

[1.1.4 Генераторы на пьезокерамических фильтрах 18](#_Toc378606162)

[1.2 Основные виды цифровых генераторов сигналов 19](#_Toc378606163)

[1.2.1 Генераторы на основе цифрового синтеза частоты 19](#_Toc378606164)

[1.2.2 Генераторы произвольных функций и формы сигналов 21](#_Toc378606165)

[1.3 Сравнительная характеристика цифровых и аналоговых генераторов 23](#_Toc378606166)

[1.4 Методы генерации шумоподобных сигналов 23](#_Toc378606167)

[1.4.1 Аналоговые генераторы шума 23](#_Toc378606168)

[1.4.2 Генерация случайных процессов с нормальным распределением плотности вероятности на основе генератора М-последовательностей 26](#_Toc378606169)

[1.4.3 Цифровые методы генерации случайных процессов с нормальным и равномерным распределениями плотности вероятности 28](#_Toc378606170)

[2 РАЗРАБОТКА ЦИФРОВОГО ГЕНЕРАТОРА-ИЗМЕРИТЕЛЯ СИГНАЛОВ 35](#_Toc378606171)

[2.1 Функциональная схема устройства 35](#_Toc378606172)

[2.2 Расчет характеристик и выбор элементов структурной схемы 39](#_Toc378606173)

[2.2.1 Расчет параметров цифро-аналогового преобразователя 39](#_Toc378606174)

[2.2.2 Выбор жидкокристаллического индикатора 40](#_Toc378606175)

[2.2.3 Подключение клавиш управления 43](#_Toc378606176)

[2.2.4 Выбор вычислительного устройства 44](#_Toc378606177)

[2.2.5 Выбор интерполирующего фильтра 45](#_Toc378606178)

[2.2.6 Общая структурная схема генератора 47](#_Toc378606179)

[2.3 Расчет элементов принципиальной схемы генератора 49](#_Toc378606180)

[2.3.1 Расчет схемы стабилизации 12 В 49](#_Toc378606181)

[2.3.2 Расчет схемы защищающей ножки контроллера 50](#_Toc378606182)

[2.3.3 Расчет схемы инвертирующего усилителя 50](#_Toc378606183)

[2.3.4 Расчет схемы измерителя тока 51](#_Toc378606184)

[2.3.5 Расчет схемы не инвертирующего усилитель с резистивным мостом (делителем напряжения) 52](#_Toc378606185)

[2.3.6 Расчет интерполирующего фильтра 53](#_Toc378606186)

[2.3.7 Выбор буферного усилители 55](#_Toc378606187)

[2.4 Проектирование печатного узла 56](#_Toc378606188)

[2.5 Разработка программного обеспечения цифрового устройства генератора.. 59](#_Toc378606189)

[2.5.1 Выбор среды и языка программирования для разработки программного обеспечения для МК 59](#_Toc378606190)

[2.5.2 Описание программы для генератора-измерителя НЧ сигналов 60](#_Toc378606191)

[2.5.3 Блок-схемы программного обеспечения МК генератора 71](#_Toc378606192)

[ЗАКЛЮЧЕНИЕ 73](#_Toc378606193)

[ПЕРЕЧЕНЬ ССЫЛОК 74](#_Toc378606194)

[ПРИЛОЖЕНИЕ А – ПРОГРАММА MATLAB ДЛЯ МОДЕЛИРОВАНИЯ СИГНАЛОВ НА ВЫХОДЕ ЦАП И ИХ СПЕКТРОВ 75](#_Toc378606195)

[ПРИЛОЖЕНИЕ Б – ОПИСАНИЕ ФАЙЛОВ И ФУНКЦИЙ ПРОЕКТА ГЕНЕРАТОРА-ИЗМЕРИТЕЛЯ 79](#_Toc378606196)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

AFG – Arbitrary Function Generator (генератор произвольных функций);

AWG – Arbitrary Waveform Generator (генераторы сигналов произвольной формы);

DDS – Direct Digital Synthesis (прямой цифровой синтез);

MSPS – Million Samples per Second (миллионов выборок в секунду);

АЧХ – амплитудно-частотная характеристика;

БУ – буферный усилитель;

ДПКД – делитель с переменным коэффициентом деления;

ЖКИ – жидкокристаллический индикатор;

ИФ – интерполирующий фильтр;

ИЦ – избирательная цепь;

КЛ – клавиатура;

МДНФ – минимальная дизъюнктивная нормальная форма;

МК – микроконтроллер;

НЧ – низкие частоты;

ОЗУ – оперативное запоминающее устройство;

ОС – обратная связь;

ОУ – операционный усилитель;

ПЗУ – постоянное запоминающее устройство;

ПСП – псевдослучайная последовательность;

ТТЛ – транзисторно-транзисторная логика;

ФНЧ – фильтр нижних частот;

ФЧХ – фазо-частотная характеристика;

ЦАП – цифроаналоговый преобразователь;

ЦВУ – цифровое вычислительное устройство;

ЦЛЗ – цифровая линия задержки.

# ВВЕДЕНИЕ

Цифровой генератор сигналов – это генератор сигналов на основе программного и аппаратного обеспечения, который может генерировать основные сигналы.

В инженерной работе различные сложные сигналы состоят из этих основных сигналов, которые могут быть просто сформированы с помощью цифровых генераторов сигналов. Цифровыми методами можно генерировать сигналы произвольной формы. Это сигналы широко используется при анализе техники и экспериментальном обучения. Цифровые генераторы низких частот по сравнению с аналоговыми характеризуются более эффективными метрологическими характеристиками: высокой точностью установки и стабильностью частоты, малым коэффициентом нелинейных искажений (строго синусоидальной формой), постоянством уровня выходного сигнала. Цифровые генераторы, получающие все более широкое распространение, удобнее аналоговые в эксплуатации: выше быстродействие, существенно проще установка требуемой частоты, более наглядна индикация. Кроме того, цифровые генераторы имеют возможность автоматической перестройки частоты по заранее заданной программе и применения в сочетании с цифровыми средствами обработки информации. Поэтому они получают всё большее распространение, как узлы электронной аппаратуры, так и как самостоятельные устройства, применяемые при измерениях и налаживании систем, работающих со сложными сигналами.

Целью данного проекта является разработка цифрового НЧ генератора-измерителя. Это устройство должно генерировать сигналы заданных форм на выходах двух, независимых, аналоговых каналов, производить измерние их параметров (генерируемый ток и напряжение) и записывать на sd-карточку. Форма и параметры сигналов для каждого из независимых каналов должны задаваться с помощью клавиатуры и отображаться на жидкокристаллическом индикаторе (ЖКИ). В процессе проектирования я выучил теорию основных видов аналоговых и цифровых генераторов сигналов, подробнее разобрался с принципом действия соответствующих компонентов и программного обеспечения (Matlab и OrCAD).

# ОБЗОР МЕТОДОВ И УСТРОЙСТВ ГЕНЕРАЦИИ СИГНАЛОВ

* 1. **Основные виды аналоговых генераторов периодических сигналов**
     1. **RC-генераторы аналоговых сигналов**

Генераторы, которые используют RC цепи, комбинацию резисторов и конденсаторов, в их частотно избирательных частях, называются RC генераторами. RC-генераторы используются для генерации сверхнизких и низких частот, а также радиочастот примерно до 2-5 МГц, Как правило, модуляции у таких генераторов не используется за исключением некоторых моделей с частотой генерируемых сигналов выше 100кГц [1].

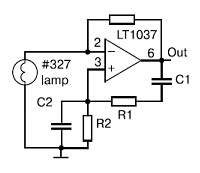


Рисунок 1.1 – Типичная схема RC-генератора на операционном усилителе

На рис. 1.1 показана одна схема RC-генераторов с Г-образной избирательной цепей (ИЦ), образованной последовательной (R1C1) и (R2C2) RC-цепями.

Из рис. 1.2 можно получить выходное напряжение:

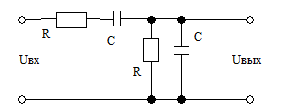
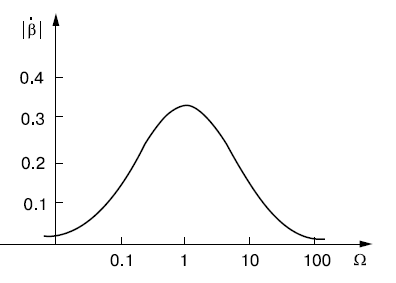
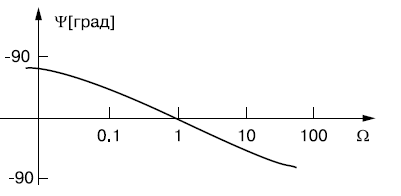


Рисунок 1.2 – Сокращенная схема RC-генератора

 (1.1)

Амплитудно-частотная характеристика (АЧХ) и фазо-частотная характеристика (ФЧХ) именно этой цепи показаны на рис. 1.3 а) и б). На частоте генерации такая цепь имеет угол сдвига фазы, который равно 0. На приведенных, на рис. 1.1, данных, схема рассчитана на генерацию фиксированной частоты в 1000 Гц, задаваемой с высокой точностью.

а) АЧХ RC-генератора б) ФЧХ RC-генератора

Рисунок 1.3 – АЧХ и ФЧХ RC-генератора

Частота генерируемого напряжения (обычно *R*1=*R*2=*R* и *C*1=*C*2=*C*) определяется так:

 (1.2)

В этом случае цепь положительной обратной связи на частоте f0 вносит ослабление 1+(R1/R2)+(C1/C2) равное 3 при *R*1=*R*2 и *C*1=*C*2. Для получения синусоидальных колебаний вводится цепь отрицательной обратной связи, которая должна обеспечивать усиление на уровне чуть больше 3 (условия баланса амплитуд).

Как видно из рис. 1.3 а)и б), АЧХ и ФЧХ ИЦ в данном генераторе довольно пологие. Это говорит о том, что стабильность частоты не может быть высокой. Это характерно практически для всех известных схем *RC*-генераторов синусоидального (или почти синусоидального) напряжения. Главные достоинства таких генераторов заключаются в отсутствии катушек индуктивности, сложных в изготовлении и громоздких, а также в широком перекрытии частоты при ее изменении изменением *R* или *C*.

* + 1. **LC-генераторы аналоговых сигналов**

LC-генераторы – это генераторы с индуктивной обратной связью. Принцип автогенератора можно понять, если представить себе, что в рассмотренном ранее резонансном усилителе электрические колебания поступают на вход не от внешнего источника, а с выхода этого же усилителя через цепь обратной связи. Если на схему подано напряжение питания, то в коллекторной (стоковой) цепи протекает ток, который содержит как постоянную, так и флуктуирующую составляющие. С чем связано появление флуктуирующей составляющей? Электрические заряды в цепи имеются в определенном количестве. Причем в любой цепи реальное количество носителей заряда постоянно меняется. Амплитудный спектр флуктуирующего сигнала равномерен вплоть до частот 1012 Гц. В спектре флуктуаций всегда найдется составляющая, частота которой близка к резонансной частоте контура, включенного в выходную цепь транзистора. За счет избирательных свойств контура эта спектральная составляющая будет выделена, а через цепь обратной связи (ОС) поступит на вход усилителя.

Колебание, поступившее на вход, будет усиленно, как в обычном усилителе. В дальнейшем произойдет нарастание сигнала. Для этого необходимо:

* ОС между входом и выходом должна быть положительной (это фазовое условие самовозбуждения);
* коэффициент усиления (Кус) должен превышать определенное значение (амплитудное условие самовозбуждения).

На высоких частотах (от десятков кГц до сотен МГц и выше) применяются *LC-*генераторы на основе высокодобротных *LC-*контуров. Частота контура (последовательного или параллельного), на которой фазовый сдвиг равен 0, обычно очень близка к резонансной частоте идеального последовательного *LC*-контура:

 (1.3)

Фильтрующая способность *LC*-контура определяется его добротностью *Q* - отношением реактивного сопротивления элементов *L* или *C* к активному сопротивлению потерь r контура. Значения *Q* у радиочастотных контуров составляют несколько десятков и даже сотен. При этом *Q* определяет и полосу частот контура *f*=*f*0/*Q*. Очевидно, что чем выше *Q*, тем более стабильна частота генератора. Стабильность частоты *LC-*генераторов на 1–2 порядка выше, чем у *RC-*генераторов. Но относительная нестабильность частоты редко получается меньшей 10-4.

Благодаря высокой фильтрующей способности колебательных *LC*-контуров получение синусоидальной формы от *LC*-генераторов оказывается более простой задачей, чем в случае построения *RC*-генераторов. Однако и тут простые схемы могут давать *K*Г до нескольких процентов. Часто вместо *K*Г чистоту спектра высокочастотных генераторов оценивают просто по уровню гармоник сигнала.

Классическая схема автогенератора на транзисторе, включенном по схеме с общей базой, представлена на рис. 1.4. Частота генерации задается параллельным *LC-*контуром. Каскад с общей базой не инвертирует фазу, поэтому для создания положительной обратной связи достаточно подать сигнал с части контура на эмиттер транзистора. Эта схема (при использовании соответствующего транзистора и контура) может работать на частотах от десятков кГц до сотен МГц и выше. Поскольку входное сопротивление каскада с общей базой мало, то необходимо согласование между высокоомной выходной цепью усилителя и его низкоомной входной цепью. Оно и достигается неполным включением контура.

Еще одна классическая схема *LC*-генератора показана на рис. 1.5. Здесь используется каскад с общим коллектором (эмиттерный повторитель), который тоже не инвертирует фазу входного сигнала, но имеет коэффициент передачи несколько меньший 1. Поэтому для соблюдения условия баланса амплитуд надо использовать повышающее напряжение автотрансформаторное включение колебательного контура.

Заметим, что каскад с общим коллектором, как и каскад с общей базой, имеют наилучшие частотные свойства, чем каскад с общим эмиттером. Это гарантирует устойчивую работу автогенератора на высоких частотах.

Множество генераторов создано на основе каскада с общим эмиттером, дающего, как известно, наибольшее усиление по мощности. Однако, эта схема каскада не очень удачна для построения генераторов из-за намного худших частотных свойств биполярного транзистора, включенного с общим эмиттером.

Подобные автогенераторы строятся и на полевых транзисторах по схеме с общим истоком, достоинством которых является высокое входное сопротивление.

Главные недостаток таких генераторов: для формирования высоких частот генерации, нужно увеличивать значение L, методом увеличение число витков индуктивности, но при этом увеличиваются и размеры катушек.

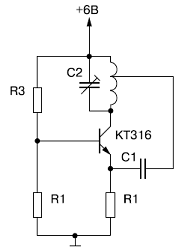


Рисунок 1.4 – LC-генератор на транзисторе, включенном по схеме с общей базой, и контуре с неполным включением

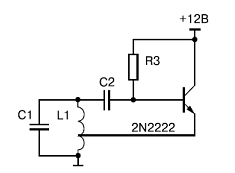


Рисунок 1.5 – LC-генератор на транзисторе, включенном по схеме с общим

коллектором с автотрансформаторным включением контура

* + 1. **Кварцевые генераторы**

Относительная нестабильность частоты автогенераторов, выполняемых на резонаторах в виде LC-контуров и RC-контуров, обычно не ниже 10-3...10-4 [2].

Стабильность частоты генератора существенно зависит от добротности и стабильности колебательной системы. Добротность LC-контура обычно не выше 200...300. К современным радиопередатчикам и приемникам предъявляются более высокие требования по стабильности частоты. Обычно требуется долговременная относительная нестабильность частоты не хуже чем 10-6...10-8, что можно обеспечить, применяя кварцевые резонаторы. Добротность кварцевых резонаторов во много раз превышает добротность резонаторов на LC-контурах и составляет 104...106.

Существует много схем кварцевых автогенераторов. Поэтому рассмотрим наиболее часто применяемые на практике схемы.

Общепринятая эквивалентная схема кварцевого резонатора изображена на рис. 1.6. Динамическая индуктивность Ls, динамическая емкость Cs и сопротивление потерь Rs обусловлены наличием прямого и обратного пьезоэффекта и резонансными свойствами пьезоэлемента. Параллельная емкость Ср обусловлена межэлектродной емкостью пьезоэлектрика, емкостью корпуса и монтажа. Резонансная частота динамической ветви называется частотой последовательного резонанса кварцевого резонатора Fs.

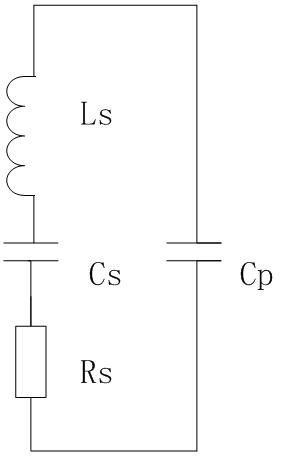


Рисунок 1.6 – Общепринятая эквивалентная схема кварцевого резонатора

Добротность кварцевого резонатора Q определяется динамической ветвью в соответствии с формулой для последовательного колебательного контура:

Частота параллельного резонанса Fp несколько выше Fs, что обусловлено параллельным резонансом Ср, Cs и Ls. Важным параметром кварцевого резонатора является отношение его параллельной емкости к динамической, обозначаемое г и называемое емкостным коэффициентом r = Cc/Cs.

По разным литературным источникам, емкостной коэффициент для АТ-среза кварца равен 220...250.

Учитывая, что Cs/Cp<0,1, можно пользоваться приближенным выражением для частоты параллельного резонанса:

Fp = Fs(1 + (Cs/2Cp)) (1.5)

Для емкостного коэффициента г > 25 резонансный интервал, определяемый как разность между частотами параллельного и последовательного резонансов кварцевого резонатора, можно записать в виде dF=Fs/2r.

На механических гармониках кварцевого резонатора резонансный интервал уменьшается и определяется выражением:

dFn = Fs/(2rn2), (1.6)

где n - номер гармоники.

Емкостной коэффициент определяет величину резонансного промежутка резонатора, следовательно, девиацию частоты управляемого кварцевого генератора, стабильность частоты при изменении параметров схемы, условия возникновения и поддержания колебаний в схеме кварцевого автогенератора. Для оценки способности кварцевого резонатора возбуждаться, в некоторых схемах кварцевых генераторов используют параметр, называемый фактором качества. Он определяется как отношение добротности резонатора к его емкостному коэффициенту м = Q/r.

Для кварцевых резонаторов значения М лежат в пределах от 1 до 10000. При М < 2 реактивное сопротивление резонатора оказывается положительным (емкостным) и не имеет области индуктивной реакции. Следовательно, возбуждение такого резонатора в схемах кварцевых генераторов, требующих индуктивной реакции, становится невозможным. При М > 2 резонатор имеет область индуктивной реакции, и чем больше значение М, тем эта область шире.

На практике шире всего распространены два вида кварцевых генераторов:

а) генераторы, в которых кварцевый резонатор является частью колебательного контура и эквивалентен индуктивности;

б) генераторы, в которых кварцевый резонатор включен в цепь обратной связи, используется как узкополосный фильтр и эквивалентен активному сопротивлению.

Кварцевые генераторы, в которых кварцевый резонатор используется в качестве элемента контура с индуктивной реакций, называют осцилляторными, а генераторы, в которых кварцевый резонатор включен в цепь обратной связи, называют генераторами последовательного резонанса.

* + 1. **Генераторы на пьезокерамических фильтрах**

Колебательная система автогенератора вовсе не обязательно должна быть *LC-*контуром. Возможно применение пьезокерамических фильтров, кварцевых резонаторов и даже камертонов с электромагнитами. На рис. 1.7 показаны две схемы автогенераторов с пьезокерамическими фильтрами, обычно предназначенными для применения в усилителях промежуточной частоты супергетеродинных радиоприемников.

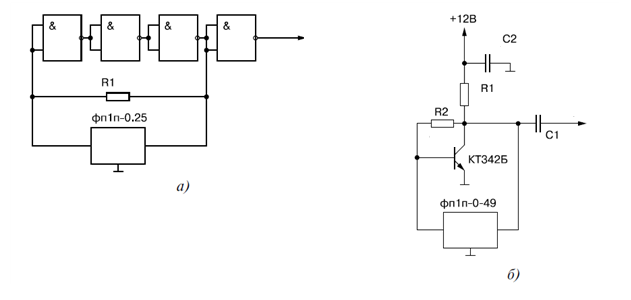


Рисунок 1.7 – Автогенераторы на основе пьезокерамических фильтров на логической схеме а) и биполярном транзисторе б)

Такие фильтры выпускаются в больших количествах, они дешевы и доступны. Фильтры обладают свойством инвертирования входного сигнала и потому для получения положительной обратной связи надо использовать инвертирующий усилитель. В генераторе на рис. 1.12, а)усилитель построен на основе ТТЛ-схем, а в генераторе на рис. 1.12, б)— на биполярном транзисторе.

Добротность пьезокерамических фильтров как резонаторов выше добротности *LC-*контура. Поэтому генераторы имеют более высокую стабильность частоты, но уступающую стабильности частоты кварцевых генераторов.

* 1. **Основные виды цифровых генераторов сигналов**
     1. **Генераторы на основе цифрового синтеза частоты**

Достоинством обычных генераторов синусоидальных сигналов является возможность получения синусоидальной формы выходного сигнала с малыми нелинейными искажениям. А главным недостатком — низкая стабильность частоты. Исключением являются кварцевые генераторы, но они обычно генерируют сигналы только одной частоты — основной или ее гармоники.

Для генерации синусоидальных сигналов с высокой стабильностью частоты используются генераторы на основе цифрового синтезатора частот. Типичная функциональная схема такого генератора показана на рис. 1.8. Основу генератора составляют два генератора. Первый генератор — это высокостабильный опорный генератор эталонной частоты *f*эт и делитель частоты с фиксированным коэффициентом деления M. Он формирует разрешение по частоте *f*1 = *f*эт/M.

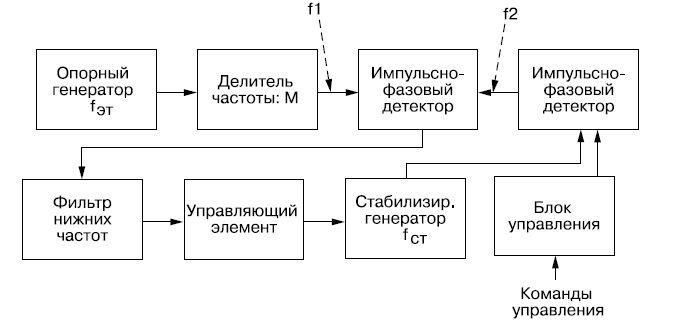


Рисунок 1.8 – Функциональная схема генератора синусоидального сигнала на основе цифрового синтезатора частоты

Второй генератор — это стабилизированный генератор, перестраиваемый в достаточно широких пределах цепью фазовой автоподстройки частоты. Его сигнал делится в *N* раз с помощью делителя с переменным коэффициентом деления (ДПКД). Для перестройки делителя используется блок управления. Сигнал с частотой *f*1 = *f*ст/*N* сравнивается с сигналом с частотой *f*2 = *f*эт/*M* с помощью импульсного фазового детектора. Его выходной сигнал фильтруется фильтром низких частот и подается на регулирующий элемент, меняющий частоту стабилизированного генератора до тех пор, пока не будет обеспечено условие *f*1 = *f*2, что соответствует установившейся частоте стабилизированного генератора *f*ст = *f*эт•*N*/*M*.

Например, если нужно получить частоты от 1 до 2 МГц с шагом в 1 Гц, надо при *f*эт =1 МГц иметь *M* = 1•106 и *N* от 1•106 до 2•106.

Стабилизированный генератор с управляющим элементом может строиться по любой известной схеме, например, *LC*-генератора с перестройкой частоты варикапом или электромагнитом. Если нужна перестройка по частоте в широких пределах (например, почти с 0), то можно использовать пару генераторов со смесителем, т. е. генератор на биениях.

Поскольку в генераторах этого типа не используется квантование по амплитуде сигнала, последний является непрерывным и может иметь весьма близкую к синусоидальному форму. Генераторы с функциональной схемой (Рис. 1.8) и разными ее вариантами выпускаются в виде специализированных интегральных микросхем частотных синтезаторов.

* + 1. **Генераторы произвольных функций и формы сигналов**

Генераторы второго класса — это приборы серии AFG, использующие технологию прямого цифрового синтеза сигналов (Direct Digital Synthesis или DDS), имеют функциональную схему, упрощенно представленную на рис. 1.9. Корпорация Tektronix называет такие генераторы как AFG (Arbitrary Function Generator — Генератор Произвольных Функций). Генерация сигналов в этом новейшем поколении цифровых генераторов основана на использовании заранее подготовленных и хранящихся в памяти оцифрованных *N* выборок сигнала за его один период. Типичное число выборок составляет от нескольких сотен до многих тысяч. Каждая выборка представляет значение сигнала в виде числа с некоторой разрядностью, которая имеет значения от 8 до 14 бит и определяет разрядность кодирования сигналов по амплитуде.

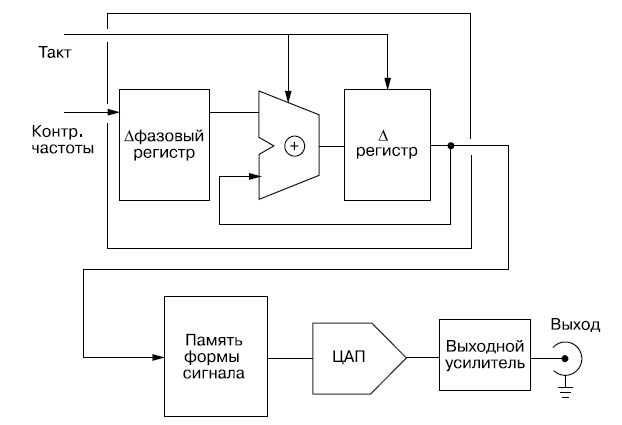


Рисунок 1.9 – Функциональная схема генератора AFG (Arbitrary Function Generator —Генератор Произвольных Функций), использующего прямой цифровой синтез сигналов

Генерация сигнала с точно заданной частотой основана на выработке адреса отсчетов сигнала на основе фиксации дискретного фазового сдвига с помощью Δ-фазового регистра и уровня сигнала с помощью Δ-регистра. Как только набегает фазовый сдвиг 360°, выработка адреса прекращается (в режиме единичного запуска) или возобновляется заново (в непрерывном режиме или в режиме генерации пачки сигналов). Блок памяти форм хранит шаблоны цифровых сигналов различной формы (в том числе произвольной, заданной пользователем). В этой главе мы пока рассматриваем реализацию генерации только синусоидального сигнала. Цифроаналоговый преобразователь (ЦАП) соответствующей разрядности и быстродействия преобразует цифровые сигналы в аналоговые, которые усиливаются выходным усилителем.

Генераторы класса AWG (Arbitrary Waveform Generator — генераторы сигналов произвольной формы) основаны на более простом, но тоже вполне современном, способе задания адресов (Рис. 1.10), что обеспечивает получение сигналов типовых форм, но несколько меньшие возможности управления ими. Корпорация Tektronix ранее выпускала серии таких генераторов AWG400/500/700. В настоящее время они заменяются более новыми и уже выпускаемыми генераторами серий AWG5000/7000. О выпуске генераторов серии AWG7000 было объявлено в марте 2007г.

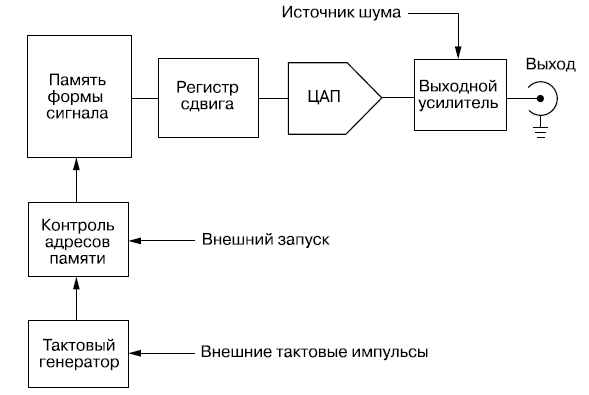


Рисунок 1.10 – Упрощенная функциональная схема генераторов класса AWG

Высокие (до 20 Гвыб/с) скорости выборки у этих приборов достигнуты за счет применения специальной технологии сверхбыстрых переключающих гетеропереходных приборов на германии и арсениде галлия. Разумеется, это удорожает эти уникальные приборы.

* 1. **Сравнительная характеристика цифровых и аналоговых генераторов**

Бурное развитие цифровой электронной техники позволяет во все большем числе случаев формирования аналоговых сигналов использовать цифровые методы. Так как цифровые генераторы аналоговых сигналов обладают рядом достоинств:

- универсальность, поскольку они позволяют генерировать аналоговый сигнал с произвольной, заданной пользователем, формой;

- отсутствие ограничения по минимальной частоте;

- высокая стабильность параметров выходного сигнала

и другие.

Цифровые генераторы обладают универсальностью, точностью и удобством настройки. Поэтому они получают всё большее распространение как узлы электронной аппаратуры, тат и как самостоятельные устройства, применяемые при измерениях и налаживании систем, работающих со сложными сигналами. Аналоговые генераторы используются в тех случаях, когда нет высоких требований к параметрам генератора, или важна простота и минимальная стоимость узла.

* 1. **Методы генерации шумоподобных сигналов**
     1. **Аналоговые генераторы шума**

Генераторы шума применяются в качестве источников флуктуационных помех при исследовании предельной чувствительности радиоприемных и усилительных устройств, в качестве калиброванных источников мощности при измерении напряженности поля или шумов внеземного происхождения, в качестве имитаторов полного сигнала в многоканальной аппаратуре связи, для измерения нелинейных искажений и частотных характеристик радиоустройств.

Основным требованием к генераторам шума является равномерность спектрального состава шумового сигнала в возможно большей полосе частот, от 0 до  «белый» шум, а практически - от единиц герц до десятков гигагерц. Такой измерительный сигнал позволяет исследовать устройство или систему одновременно во всем диапазоне рабочих частот. В реальных генераторах «белый» шум получить невозможно, но для любого устройства, полоса пропускания которого во много раз меньше спектра шумового сигнала, последний можно считать «белым».

Обобщенная структурная схема генератора шума (рис. 1.11) стоит из источника шума ИШ‚ широкополосного усилителя У и аттенюатора Ат [3]. Измеритель выхода позволяет контролировать уровень выходного сигнала в единицах напряжения или в единицах спектральной плотности мощности шума. К источнику шума предъявляются следующие требования: равномерность спектральной плотности мощности в заданной полосе частот; достаточное выходное напряжение (мощность) шумового сигнала; неизменность и воспроизводимость характеристик шума во времени и при изменении внешних влияний; заменяемость после истечения гарантийного срока работы без нарушения выходных параметров генератора. Наибольшее распространение в качестве источников шума получили резисторы, вакуумные и полупроводниковые диоды, фотоэлектронные умножители и газоразрядные лампы.



Рисунок 1.11 – Упрощенная структурная схема генератор шумовых сигналов

Шум, возникающий в резисторе, обусловлен хаотическим тепловым движением электронов, которое прекращается только при абсолютном нуле. Среднеквадратическое значение напряжения шумового сигнала резистора определяется следующей формулой:

, (1.7)

где — постоянная Больцмана; Т — температура, К; R—сопротивление резистора, Ом, при нормальной температуре Т0=290 К; — эквивалентная полоса пропускания, в которой определяется напряжение, Гц.

Если нагрузить шумящий резистор другим, равным ему по сопротивлению, то на втором резисторе выделится мощность:

 (1.8)

Отсюда можно определить спектральную плотность мощности шума:

 (1.9)

Спектральная плотность мощности шума резистора при нормаль­ной температуре равна S(f) = kT0 = 4.10-21 Вт/Гц. Произведение kT0 удобно использовать в качестве единицы спектральной плотности мощности. Например, 5kT0 означает, что температура шумящего резистора в пять раз выше нормальной и спектральная плотность равна 2**.**10-20 Вт/Гц.

Из выражения (1-7) можно найти сопротивление резистора:; отсюда следует, что активные элементы, в которых возникают шумы, можно замещать эквивалентным шумящим резистором, шумовое сопротивление Rш которого при нормальной температуре Т0 равно: .

Вакуумный диод, работающий в режиме насыщения, является источником шума вследствие случайного характера процесса термоэлектронной эмиссии.

Среднеквадратическое значение шумового тока диода определяется известным выражением ‚ где е — заряд электрона (); Is — ток насыщения А;  — полоса пропускания устройства‚ на вход которого поступает ток насыщения диода, Гц. Вакуумные диоды, например типа 2Д2С, генерируют шум в диапазоне частот 1—600 МГц. Напряжение и уровень спектральной плотности мощности на выходе генератора регулируется изменением тока накала диода.

В качестве источника шума широко используются полупроводниковые диоды; низкочастотные и высокочастотные, работающие в диапазоне 20 Гц— 20 кГц и 60—80 МГц соответственно. Последние часто используются и в низкочастотных генераторах шума (путем гетеродинного переноса частот).

Газоразрядные трубки являются источниками шума в диапазоне сверхвысоких частот — от 500 МГц до 12 ГГц. Шум обусловлен беспорядочным движением электронов в ионизированном газе (плазме). Под влиянием приложенного электрического поля они движутся с высокой скоростью, поэтому мощность шума достигает относительно больших значений. Спектральная плотность мощности равна kТе, где Те — «электронная температура», зависящая от состава газа и его давления. Значение Те достигает нескольких десятков тысяч кельвинов.

Рассмотрим особенности построения генераторов шумовых сигналов в зависимости от диапазона частот.

Низкочастотный генератор шума (Г2-47) строится по схеме прямого усиления шумовых сигналов, получаемых от полупроводникового диода в диапазоне 0-20 кГц. Усиление сигнала осуществляется транзисторными усилителями, между которыми включены полосовые фильтры, формирующие диапазоны частот 250-3500 Гц и 40-12000 Гц. Выходной усилитель мощности с переключаемой обратной связью обеспечивает выход сигнала на нагрузки 6,60 и 600 Ом. Предусмотрен ступенчатый аттенюатор до 100 дБ и вольтметр, шкала которого проградуирована в среднеквадратических значениях напряжения. Неравномерность спектра «белого» шума не более 2 дБ.

Низкочастотный генератор шума (Г2-З7), работающий в диапазоне (15 Гц—— 6,5 МГц), строится на принципе переноса спектра источника шума из области высоких частот в рабочий диапазон методом гетеродинирования. Источник шума — полупроводниковый диод 2Г401Б — вырабатывает шум в диапазоне частот до 80 МГц. Полосовой усилитель с полосой 63—77 МГц соединен со смесителем, на второй вход которого подано напряжение гетеродина, работающего на частоте 70 МГц. В результате на выходе смесителя получаются два сигнала разностных частот, лежащих выше и ниже частоты гетеродина. Частотный диапазон каждого из них 0-7 МГц. Оба сигнала суммируются и поступают на фильтры нижних частот формирующие рабочие полосы диапазонов 0-20 кГц, 0-600 кГц или 0-6‚5 МГц. Низкочастотные составляющие 0-15 ГЦ подавляются в последующем видеоусилителе, с выхода которого сигнал поступает на ступенчатый аттенюатор и вольтметр. Выходное сопротивление 50 и 600 Ом. Выходное напряжение регулируется в пределах 3 мкВ — 1 В плавно и ступенями через 10 дБ при внешней нагрузке не менее 10 кОм.

* + 1. **Генерация случайных процессов с нормальным распределением плотности вероятности на основе генератора М-последовательностей**

Для генерации М-последовательностей с одним элементом исключающее или получены таблицы подключений входов элемента к выходам Q0,...,QN-1 N-разрядного сдвигающего регистра, обеспечивающих получение псевдослучайной последовательности максимальной длины. Такая таблица приведена на рис. 1.12,а.

На рис. 1.12,б показана схема генератора псевдослучайного последовательности (ПСП) при N = 4. Результаты анализа состояний схемы как цифрового автомата сведены в таблицу на рис. 1.12,в. Для каждого текущего состояния дано значение сигнала на входе триггера D0 = Q2Q3, которое в результате поступления тактового импульса C в следующем состоянии фиксируется на выходе Q0. Остальные триггеры работают аналогично – происходит сдвиг кода D0Q0Q1Q2 в следующем состоянии на один разряд вправо. Всего существует 15 различных состояний регистра. Это максимальное число состояний для N = 4 с элементом Исключающее ИЛИ в цепи обратной связи. Следовательно, период ПСП равен 2N – 1, цифры ПСП повторяются через 2N – 1 тактовых импульсов.

Состояние 0000 не может существовать в регистре и в ПСП, так как попадание в него приводит к блокировке регистра. Состояние 0000 не может измениться, поскольку на вход D0 всегда будет подаваться 0. Для вывода регистра из состояния блокировки при включении питания или в результате сбоя можно использовать специальные сигналы начальной установки (стартовые сигналы). На рис. 1.12,б – это сигнал , поступающий на асинхронные входы принудительной установки триггеров в единичное состояние.

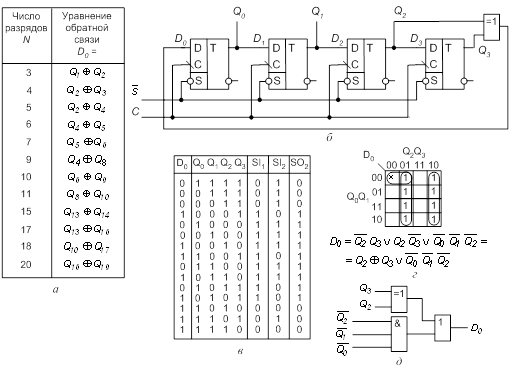


Рисунок 1.12 –Генерация случайных процессов с нормальным распределением плотности вероятности на основе генератора М-последовательностей

Другой способ вывода регистра из состояния блокировки – дополнение цепи обратной связи генератора (рис. 1.12,б) логической схемой самозапуска. Суть самозапуска генератора (рис. 1.12,б) выявляется при анализе карты Карно (рис. 1.12,г) для функции управления входом D0. Карта Карно составлена по таблице состояний генератора ПСП, в которой отсутствует состояние 0000. Поэтому в соответствующей клетке карты стоит знак факультативности , которому соответствует произвольное значение функции D0. Минимизация по единичным значениям функции D0 без включения факультативной клетки в единичные подклети соответствует доопределению функции D0 = 0 при текущем состоянии регистра 0000, следовательно, и следующим состоянием регистра будет 0000 – регистр заблокирован.

Чтобы не допустить этого, следует доопределить факультативное значение функции единицей, т.е. положить = 1. Таким образом, при состоянии регистра 0000 D0 = 1 (следующим его состоянием будет 1000) блокировка не происходит.

На рис. 1.12,г для такого варианта построения генератора ПСП приведена карта Карно для функции входа D0 . Минимальная дизъюнктивная нормальная форма (МДНФ) уравнения для этой функции (рис. 1.12,г) определяет структуру схемы обратной связи (рис. 1.12,д), обеспечивающей генератору ПСП свойство самозапуска.

Таблица рис. 1.12.в иллюстрирует эффект от использования генератора ПСП в схеме скремблера. Для примера взят исходный последовательный сигнал SI1, содержащий длинную серию единиц и подлежащий передаче по каналу связи. В результате скремблирования (перемешивания) на приемную сторону поступает сигнал SI2 = SO1, не содержащий длинных серий единиц, имеющий характер псевдослучайной последовательности. Сигнал SO2 на выходе дескремблера, полученный с использованием идентичного передающему генератора ПСП, полностью повторяет исходный сигнал SI1, т.е. SO2 = SI1.

* + 1. **Цифровые методы генерации случайных процессов с нормальным и равномерным распределениями плотности вероятности**

В настоящее время в тех или иных цифровой вычислительной машины (ЦВМ) имеется программа, оформленная в виде процедуры, предназначенная для генерирования случайных чисел с равномерной плотностью распределения вероятностей в диапазоне значений [0, 1] и которая часто носит название rnd(*x*).

Равномерная плотность распределения вероятностей в диапазоне значений [0, 1] записывается в виде *W* (ξ) = const = *C* . Вид этой плотности распределения вероятностей изображен на рис. 1.13.

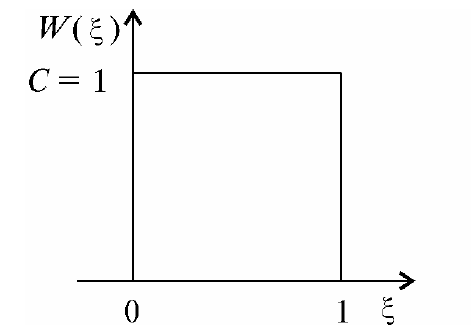


Рисунок 1.13 – Вид равномерной плотности распределения вероятностей псевдослучайных чисел

В силу того, что

 (1.10)

При этом генерируемые числа, получаемые обращением каждый раз к процедуре rnd( *x*) , имеют:

- математическое ожидание

 (1.11)

- диспепсию

 (1.12)

Для получения указанной равномерной плотности распределения вероятностей в ЦВМ используется ряд методов. Наиболее часто используют так называемый конгруэнтный метод, который состоит в том, что очередное случайное число ξ(*n*) получают из предыдущего числа ξ(*n* −1) с использованием следующего правила:

, (1.13)

где *p* –большое простое число, *A* –соответствующим образом выбранная константа.

При определенных значениях *A* по правилу (1.13) в случайном порядке генерируются целые числа в интервале от 1 до (p-1), причем их последовательность периодически повторяется.

Преимуществом данного метода является его простота и возможность использования в процессе вычислений памяти малого объема. Однако ему свойственна низкая скорость генерации, т.е. при каждой итерации приходится выполнять одно умножение (и обычно деление). Также для этого метода характерна большая чувствительность к значениям *A* и *p*.

Рассмотрим второй метод генерации случайных чисел [4]. На рис. 1.14 показана структурная схема генератора равномерно распределенных случайных чисел в диапазоне значений от –0,5 до +0,5.

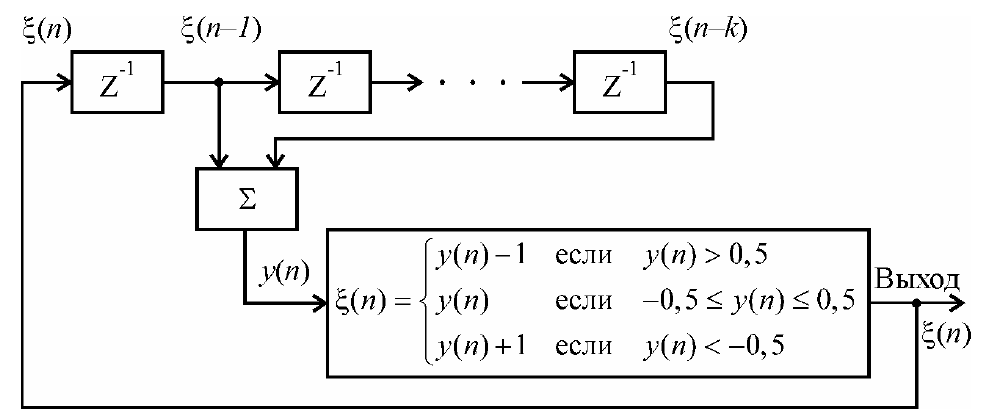
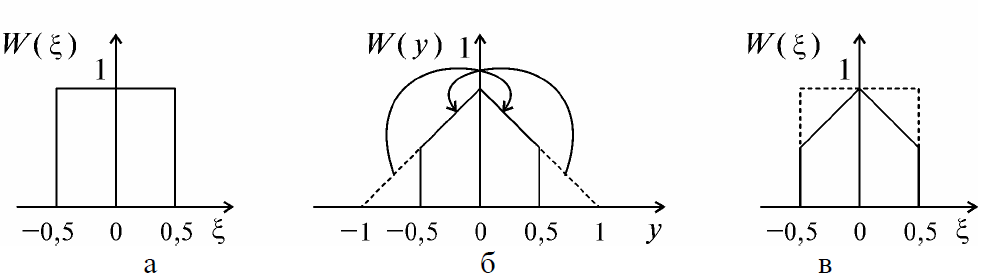


Рисунок 1.14 – Структурная схема генератора равномерно распределенных случайных чисел

Предполагается, что в *K* регистрах памяти содержатся предварительно записанные случайные числа в диапазоне значений величин от –,5 до +0,5 (их можно взять из таблиц случайных чисел). Новое случайное число ξ(*n*) генерируется по правилу:

 (1.14)

Выполнение операции по модулю 0,5 поясняется на рис. 1.15.

Рисунок 1.15 – Пояснение «наворачивания» плотности распределения вероятностей при переполнениях

На рис. 1.15,а) изображен требуемый вид плотности распределения вероятностей.

Треугольный вид плотности распределения вероятностей суммы двух случайных величин (рис. 1.15,б) следует из того, что она является результатом свертки их плотностей, каждая из которых имеет равномерное распределение.

Видно, что если результат сложения > 0,5 , то из него вычитается единица, если результат< −0,5 ,то единица добавляется. Поэтому случайная величина ξ(*n*) всегда будет находиться в пределах −0,5 ≤ ξ(*n*) ≤0,5.

В таком случае, если случайные величины ξ(*n* −1) и ξ(*n* −*k*) равномерно распределены на одном и том же интервале от −0,5 до +0,5 , то и случайная величина ξ(*n*) также будет распределена равномерно в этом же интервале.

Преобразование плотности распределения вероятностей треугольной формы в равномерное распределение получается после взятия суммы по (mod 0,5) (рис. 1.15,в).

Основное преимущество рассматриваемого метода состоит в том, что все вычисления в пределах итерации сводятся к одному сложению. Отметим, что для хранения чисел от ξ(*n*−1) и ξ(*n*−*k*) в генераторе равномерно распределенных случайных чисел требуется примерно 50 регистров.

Проведенные измерения плотности распределения вероятностей случайных чисел, получаемых при использовании рассмотренного генератора, показали, что она очень близка к равномерной.

Нормальный случайный процесс с 2q-мерной плотностью распределения вероятностей представляется совокупностью значений выборок, представляемых в виде последовательности отсчетов квадратурных составляющих этого процесса. Указанная 2q-мерная плотность распределения вероятностей нормального случайного процесса записывается в виде:

, (1.15)

где есть 2*q*-мерный вектор-столбец с компонентами *j*Re *x* и *j* Im *x* , которые соответствуют выборкам квадратурных составляющих; **M**2*q* –ar корреляционная матрица порядка 2*q*× 2*q* с элементами:

 (1.16)

где - вектор, равный математическому ожиданию вектора ; det**M**- определитель корреляционной матрицы **M**2*q.*

Здесь **M**2*q* –невырожденная матрица (detM2q≠ 0) и имеющая обратную, которая равна . Здесь I–единичная матрица размера (2*q* × 2*q*) .

Показатель экспоненты в (1.15) есть положительно определенная квадратичная форма:

 (1.17)

Конкретный вид матрицы М2q определяется энергетическим спектром генерируемого процесса.

В настоящее время наиболее экономичным методом генерации независимых случайных чисел с нормальным законом распределения плотности вероятностей является метод, основанный на свойстве сходимости сумм независимых случайных величин с произвольными плотностями распределения вероятностей.

При применении данного метода производится суммирование n независимых случайных величин. Эти величины являются выборками из совокупности случайных чисел, имеющих равномерную плотность распределения вероятностей с параметрами  и , определяемыми в режиме равномерных распределения. Следовательно, используется процедура rnd(x).

При этом получаются случайные независимые отсчеты с нормальной плотностью распределения вероятностей с параметрами:

 (1.18)

Когда для получения независимых выборок Xj нормального случайного процесса с M=0 и D= используется рассмотренный генератор случайных чисел с нормальной плотностью распределения вероятностей, то используется процедура rnd(x) и применяется выражение:

, (1.19)

где 

Для формирования квадратурных составляющих «белого» шума можно использовать один датчик случайных чисел, распределенных по нормальному закону. При этом положим, что сформированные случайные числа с нечетными номерами в их последовательности поступают в канал устройства обработки, который полагается действительным (реальным), а случайные числа с четными номерами – в канал устройства обработки, который полагается мнимым. На рис. 1.14 изображена структурная схема устройства цифрового формирования квадратурных отсчетов «белого» шума с параметрами N(0,).

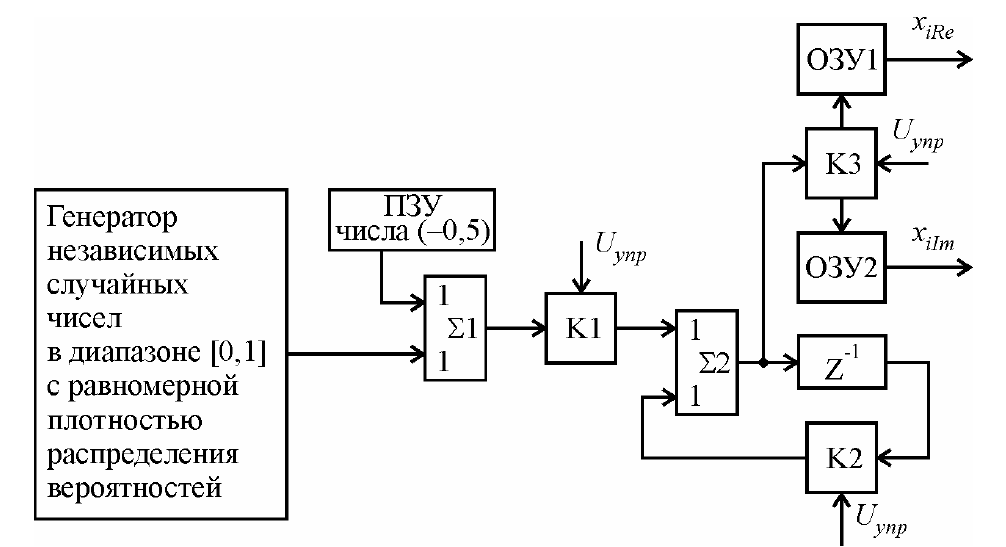


Рисунок 1.16 – Структурная схема генератора квадратурных составляющих «белого» шума

Работает генератор следующим образом. Положим, что до момента времени t = 0 в цифровой линии задержки (ЦЛЗ)  записан нуль, входы коммутаторов К1 и К2 соединены с их выходами. Коммутатор К3 имеет три состояния – когда его вход подключен к действительному выходу устройства через оперативное запоминающее устройства (ОЗУ) 1 к мнимому выходу через ОЗУ2 и когда оба выхода отключены от его входа. В момент времени К3 отключен от его входа. В момент времени t=0 вырабатывается k-разрядное случайное число, являющееся выборкой из совокупности случайных чисел с равномерной плотностью распределения вероятностей в диапазоне чисел [0,1]. Это число поступает на первый вход сумматора Σ1, на второй вход которого из постоянного запоминающего устройства (ПЗУ) поступает код числа -0,5. Это число вычитается из случайного числа в сумматоре Σ1 и полученный разностный результат через коммутатор К1 поступает на меньшие разряды одного из входов сумматора Σ2. На другой вход сумматора Σ2 через коммутатор К2 поступает нуль с выхода ЦЛЗ . Разрядность этой ЦЛЗ определяется из условия . Полученный результат суммирования записывается в ЦЛЗ .

В момент времени t = T, где Т – временной интервал, определяемый быстродействием генератора k-разрядных случайных чисел, второе число поступает на сумматор Σ1 и из него вычитается число 0,5, поступившее из ПЗУ. Полученный разностный результат поступает через коммутатор К1 на сумматор Σ2, на второй вход которого с выхода коммутатора К2 поступает результат, полученный ранее. Полученный второй результат суммирования в сумматоре Σ2 записывается в ЦЛЗ .

Далее работа устройства повторяется до тех пор, пока с генератора k разрядных случайных чисел в момент времени nT не поступит через коммутатор К1 случайное число на один из входов сумматора Σ1. Аналогично описанному ранее, на другой вход сумматора Σ1 поступит число -0,5 и полученный разностный результат через коммутатор К1 поступит на сумматор Σ2. Сумма (n-1) случайных чисел поступит с выхода ЦЛЗ  через коммутатор К2 на другой вход сумматора Σ2. Полученный результат суммирования с выхода сумматора Σ2 поступит на ЦЛЗ  и на вход коммутатора К3, который подключит вход ОЗУ1 к выходу сумматора Σ2. Полученное случайное число записывается в ОЗУ1.

Одновременно коммутаторы К1 и К2 разомкнутся, рекурсивная ячейка «обнулится» и подготовится к обработке следующей группы отсчетов.

В момент времени t = T + nT коммутаторы К1 и К2 замкнутся, а коммутатор К3 разомкнет свой выход с входом ОЗУ1, выработанное число в генератор k-разрядных случайных чисел поступит на вход сумматора Σ1. На другой вход сумматора Σ1 поступит из ПЗУ число -0,5, которое вычитается из выработанного в этом генераторе случайного числа, Полученный разностный результат поступает на вход сумматора Σ2, задругой вход которого поступает нулевое число через коммутатор К2 с выхода ЦЛЗ . Результат суммирования этих чисел в сумматоре Σ1 записывается в ЦЛЗ . Далее работа устройства повторяется до момента времени . В этот момент времени вырабатывается k-разрядное случайное число, из которого вычитается число -0,5 в сумматоре Σ1 и разностный результат поступает на вход сумматора Σ2. В сумматоре Σ2 происходит сложение с числом, полученным в результате суммирования (n-1) чисел на интервале времени nT<t<(2n-1)T. После суммирования результат с выхода сумматора Σ2 поступает в ЦЛЗ  и на подключенный вход коммутатора К3, выход которого подключен к ОЗУ2. Одновременно размыкаются коммутаторы К2 и К2 и к моменту времени t=2nT устройства готово к обработке следующей группы из n случайных k-разрядных чисел.

Одновременно два первых случайных числа  и  с выходов ОЗУ1 и ОЗУ2 параллельно поступают на выходы устройства. Эти числа представляют собой отсчеты квадратурных составляющих «белого» шума.

Далее процесс генерации следующей пары квадратурных составляющих «белого» шума повторяется. Видно, что интервал времени 2nT= определяется из условий выполнения теоремы Котельникова. Следовательно, узлы устройства генерации должны быть в 2n раз более быстродействующими относительно выбранной частоты дискретизации , где-ширина спектра полезного сообщения.

# РАЗРАБОТКА ЦИФРОВОГО ГЕНЕРАТОРА-ИЗМЕРИТЕЛЯ СИГНАЛОВ

* 1. **Функциональная схема устройства**

Проектируемый генератор-измеритель должен представлять собой переносной прибор с питанием от сети переменного тока общего пользования для работы в лабораторных условиях. Устройство должно генерировать сигналы заданных форм на выходах двух независимых аналоговых каналах. Форма и параметры сигналов для каждого из независимых каналов должны задаваться с помощью клавиатуры (КЛ) и отображаться на жидкокристаллическом индикаторе (ЖКИ). Также должно производиться измерение параметров генерируемых сигналов (ток, напряжение и мощность) и их запись на sd-карточку.

При этом можно определить функциональную схему устройства (рис. 2.1).



Рисунок 2.1 – Функциональная схема цифрового генератора, где

CPU– цифровое вычислительное устройство; Keyboard – клавиатура; DAC – цифро-аналоговый преобразователь; IF– интерполирующий фильтр; Buffer amplifier – буферный усилитель, Current meter – измеритель тока

Центральным элементом системы является ЦВУ, в памяти программ которого хранится программа, реализующая определенную форму сигнала на выходе, с заданной амплитудой и частотой. Амплитуда, частота и форма сигнала вводятся с клавиатуры и отображаются на ЖКИ. ЦАП для преобразования цифрового (двоичного) кода в аналоговый сигнал. На выходах нужно два независимых канала, поэтому используется два канала ЦАП.

На выходе каждого канала ЦАП формируется свой аналого-дискретный сигнал. Рассмотрим 3 вида сигналов, из числа заданных в техническом задании:

– непрерывный гармонический сигнал;

– прямоугольные периодические импульсы;

– пилообразные периодические импульсы;

Все три сигнала имеют максимальную частоту 100 кГц, то есть минимальная продолжительность одного периода сигнала равна 10 мкс.

Исходя из этих данных, я моделировал (смотри приложение А) в среде MATLAB прямоугольные и пилообразные периодические импульсы на выходе ЦАП, потому что эти периодические сигналы будут иметь ширину спектра больше чем гармонический сигнал. Результаты показаны на рис. 2.2–2.5.



Рисунок 2.2 – Периодические импульсы со скважностью 5% (1) и 50% (2)



Рисунок 2.3 – Спектр прямоугольных периодических импульсов со скважностью 5% (1) и 50% (2)



Рисунок 2.4 – Пилообразные периодические импульсы



Рисунок 2.5 – Спектр пилообразных периодических импульсов

Спектр конечной последовательности импульсов бесконечный, поэтому в нем нельзя определить конечную верхнюю граничную частоту. Но мы определяем верхнюю граничную частоту из «эффективной» ширины спектра. Из рис. 2.3, 2.5 получаем максимальную верхнюю граничную частоту в спектре генерируемых сигналов  (первые 3 «нуля» функции sinc, которая представляет огибающую спектра прямоугольных периодических импульсов).

Однако по ТЗ поставлена задача пропускать простое гармоническое колебание с частотой до 30 кГц, поэтому этой частотой, как верхней частотой в спектрах генерируемых сигналов, мы и ограничимся.

Спектр аналого-дискретного сигнала в общем случае представляет собой периодическую функцию по частоте (рис. 2.6).



Рисунок 2.6 – Спектры сигналов на выходах ЦАП и IF

Выделение спектра аналогового сигнала из спектра аналого-дискретного сигнала осуществляется с помощью интерполирующего фильтра. ИФ – это фильтр нижней частоты.

Разрабатываемый генератор может использоваться как источник сигнала для других устройств. Входное сопротивление этих устройств может быть маленьким, и оно может влиять на работу выходных каскадов генератора. Поэтому амплитуда сигнала на входе получателя сигнала может не соответствовать той, которая установлена в генераторе. Для согласования выхода генератора (источника сигнала) с входом получателя используется БУ.

* 1. **Расчет характеристик и выбор элементов структурной схемы**
     1. **Расчет параметров цифро-аналогового преобразователя**

Для выбора цифро-аналогового преобразователя, нужно вычислить следующие параметры: разрядность и требуемое быстродействие.

Разрядность определяет количество различных уровней выходного сигнала, которые ЦАП может воспроизвести. Напряжение на выходе ЦАП должно задаваться в диапазоне 0В…+5 В с дискретностью . Точность установки и поддержания параметров сигналов согласно технического задания – , можно определить величину уровня квантования:

. (2.1)

Поэтому количество уровней квантования равняется:

, (2.2)

а количество разрядов ЦАП равно .

Требуемое быстродействие определяется требуемой частотой дискретизации сигналов. Согласно теореме Котельникова (известной также как теорема Шеннона-Найквиста) частота дискретизации сигналов должна быть: . Для упрощения интерполирующего фильтра частоту дискретизации выбираем с запасом .

Таким образом, чтобы выбирать ЦАП, нужно рассматривать следующие параметры ЦАП:

1. разрядность ≥ 12 бит;
2. быстродействие ≥ 0,1 MSPS;
3. число каналов ≥ 2;
4. цена: как можно дешевле.

Параметры некоторых ЦАП показаны в таблице 2.1:

Таблица 2.1 – Сравнения различных типов ЦАП

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тип ЦАП | Разрядность, (бит) | Быстродействие, (MSPS) | Количество каналов | Интерфейс | Цена, (грн) | Производитель |
| AD9765 | 12 | 125 | 2 | параллельный | 115.52 | AnalogDevices |
| AD9767 | 14 | 125 | 2 | параллельный | 114.32 | AnalogDevices |
| MAX5873 | 12 | 200 | 2 | параллельный | 81.28 | Maxim |
| MAX5874 | 14 | 200 | 2 | параллельный | 106.00 | Maxim |
| DAC5662 | 12 | 275 | 2 | параллельный | 114.00 | TEXAS INSTRUMENTS |
| DAC2902 | 12 | 125 | 2 | параллельный | 225.52 | TEXAS INSTRUMENTS |
| STM32F103 (inner DAC) | 12 | 0.25 | 2 | - | - | ST microelectronics |

Из таблицы 2.1 оптимальным по указанным выше параметрам является STM32F103 (inner DAC).

* + 1. **Выбор жидкокристаллического индикатора**

Для выбора ЖКИ, нам нужно определить какой текст может отображаться на ЖКИ для всех сигналов, заданных в ТЗ. Во-первых, текст отображается на ЖКИ при генерации сигналов двух каналов. Нам нужно показать амплитуду, смещение, частоту, скважность (для импульса) сигнала в каждом канале. В таблице 2.2 показана максимальная длина символов для каждого из параметров.

Из таблицы 2.2 можно получить максимальное количество символов, которые нужно отображать на ЖКИ: Nmax = (9+8+8+9+8)\*2 = 84. Среди существующих стандартных форматов ЖКИ-модулей (символов x строк): 8x2, 16x1, 16x2, 16x4, 20x1, 20x2, 20x4, 24x2, 40x2, 40x4, подходящими являются форматы: 20x4, 40x2 и 40x4. Последний слишком большой, поэтому его отбросим. Поскольку не все параметры будут присутствовать на ЖКИ одновременно, выбор остановился на ЖКИ с размерами 20x4, как на более удобном, с точки зрения визуального восприятия (горизонтальное расположение).

Таблица 2.2 – Максимальное количество символов для отображения параметров

|  |  |  |
| --- | --- | --- |
| Параметр | Примеры символов для отображения на ЖКИ | Максимальное количество символов |
| Номер канала и форма сигнала в канале | +++Ch1+++ | 9 |
| SF:sawtoo |
| Амплитуда | А = 1.5В | 8 |
| Смещение | О = 1.0В | 8 |
| Частота | F=99999Hz | 9 |
| Скважность | D = 10% | 8 |
| D = 100% |

На рис. 2.7 и рис. 2.8 показаны две формы примера текста для ЖКИ-модулей.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | + | + | C | h | 1 | + | + | + |  |  |  |  |  | C | h | 2 |  |  |  |
| S | F | : | s | a | w | t | o | o |  |  | S | F | : | p | u | l | s | e |  |
| A |  | = |  | 1 | . | 0 | В |  |  |  | F | = | 9 | 9 | 9 | 9 | 9 | H | z |
| O |  | = |  | 2 | . | 3 | В |  |  |  | D |  | = |  |  | 5 | 0 | % |  |

Рисунок 2.7 – Пример отображения текста на ЖКИ-модуле 20x4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| KК | 11 | :: | ss | ii | nn |  | AA | -= | 55 | /. | 00 | ВB |  | FF | == | 99 | 99 | 99 | 99 | HH | zz |  | OО | == | 11 | /. | 44 | ВВ | В |  |  |  |  |  |  |  |  |
| KК | 12 | :: | зp | uu | nl | ss | ee |  | AA | -= | 55 | /. | 00 | ВB |  | FF | == | 99 | 99 | 99 | 99 | HH | zz |  | OО | == | 11 | /. | 44 | ВВ | A | DQ | == | 00 | .. | 00 | 55 |

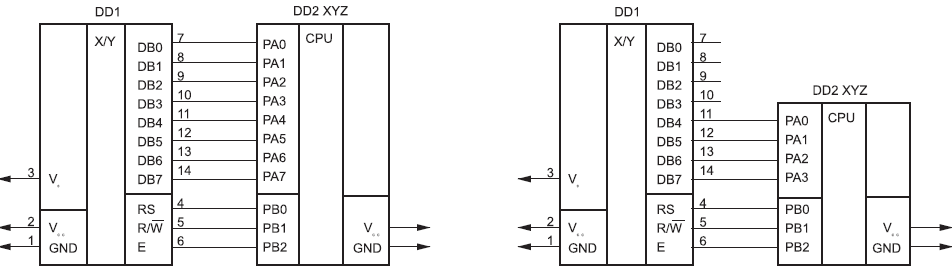
Рисунок 2.8 – Пример отображения текста на ЖКИ-модуле 40x2

Я выбрал формат 20x4, потому что на ЖКИ этого формата удобно различать текст. При этом была выбрана модель ЖКИ WH2004A [5](Winstar Display Co., Ltd.). Назначение его выводов показанj в таблице 2.3.

Таблица 2.3 – Назначение выводов символьного ЖКИ типа WH4002A

|  |  |  |
| --- | --- | --- |
| Вывод | Обозначение | Функция |
| 1 | Vss | GND |
| 2 | Vdd | +3V or +5V |
| 3 | Vo | Contrast adjustment |
| 4 | RS | H/L Register select signal |
| 5 | R/W | H/L Data Read/Write |
| 6 | E | H→L Enable signal |
| 7-14 | DB0-7 | H/L Data Bus Line |
| 15 | A/Vee | +4.2V for LED / Negative Voltage output |
| 16 | K | Power Supply for B/L(0V) |

Для соединения модуля с управляющей системой можно выбрать один из двух вариантов: по 8-ми или 4-х разрядной шине. В первом случае потребуется 11 сигнальных линии, во втором - только 7 (еще 3 линии: линия выбора операции R/W, линия выбора регистра RS, линия стробирования/синхронизации Е). На рис. 2.9,а приведена схема подключения ЖКИ-модуля с 8-ми разрядной шиной к ЦВУ XYZ. Это ЦВУ содержит два порта: 8-ми разрядный двунаправленный PA0...PA7, к которому подключена шина DB0...DB7 ЖКИ-модуля, и 3-х разрядный PB0...PB2, к которому подключены линии управляющих сигналов: E, RS, R/W. На рис. 2.9,б можно видеть схему подключения ЖКИ-модуля к этому же ЦВУ в 4-х разрядном режиме, в этом режиме используется старшая тетрада шины данных - DB4...DB7.



а) б)

Рисунок 2.9 – Подключение ЖКИ к управляющему ЦВУ

В 4-х разрядном режиме немного усложняются операции обмена данными с ЖКИ-модулем (байт данных передается последовательно, по тетрадам). Реализации этих операций - единственное, что отличает процесс обмена по 8-ми разрядной шине от обмена по 4-х разрядной шине. Чтобы уменьшить количество используемых линий ввода/вывода ЦВУ, можно использовать подключение ЖКИ-модуля с 4-х разрядной шиной.

* + 1. **Подключение клавиш управления**

По ТЗ требуется подключение к ЦВУ восьми клавиш управления режимами работы и выбора параметров сигналов генератора:

1) «Channel» - выбор канала генерации для редактирования параметров сигнала;

2) «Signal Form» - выбор формы сигнала;

3) «Amplitude» - амплитуда (или среднеквадратическое отклонение) сигнала;

4) «Offset» - смещение (или мат. ожидание) сигнала;

5) «Frequency» - частота сигнала;

6) «Duty» - скважность импульса;

7) «+» - увеличение;

8) «-» - уменьшение.

При небольшом количестве (N ≤ 8) подключаемых клавиш можно воспользоваться одним из аппаратных способов подавления дребезга [6]. Используем способ подавления дребезга с помощью простейшего фильтра нижних частот (ФНЧ) (RC-цепочки), так как он требует меньших аппаратных затрат. Схема подключения клавиш с аппаратным подавлением дребезга контактов к ЦВУ показана на рис. 2.10.

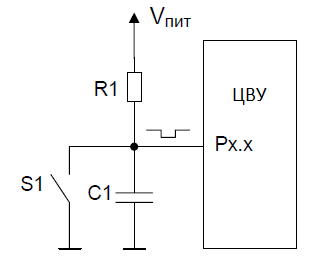


Рисунок 2.10 – Подключение клавиш к ЦВУ с использованием RC-цепочки

Таким образом, количество линий ввода/вывода для подключения клавиатур к ЦВУ равно 8. Суммарное количество выводов ЦВУ для подключения ЖКИ и клавиатур Ns = 7 + 8 = 15.

* + 1. **Выбор вычислительного устройства**

С целью упрощения устройства и уменьшения его цены можно выбрать в качестве ЦВУ микроконтроллер (МК). Выбор осуществляется по таким критериям:

1) внутренний ЦАП (параметры 0,25 MIPS x 12 bit);

2) количество цифровых линий ввода/вывода – больше 154;

3) цена – как можно меньше.

Параметры некоторых распространенных микроконтроллеров с внутренним ЦАП показаны в таблице 2.4. Я рассматривал 5 возможных семейств МК: STM32, ST7...ST9, AVR, MSP430, PIC.

Таблица 2.4 – Параметры некоторых МК

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Модель МК | Такто- вая частота(MГц) | Объем ПЗУ, (кБ) | Быстро- действие ЦАП (MSPS) | Разряд-ность ЦАП (бит) | Количество линий вв./выв. | Напряже-ние питания (В) | Цена, (грн) |
| STM32F103RCT6 | 72 | 32 | 0,25 | 32 | 51 | 2 – 3,6 | 103,20 |
| AT32UC3C0512C | 66 | 16 | 0,4 | 32 | 123 | 4,5 – 5,5 | 105,76 |
| MSP430FG439IPN | 8 | 2 | 0,1 | 16 | 48 | 1,8 – 3,6 | 131,84 |

Но для серии MSP430 скорость действия ЦАП меньше 0.25 MSPS, а у ST7...ST9, PIC нет внутренних ЦАП. Поэтому я выбрал модель МК STM32F103RCT6 [7], так как он имеет меньшую цену, а также большее быстродействие чем МК семейства AVR.

* + 1. **Выбор интерполирующего фильтра**

Для преобразования аналого-дискретного сигнала на выходе ЦАП в аналоговый сигнал мы используем ФНЧ как интерполирующий фильтр. Важное условие для ИФ – он не должен искажать форму аналогового сигнала.

Среди активных аналоговых фильтров различают такие основные типы:

– Фильтр Баттерворта;

– Фильтр Чебышева;

– Фильтр Бесселя;

Фильтр Баттерворта обеспечивает наиболее плоскую характеристику в полосе пропускания, что достигается ценой плавности характеристики в переходной области, т. е между полосами пропускания и задерживания. Активные фильтры, построенные из элементов, номиналы которых имеют некоторый допуск, будут обладать характеристикой, отличающейся от расчетной, а это значит, что в действительности на характеристике фильтра Баттерворта всегда будет иметь место некоторая неравномерность в полосе пропускания. Поэтому весьма рациональной структурой является фильтр Чебышева. Его характеристика в области перехода имеет большую крутизну, но в полосе пропускания (или задерживания) распределено несколько равновеликих выбросов АЧХ. Отличие фильтра Бесселя от других типов фильтров – линейная фазо-частотная характеристика. Но за постоянство времен запаздывания у фильтра Бесселя приходится расплачиваться тем, что его амплитудно–частотная характеристика имеет еще более пологий переходной участок между полосами пропускания и задерживания, чем даже у характеристики фильтра Баттерворта [8].

Исходя из этих данных, я моделировал, в среде Simulink, импульсный ответ трех различных фильтров (на вход фильтров разных типов подавал одинаковый сигнал - прямоугольный импульс длительностью 5мкс). Результаты показаны на рис. 2.12.

Из рис. 2.12 видно, фильтр Бесселя все же обладает лучшими свойствами во временной области по сравнению с фильтрами Баттерворта и Чебышева (наилучше сохраняет форму сигнала). Сам фильтр Чебышева при его весьма крутой амплитудно–частотной характеристике имеет наихудшие параметры во временной области из всех этих трех типов фильтров. Фильтр Баттерворта дает компромисс между временными и частотными характеристиками. Поэтому я выбрал фильтр Бесселя как ИФ.



Рисунок 2.12 – Импульсный ответ через три различных фильтров

Дальше нужно определить порядок фильтра Бесселя. Крутизна АЧХ фильтра Бесселя 6\*n дБ/октаву, где n – порядок фильтра. По ТЗ требуется подавление на частоте 2fв не меньше 20дБ. Поэтому мы выбираем фильтр типа Бесселя шестого порядка.

* + 1. **Общая структурная схема генератора**

**Интерполирующий фильтр**. ЦАП, по сути, устройство, которое осуществляет интерполяцию, поэтому сигнал, полученный после него, необходимо пропустить через ФНЧ. Для получения аналогового сигнала из дискретного.

**Униполярный усилитель.** Необходим для обеспечения необходимой мощности, выдаваемой генератором-измерителем.

**Измеритель тока.** Используется для измерения тока, потребляемого нагрузкой, подключенной к выходу генератора-измерителя.

**Источник питания.** Источниками питания называют устройства, предназначенные для снабжения электронной аппаратуры электрической энергией и представляющие собой комплекс приборов и аппаратов, которые вырабатывают электрическую энергию и преобразуют её к виду, необходимому для нормальной работы каждого узла электронной аппаратуры. Электрическая энергия, вырабатываемая первичными источниками, не всегда может быть непосредственно использована для питания электронной аппаратуры, поэтому необходимым элементом является источник вторичного электропитания – устройство, в котором происходит преобразование одного вида электрической энергии в другой.

Если источник первичного питания создаёт переменное напряжение, а прибор должен питаться постоянным напряжением, то основными узлами источника вторичного питания является: выпрямитель, сглаживающий фильтр, стабилизатор выходного напряжения (рис. 2.13).



Рисунок 2.13 – Структурная схема источники вторичного питания

Мы уже знаем, что напряжение питания ЦВУ равно 2.0 - 3.6 В (для определенности выберем стандартное номинальное значение +3,3 В), напряжение питания ЖКИ равно 5 В, а операционных усилителей 12,5 В. Для преобразования переменного сетевого напряжения 220 В на постоянное +3,3, +5,0 В и 12,5 В нужно использовать источник вторичного питания (блок питания). Так как по заданию источник первичного питания (сеть) создаёт переменное напряжение 220 В, то основными узлами схемы будут преобразователь уровня переменного напряжения, выпрямитель, сглаживающий фильтр и стабилизатор выходного напряжения.

Для выбора трансформатора нужно посчитать максимальную потребляемую мощность устройства, для этого посчитать суммарные максимальные токи потребления каждой микросхемы для каждого напряжения питания:

максимальный ток потребления ЖКИ: I1 = 8 мА (U = 5V);

максимальный ток потребления МК: I2 = 150 мА (U = 3.3V);

максимальный ток потребления ОУ (LM6142): I3 = 50 мА (U = 3.3V).

Получаемая максимальная потребляемая мощность устройства равна:

Pmax = U\*IΣ = 0.008\*5 + (0.15 + (0.05\*4)) \* 3.3 ≈ 1.2 Вт.

При этом структурная схема цифрового генератора определяется таким образом:

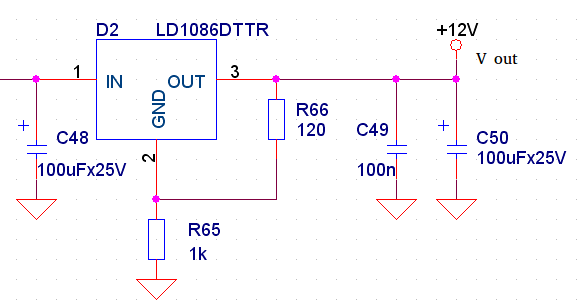


Рисунок 2.14 – Структурная схема цифрового генератора

* 1. **Расчет элементов принципиальной схемы генератора**
     1. **Расчет схемы стабилизации 12 В**

Схема стабилизации построена на стабилизаторе напряжения LD1086DTTR.

Схема включения LD1086DTTR:



где C48, C50 и R66 взяты из даташита на LD1086DTTR, а R65 посчитан исходя из того, что на выходе необходимо получить напряжение (V out) 12 В.

Формула для расчета R65:

R65 = R66 \* ( V out / V ref – 1),

исходя из:

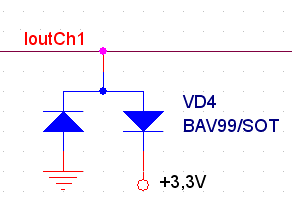
V out = V ref \* ( 1 + R65 / R66 ) (взята из даташита на LD1086DTTR),

где V ref –

* + 1. **Расчет схемы защищающей ножки контроллера**

Схема защиты построена на переключающемся диоде BAV99.

Схема включения BAV99:



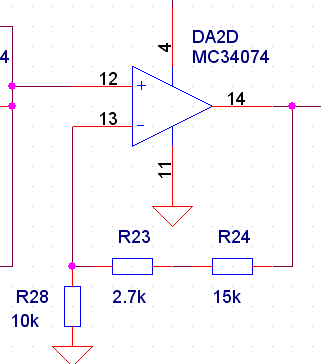
где IoutCh1 – ножка контроллера.

Напряжение 3.3 В выбрано поскольку необходимо защитить ножку от подачи напряжения больше чем 3.3 В. Когда напряжение на диоде становится больше чем 3.3 В, один из диодов открывается, через него начинает протекать ток, а значит на нем падает часть напряжения, которая является разницей между напряжением приложенным к диоду и 3.3 В. Другой диод служит для этой же цели, только для отрицательных напряжений.

* + 1. **Расчет схемы инвертирующего усилителя**

Схема построена на операционном усилителе MC34074. Данный усилитель имеет коэффициент усиления чуть больше единицы и предназначен для корректировки коэффициента передачи интерполирующего фильтра, построенного на операционных усилителях.

Схема включения операционного усилителя MC34074:

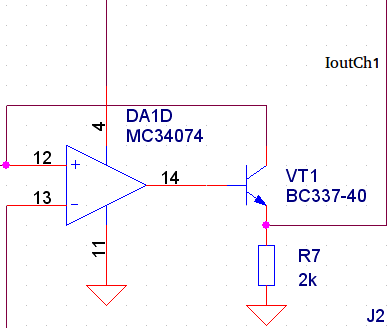
****

R23 и R24 выбраны экспериментально, для обеспечения коэффициента передачи цепочки фильтр - не инвертирующий усилитель равным 1.

* + 1. **Расчет схемы измерителя тока**

Измеритель тока построен на операционном усилителе MC34074.

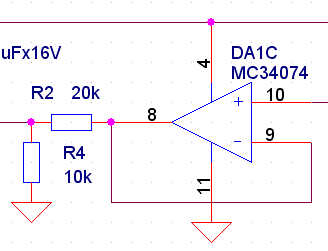
Схема включения измерителя тока:



* + 1. **Расчет схемы не инвертирующего усилитель с резистивным мостом (делителем напряжения)**

Данный усилитель построен на операционном усилителе MC34074.

Схема включения данного усилителя:



где R2 и R4 являются делителем напряжения. Их значения выбирались из расчета ограничения максимального напряжения на выходе этого делителя — 3.3 В (выход делителя соединен со входом АЦП контроллера).

* + 1. **Расчет интерполирующего фильтра**

Фильтр на источнике напряжения управляемом напряжением (ИНУН), известный также просто как фильтр с управляемым источником. В этом случае повторитель с единичным коэффициентом усиления заменен не инвертирующим усилителем с коэффициентом усиления, большим 1. На рис. 2.15 приведена схема для реализации фильтра нижних частот на ИНУН. С помощью присоединенных к выходу ОУ резисторов, образован не инвертирующий усилитель напряжения с коэффициентом усиления К, а остальные R и C формируют частотную характеристику. Такой двухполюсный фильтр может быть фильтром Баттерворта, Бесселя или Чебышева за счет определенного выбора параметров элементов. Любое число двухполюсных секций на ИНУН может быть соединено каскадно для создания фильтров более высокого порядка.



Рисунок – 2.15 Схема активных фильтров нижних частот на ИНУН

Для конструирования n-полюсного фильтра (при четном n) нужно соединить каскадно n/2 секций на ИНУН. Чтобы построить 6-полюсный фильтр Бесселя нижних частот, мы соединяем каскадно три вышеописанные секции с коэффициентами 1.040, 1.364 и 2.023, принципиальная схема показана на рис. 2.16.

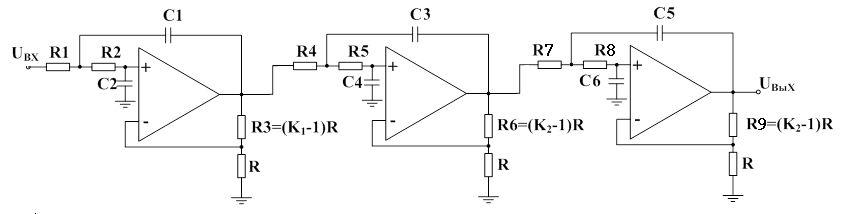


Рисунок – 2.16 Принципиальная схема 6-полюсного ФНЧ Бесселя на ИНУН

В каждой секции R1 = R2 = R и С1= С2 =С. Но произведение RC будет для каждой секции свое и должно вычисляться с помощью нормирующего множителя *fn* (его значения для каждой секции приведены в табл. 2.5) по формуле RC=1/2π*fcfn.* [8]. Здесь через *fc* обозначена точка, отвечающая значению -3 дБ.

Таблица 2.5 – Вспомогательные параметры для расчета ИНУН-фильтров Бесселя нижних частот

|  |  |  |
| --- | --- | --- |
| n | *fn* | К |
| 2 | 1.274 | 1.268 |
| 4 | 1.432 | 1.084 |
| 1.606 | 1.759 |
| 6 | 1.607 | 1.040 |
| 1.692 | 1.364 |
| 1.908 | 2.023 |

Как обычно в схемах на операционных усилителях, значение R выбирается в диапазоне от 10 до 100 кОм. Я выбрал R1=R2=R4=R5=R7=R8=R=10 кОм, и так как я уже получил *fc*=30 КГц, можно вычислить:

 (2.3)

 (2.4)

,

 (2.5)

 (2.6)

и

 (2.7)

 (2.8)

Для всех рассчитанных значений резисторов и конденсаторов нужно указывать номинальные значения. Они выбираются как наиболее близкие к значениям номинальных ряда Е3..Е24. Поэтому я выбрал С1 = С2 = 330.13 пФ, C3 = C4 = 313.54 пФ, C5 = C6 = 1,39, R3 = 400 Ом, R6 = 3640 Ом и R9 = 10,23 кОм. В качестве ОУ для фильтров можем выбрать ИМС MC34074 []. У них частота единичного усиления минимум в 2,5 раза выше чем fв = 6МГц.

* + 1. **Выбор буферного усилители**

Буферный усилитель – это усилительный каскад который имеет единичный коэффициент усиления (повторитель). Наиболее часто за основу их проектирования используют схему не инвертирующего усилителя без входного резистивного делителя, что обеспечивает очень большое входное сопротивление. Наиболее просто реализуется схема повторителя на рис. 2.20. Здесь R1=R2=10 кОм.



Рисунок 2.20 – Принципиальная схема буферного усилители

* 1. **Проектирование печатного узла**

После выполнения схемы цифрового генератора в OrCAD Capture, мы можем получить печатный плат (ПП) в OrCAD Layout.

Для создания новой ПП в OrCAD Layout нужно выполнить команду File/New. Система в диалоговых окнах запросит имя файла шаблона печатной платы (\*.tch), имя файла списка соединений (\*.mnl) и файла создаваемой печатной платы (\*.max).

Шаблоны печатной платы имеют расширения .tch или .tpl. Они содержат начальную информацию о печатной плате. В процессе загрузки списка соединений OrCAD ставит в соответствие каждому компоненту его корпус. Если в процессе загрузки списка соединений обнаружен компонент, которой не имеет ссылок на корпус, то выводится диалоговое окно для его определения. После завершения загрузки списка соединений в рабочем поле окна OrCAD Layout появятся изображения корпусов компонентов со связями между ними.

Перед размещением компонентов необходимо задать контур печатной ПП (Board outline). Контур создается командой Tool/Obstacle/New, с помощью которой вычерчивается замкнутый многоугольник. Толщина линий контура, тип линии (Board outline) задается в диалоговом окне. Для вызова окна свойств необходимо перед нанесением границы правой кнопкой мыши вызвать контекстное меню и там определить свойства. Эта линия должна размещаться на слое Global Layer.

Затем с помощью команды Option/System Setttings нужно установить требуемое значение шага размещения корпусов (поле Place grid).

Размещение компонентов в OrCAD Layout может выполняться либо вручную либо автоматически. Для ручного размещения необходимо выполнить команду Tool/Component, выбрать курсором требуемый корпус и переместить в соответствующее место контура платы.

Перед трассировкой необходимо выполнить подготовительные действия:

1. С помощью команды Option/System Setting устанавливаются требуемая сетка трассировки (Routing grid), сетка размещения барьеров и текста (Detail grid) и сетка переходных отверстий (Via grid).
2. С помощью команды Tool/Layer/Select from Spreadsheet или View/Database Spreadsheets/Layers просматривается и редактируется структура слоев проекта. Редактирование осуществляется двойным щелчком по соответствующему полю.



Рисунок 2.23 – Топология печатного узла клавиатуры (1)

и основной части генератора (2)

Трассировка проводников в пакете OrСАD может выполняться в ручном и автоматическом режиме. Трассировка проводников в ручном режиме осуществляется с помощью команд Tools/Track, Tool/Track Segment и Tool/Via. Трассировка в ручном режиме должна проводиться для высокочастотных целей, критических целей и корректировки печатных плат после трассировки в автоматическом режиме. Изменение направления проводника фиксируется щелчком левой кнопкой мыши. Для перехода на другой слой можно вызвать контекстное меню, выбрать пункт Add via, сделать активным требуемый слой и продолжать проведение трассы в этом слое.

Топология печатного узла клавиатуры и основной части генератора показаны на рис. 2.23.

* 1. **Разработка программного обеспечения цифрового устройства генератора**
     1. **Выбор среды и языка программирования для разработки программного обеспечения для МК**

Для большинства микроконтроллеров существуют низкоуровневый – ассемблер и высокоуровневый – Си языки программирования. В этих языках реализованы все необходимые возможности по управлению аппаратными средствами микроконтроллеров.

Ассемблер – это низкоуровневый язык программирования, использующий непосредственный набор инструкций микроконтроллера. Создание программы на этом языке требует хорошего знания системы команд программируемого чипа и достаточного времени на разработку программы. Ассемблер проигрывает Си в скорости и удобстве разработки программ, но имеет заметные преимущества в размере конечного исполняемого кода, а соответственно, и скорости его выполнения.

Си позволяет создавать программы, предоставляя разработчику все преимущества языка высокого уровня. Компиляция исходных текстов, написанных на Си, осуществляется быстро и дает компактный, эффективный код.  Основные преимущества Си перед ассемблером: высокая скорость разработки программ; универсальность, не требующая досконального изучения архитектуры микроконтроллера; лучшая документируемость и читаемость алгоритма; наличие библиотек функций; поддержка вычислений с плавающей точкой.

В языке Си гармонично сочетаются возможности программирования низкого уровня со свойствами языка высокого уровня. Возможность низкоуровневого программирования позволяет легко оперировать непосредственно аппаратными средствами, а свойства языка высокого уровня позволяют создавать легко читаемый и модифицируемый программный код. Кроме того, практически все компиляторы Си имеют возможность выполнения и занимаемым ресурсам участков программы.

Чтобы преобразовать исходный текст программы в файл прошивки микроконтроллера, применяют компиляторы. В настоящее время представлено достаточно много компиляторов Си для МК STM32. Самым мощным из них считается компилятор фирмы IAR Systems. IAR C Compiler имеет широкие возможности по оптимизации кода и поставляется в составе интегрированной среды разработки IAR Embedded Workbench (EWB), включающей в себя также компилятор ассемблера, линкер, менеджер проектов и библиотек, а также отладчик.

* + 1. **Описание программы для генератора-измерителя НЧ сигналов**

Исходя из ТЗ, программа должна выполнять следующие действия:

* опрос клавиатуры
* обеспечение реакции на события от клавиатуры
* вывод информации на LCD
* вычисление отсчетов генерируемых сигналов (на двух каналах)
* измерение напряжений на соответствующих выводах контроллера
* запись измеренных значений на sd карточку
* вывод усредненного значения тока, потребляемых нагрузками, напряжения питания нагрузок и, потребляемые нагрузками, мощности на экран.
* выдача звуковых сигналов

Ввиду большого количества задач, которые необходимо выполнять параллельно, в целях облегчения разработки и в целях изучения было решено использовать операционную систему реального времени FreeRTOS. Конечно, это не операционная система общего назначения, как UNIX-подобные системы, она не имеет виртуальной памяти, файловой системы и значительно уступает операционным системам в плане безопасности и универсальности, однако для данной задачи она подходит.

FreeRTOS – операционная система (далее ОС) реального времени (Free Real Time Operation System) позволяет, довольно легко, делить большую задачу на совокупность маленьких независимых задач (потоков) и предоставляет средства синхронизации и управления ими. Под средствами синхронизации имеются ввиду такие объекты ОС: семафоры, мютексы и сообщения. Переключение аппаратного контекста и стека задач ОС выполняет сама, освобождая от этого разработчика.

Исходя из списка задач, которые должна решать программа генератора-измерителя НЧ сигналов составим список потоков для FreeRTOS:

Measures printing thread – занимается выводом измеренных значений тока, потребляемых нагрузками, напряжения питания нагрузок и, потребляемые нагрузками, мощности на экран.

LCD thread – является промежуточным слоем (“драйвером”) между другими потоками и LCD аппаратурой.

Menu thread – реализует соответствующую реакцию на события от потока опроса клавиатуры, т. е. меню. Рассылает задания (сообщения) остальным потокам.

Measuring thread – является промежуточным слоем (“драйвером”) между другими потоками и аппаратурой занятой измерениями напряжений.

Oscillation thread – занимается генерацией указанных (из Menu thread) сигналов на двух каналах.

Polling button thread – является промежуточным слоем (“драйвером”) между другими потоками и аппаратурой занятой опросом клавиатуры.

SD service thread – является промежуточным слоем (“драйвером”) между другими потоками и аппаратурой занятой работой с sd-карточкой.

Beep thread – является промежуточным слоем (“драйвером”) между другими потоками и аппаратурой занятой работой с пьезоэлементом.

“Сердцем” программы является поток Menu thread. Этому потоку посылается сообщение от потока Polling button thread, если есть какое-нибудь событие (кнопка нажата, отжата, зажата). Основываясь на текущем состоянии программы, поток Menu thread может посылать сообщения другим потокам: о необходимости включить/выключить измерение напряжений, о необходимости включения/выключения записи на sd-карточку измеренных значений, о необходимости звуковой и визуальной сигнализации, о необходимости перестройки параметров сигналов на каналах.

Параллельно с опросом клавиатуры работает поток Oscillation thread и SD service thread (если запись на sd-карточку включена). Эти потоки могут (и в большинстве случаях так и происходит) получать сообщения из обработчиков прерывания, обслуживающих два double-bufferа, связанных с выводом сгенерированных отсчетов сигналов и с сохранением измерений напряжений.

Рисунок 2.24 – Структурная схема программы

Параллельно с программной частью, работает аппаратная часть, которая считает тайм-ауты для потоков, следит за состоянием некоторых портов ввода/вывода, выполняет ADC и DAC преобразования, обслуживает запросы DMA и посылает сигналы прерывания, когда необходимо вмешательство программной части.

**Поток Polling button thread**

Данный поток занимается настройкой портов ввода/вывода, для опроса клавиатуры, опрашивает клавиатуру (ножки мк) и посылает сообщения различных типов потоку Menu thread. Опрос клавиатуры происходит каждые 10 мс (1 тик ОС FreeRTOS). Реализована state-machine с 3 состояниями, позволяющая избавиться от проблемы антидребезга и реализовать набор событий, которые могут быть посланы потоку Menu thread. А именно события о нажатии/отпускании клавиши, событие о том, что кнопка зажата уже больше 1 секунды. Также возможна посылка периодических событий, о том, что кнопка зажата (может быть использовано для ускорения перебора числовых значений, например кнопка “+”).

Сначала происходит инициализация структур и переменных, ответственных за работу потока, затем настройка аппаратуры и переход в ожидание тайм-аутов от ОС, для периодического опроса клавиатуры и посылки сообщения о событии, если оно произошло



Рисунок 2.25 –Диаграмма состояний state-machine, используемая потоком Polling button thread



Рисунок 2.26 –Структурная схема потока Polling button thread

**Поток Menu thread**

Поток принимает сообщения от потока Polling button thread и, в зависимости от текущего состояния программы, рассылает сообщения другим потокам. Реализует меню, для контакта с человеком, позволяющее управлять работой генератора-измерителя. Предоставляет несколько режимов отображения информации: режим генерации, режим измерения и режим управления качания частоты генерирования. Только этот поток посылает сообщения LCD, Beep и Measuring потокам, никакой другой поток или обработчик прерывания этого не делает.

Сначала происходит инициализация структур и переменных, ответственных за работу меню, затем поток переходит на ожидание и обработку поступающих сообщений.



Рисунок 2.27 – Структурная схема потока Menu thread

**Поток Measuring thread**

Поток настраивает аппаратуру, занимающуюся измерениями напряжений на ножках контроллера (ADC, TIM, DMA). После настройки поток ожидает команд (сообщений) от потока Menu thread на запуск/останов измерений. Аппаратура настраивается на измерение напряжений и перенос результатов измерений в double-buffer с генерацией прерывания при заполнении половины/всего буфера.

По событию таймера (не прерывание) ADC начинает последовательность преобразований 4 каналов (ножек контроллера). После завершения каждого преобразования генерируется DMA запрос на пересылку данных в double-buffer, из которого данные могут быть, потом, записаны на sd-карточку. Как только половина буфера (4 кбайта) заполнится, генерируется прерывание, которое посылает сообщение потоку SD service thread сигнал о необходимости записи кусочка данных. При частоте таймера 100 кГц, разрядности ADC 12 бит (16 бит для хранения, без сдвигов для экономии и т.д.), наборе из 4 каналов к преобразованию, получаем скорость заполнения этого буфера – 800 000 байт/с. Учитывая размер double-bufferа в 8 кбайт получаем, что прерывания о заполнении половины/всего буфера приходят с частотой 800 000 / 8096 \* 2 = 198 Гц, то есть каждые 5 миллисекунд необходимо записывать 4 кбайта данных на sd-карточку (≈800 кбайт/с).

Данные складываются в буфер в таком формате:



Сначала происходит инициализация структур и переменных, ответственных за работу данного потока, настраивается аппаратура, а затем поток переходит на ожидание и обработку поступающих сообщений.



Рисунок 2.28 – Структурная схема потока Measuring thread



Рисунок 2.29 – Аппаратная часть измерения напряжений

**Поток Oscillation thread**

Поток выполняет настройку аппаратуры, занимающейся генерацией сигналов заданной формы. После инициализации аппаратуры, поток выдает на оба канала нулевой сигнал и ожидает команд (сообщений) от Menu thread о типах сигналов и их параметрах. Также обслуживает сообщения, приходящие от обработчика прерывания, о заполнении половины/всего double-bufferа.

Генерация сигналов осуществляется методом DDS (Direct Digital Synthesis) – прямой цифровой синтез, реализованный программно. Аппаратура DAC позволяет “выбрасывать”, на выходы 2 каналов генератора, отсчеты одновременно.

Поэтому, при вычислении отсчетов, они сразу складываются в следующий формат:

,

где имеется 32 битное слово, которое будет переноситься в регистр DAC, младшие 12 бит которого будут интерпретироваться, аппаратурой DAC, как отсчеты для канала 2, а биты 27-16 - как отсчеты для канала 1.

По событию таймера (не прерывание) DAC выставляет на выходы каналов (ножки контроллера) значения отсчетов, которые были, предварительно, загружены в его регистр и делает запрос DMA на новые отсчеты, которые пересылаются в регистр DAC из double-bufferа. Как только половина данных переслана из буфера в регистр DAC генерируется прерывание, сигнализирующее (посылающее сообщение) о необходимости подготовить новый блок отсчетов. Это сообщение обслуживается потоком Oscillation thread.

Сначала происходит инициализация структур и переменных, ответственных за работу данного потока, настраивается аппаратура, а затем поток переходит на ожидание и обработку поступающих сообщений.



Рисунок 2.30 – Структурная схема потока Oscillation thread



Рисунок 2.31 – Аппаратная часть генерации сигналов

**Поток Measures printing thread**

Данный поток ожидает сообщений от Menu thread и, при поступлении сообщения, которое возможно только если включен режим измерения, вычисляет значения токов, напряжений и мощностей и посылает сообщения LCD потоку для отображения.



Рисунок 2.32 – Структурная схема потока Measures printing thread

**Поток LCD thread**

Данный поток обслуживает запросы на доступ к LCD аппаратуре. Сначала он инициализирует аппаратуру и необходимые данные потока для дальнейшего обслуживания запросов. Затем ожидает и обслуживает поступающие запросы. Поддерживаемые запросы: вывести строку (C массив) в позицию x, y; сдвиг курсора в позицию x, y; управление курсором и экраном. Есть проблема с переносимостью драйвера LCD, связанная с тем, что необходима непрерывность ножек для линий передачи данных (используется 4-битные режим). Поток создает пользовательский символ для отображения дисперсии. Также используется функция, написанная на ассемблере, скорее для интереса, чем для пользы.



Рисунок 2.33 – Структурная схема потока LCD thread

**Поток SD card writing thread**

Данный поток обслуживает запросы на доступ к аппаратуре sd-карточки. Сначала он инициализирует аппаратуру и необходимые данные потока для дальнейшего обслуживания запросов. Затем ожидает и обслуживает поступающие запросы. Запросы могут быть следующими: SD\_EVENT – необходимо подготовиться к записи (проинициализировать аппаратуру, открыть файл и записать дату открытия) или завершить запись (записать время закрытия, закрыть файл и деинициализировать аппаратуру). В качестве драйвера используется библиотека FATFS.

 Рисунок 2.34 – Структурная схема потока SD card writing thread

* + 1. **Блок-схемы программного обеспечения МК генератора**



Рисунок 2.35 – Блок-схема программного обеспечения микроконтроллера генератора тестовых сигналов

# ЗАКЛЮЧЕНИЕ

В этом проекте я проектировал цифровой генератор сигналов. Генератор может создавать прямоугольный, синусоидальный и пилообразный сигнал. Параметры сигналов можно гибко и удобно регулировать. Сигнал на выходе имеет высокую точность и относительно стабильнен. Устройство можно применяться как источник радиосигналов и шумов, что очень часто встречается в процессе анализа радиоэлектронных систем.

Основные принципы использованные при проектировании: многофункциональность, низкое энергопотреблением, разумная структура и легкость для отладки. В процессе проектирования системы, я стремился использовать простые схемы, максимально использовать ресурсы микроконтроллера в соответствии с требованиями системного проектирования. Большинство устройств, которые используются в схеме являются широко распространенными устройствами.

Ключевыми в этом проектировании являются модули: модуль генерации (МК STM32F103RCT6), модуль управления (клавиатура) и отображения (ЖКИ WH4002A) , модуль вывода (каскад ИФ-МУ-БУ). В целом конструкции этих трех модулей тесно связаны друг с другом, но проектировались они независимо. Функция каждой части влияет на общую функцию. В результате проектирования был создан прототип схемы. Некоторые функции устройства отличаются от требований проектирования и нуждаются в дальнейшем уточнении. Хотя в разработанной принципиальной схеме генератора есть некоторые незначительные недостатки, но в процессе проектирования, я получил очень хороший опыт практической работы.

# ПЕРЕЧЕНЬ ССЫЛОК

1. Дьяконов В.П. Генерация и генераторы сигналов. – М.: ДМК Пресс, 2009. – 384с.
2. О.Белоусов. Кварцевые генераторы. Режим доступа к электронному ресурсу: <http://rf.atnn.ru/s4/xg6.html>
3. Кушнир Ф.В. Электрорадиоизмерения. – Л.: Энергоатомиздат, 1983 – 320 с.
4. Литюк В.И., Литюк Л.В. Исследование цифровых генераторов шумов. – Таганрог, 2006. – 16с.
5. WH4002A Character 40x2 LCD. Datasheet. – Winstar Display Co., Ltd., 2012. – 1c. – Режим доступа к электронному ресурсу: <http://www.winstar.com.tw/products_detail_ov.php?lang=ru&ProID=40>
6. Барышев И.В., Мазуренко А.В., Горбуненко О.А. Прикладные вопросы цифровой обработки информации. Часть 1. Применение микроЭВМ в РТС управления, сбора и передачи информации. – ХАИ, 2006. – 118с.
7. STM32F103RCT6 - High-destiny performance line, ARM-based 32-bit MCU. Datasheet. – STMicroelectronics – 2009. – 9c. – Режим доступа к электронному ресурсу: http://www.alldatasheet.com/datasheet-pdf/pdf/303614/STMICROELECTRONICS/STM32F103RCT6TR.html
8. Хоровиц П., Хилл У. Искусство схемотехники, 1998. – 704с.
9. LM6142 – High Speed/Low Power 17 MHz Rail-to-Rail Input-Output Operational Amplifiers. Datasheet. – National Semiconductor – 1999. – 5c. – Режим доступа к электронному ресурсу: http://www.alldatasheet.com/datasheet-pdf/pdf/8996/NSC/LM6142.html

# ПРИЛОЖЕНИЕ А

**ПРОГРАММА MATLAB ДЛЯ МОДЕЛИРОВАНИЯ СИГНАЛОВ НА ВЫХОДЕ ЦАП И ИХ СПЕКТРОВ**

Для прямоугольных периодических импульсов со скважностью 5% и 50%

close all

clear

clc

A(1:5)=1;

A(6:100)=0;

A(101:105)=1;

A(106:200)=0;

A(201:205)=1;

A(206:300)=0;

A(301:305)=1;

A(306:400)=0;

A(401:405)=1;

A(406:500)=0; %импульсы со скважностью 5%

F1=fft(A); %быстрое преобразование Фурье

F2=fftshift(F1);

F=F2./max(F2); %нормированный спектр

for i=1:500

t(i)=1e-7\*i;

end %вектор времени

for i=1:500

f(i)=2e7/500\*i-1e7;

end; %вектор частоты

figure

plot(t,A); %рисовать импульс в временной области

figure

plot(f,abs(F)); %рисовать его спектр

B(1:50)=1;

B(51:100)=0;

B(101:150)=1;

B(151:200)=0;

B(201:250)=1;

B(251:300)=0;

B(301:350)=1;

B(351:400)=0;

B(401:450)=1;

B(451:500)=0; %импульсы со скважностью 50%

G1=fft(B); %быстрое преобразование Фурье

G2=fftshift(G1);

G=G2./max(G2); %нормированный спектр

for i=1:500

t1(i)=1e-7\*i; %вектор времени

end

for i=1:500

f1(i)=2e7/500\*i-1e7; %вектор частоты

end;

figure

plot(t1,B); %рисовать импульс в временной области

figure

plot(f1,abs(G)); %рисовать его спектр

figure

subplot(2,1,1)

plot(f,abs(F));

grid on

subplot(2,1,2)

plot(f1,abs(G));

grid on %сравнить два спектра сигнала с разной скважностью

figure

subplot(2,1,1)

plot(t,A);

ylim([-0.5 1.5])

grid on

subplot(2,1,2)

plot(t1,B);

ylim([-0.5 1.5])

grid on %сравнить два сигнала с разной скважностью

Для пилообразных периодических импульсов

close all

clear

clc

for i=1:100

A(i)=0.02\*i;

end

for i=101:200

A(i)=0.02\*(i-100);

end

for i=201:300

A(i)=0.02\*(i-200);

end

for i=301:400

A(i)=0.02\*(i-300);

end

for i=401:500

A(i)=0.02\*(i-400);

end %пилообразные периодические импульсы

F1=fft(A); %быстрое преобразование Фурье

F2=fftshift(F1);

F=F2./max(F2); %нормированный спектр

for i=1:500

t(i)=1e-7\*i; %вектор времени

end

for i=1:500

f(i)=20000\*i-5.02e6; %вектор частоты

end;

figure

plot(t,A); %рисовать импульс в временной области

figure

plot(f,abs(F))

grid on %рисовать его спектр

# ПРИЛОЖЕНИЕ Б

**ОПИСАНИЕ ФАЙЛОВ И ФУНКЦИЙ ПРОЕКТА ГЕНЕРАТОРА-ИЗМЕРИТЕЛЯ**

**Описание реализации потока Beep thread**

Аппаратные ресурсы, задействованные в реализации:

TIM2, TIM2\_IRQn (13, 1), GPIOA (1).

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

Beep\_Drv.h и Beep\_Drv.c

Beep\_Task.h и Beep\_Task.c

**Аппаратно-независимая часть**

Beep\_Task.h.

Beep\_Task.h содержит объявление функции потока void vBeepTask(void \*pvParameters), которая необходима при создании потока в функции Create\_OS\_Objects() (main.c).

Beep\_Task.c.

Beep\_Task.c включает хедеры Beep\_Task.h (определение функции vBeepTask), Beep\_Drv.h (определение аппаратно-зависимых функций) и CommonDefines.h (для взаимодействия с FreeRTOS).

Beep\_Task.c содержит в себе описание функции потока Beep thread - vBeepTask.

vBeepTask() инициализирует аппаратуру посредством вызова функции InitBeep(), которая будет описана ниже, и становится в бесконечный цикл ожидания сообщений от других потоков, посредством оператора while и вызова функции FreeRTOS xQueuePeek(), которая “усыпляет” поток до тех пор, пока не придет сообщение в очередь qTo\_Beep. Когда какой-либо поток пошлет сообщение в очередь qTo\_Beep, FreeRTOS “пробуждает” поток и передает ему управление. xqueuePeek() не извлекает сообщение из очереди. Затем вызывается функция xSemaphoreTake(), которая пытается захватить семафор, защищающий доступ к аппаратуре (о этом семафоре ниже), если удается захватить семафор, то xQueueReceive() извлекает сообщение из очереди qTo\_Beep и функция Beep() настраивает аппаратуру “beepera”, согласно данным извлеченного сообщения, если нет, то поток “засыпает” и ждет пока семафор не освободится (в обработчике прерывания).

**Аппаратно-зависимая часть (“драйвер“)**

Beep\_Drv .h.

Beep\_Drv .h включает хедер InterDefines.h (тут определены типы данных для сообщений).

Beep\_Drv .h содержит объявление функций InitBeep() (инициализация оборудования, ответственного за ”бипанье” ), Beep() (перенастройка оборудования, согласно указанным параметрам) и обновление глобальной переменной Num\_T (ответственной за реализацию серии звуковых импульсов). Также включен хедер stm32f10x.h и объявления переменных для настройки и работы с аппаратурой.

Beep\_Drv.c

Beep\_Drv.c ключает хедер Beep\_Drv.h (определение аппаратно-зависимых функций).

Beep\_Drv.c содержит в себе описание функций InitBeep() и Beep().

*InitBeep().*

InitBeep() разрешает тактирование таймера TIM2, настраивает его, настраивает NVIC на прерывание по TIM2, разрешает тактирование порта GPIOA и настраивает ножку PA1, с которой связана база транзистора, управляющего пьезоэлементом.

*Beep().*

Beep() до настраивает таймер TIM2, согласно данным полученным из BeepParam (сообщение от других потоков), вычисляет значение глобальной переменной Num\_T, запускает TIM2 и разрешает прерывание.

**Механизм защиты аппаратуры от одновременного доступа**

Поскольку пьезоэлемент один, то доступ к нему нужно контролировать. Пусть поток А посылает сообщение нашему потоку. FreeRTOS будит наш поток, передавая управление функции QueuePeek(). Затем вызывается xSemaphoreTake(), которой удается захватить семафор, значит продолжаем выполнение дальше, извлекаем сообщение и передаем его функции Beep(). Beep() настраивает TIM2, и он начинает считать. Пока таймер не досчитал до конца (и не освободил семафор) другой поток Б (а может и этот же самый А) посылает новое сообщение нашему потоку. Если бы не было механизма защиты, то Beep() перенастроила бы TIM2, не дав ему досчитать предыдущие промежутки (насчет помежуткИ, смотри реализацию обработчика прерывания). А в текущей реализации наш поток заснет, пока семафор не будет освобожден в обработчике прерывания, после того как TIM2 обслужит предыдущий запрос.

**Описание реализации потока Calc thread. (вспомогательный поток)**

Аппаратные ресурсы, задействованные в реализации:

none.

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

Calc\_Drv.h и Calc\_Drv.c

Calc\_Task.h и Calc\_Task.c

**Аппаратно-независимая часть**

Calc\_Task.h.

Calc\_Task.h содержит объявление функции потока vCalcTask, которая необходима при создании потока в функции Create\_OS\_Objects() (main.c).

Calc\_Task.c.

Calc\_Task.c включает хедеры Calc\_Task.h (определение функции vCalcTask), Calc\_Drv.h (определение аппаратно-зависимых функций) и CommonDefines.h (для взаимодействия с FreeRTOS).

Calc \_Task.c содержит в себе описание функции потока Calc thread - vCalcTask.

vcalcTask() реализует бесконечное ожидание сообщений от других потоков, посредством оператора while и xQueueReceive(), которая ожидает сообщений на очереди queu\_to\_calc. Как только сообщение было послано, нашему потоку, FreeRTOS нас будит и передает управление xQueueReceive(), которая извлекает сообщение-указатель, указывающий на буфер, данные которого необходимо обработать. После обработки данные передаются функции show\_result() для отображения на экране. Более подробно об обработке данных и буфере смотри описание потока Measuring thread.

Посылка сообщений нашему потоку возможно только лишь если переменная lock\_send\_message\_to\_calc\_thread равна нулю.

**Аппаратно-зависимая часть** (лишь для терминологии, не содержит непосредственной работы с аппаратурой)

Calc\_Drv .h.

Calc\_Drv .h содержит объявление функций init\_calc() (заглушка) и show\_result() (вывод на экран данных переданных через параметры функций).

Calc\_Drv.c

Calc\_Drv.c ключает хедер Calc\_Drv.h (определение аппаратно-зависимых функций).

Calc\_Drv.c содержит в себе описание функций init\_calc(), show\_result() и обьявление внешней функции lcd\_write() для вывода информации на экран.

*show\_result().*

Создает форматированные строки, содержащие информацию о токах, напряжениях и мощностях, для вывода на LCD экран. Токи в мА, напряжения в В, мощности в мВт.

Это вспомогательный поток, задача которого является вывод мгновенных, не усредненных значений токов, напряжений и мощностей для визуального восприятия человеком.

**Описание реализации потока LCD thread**

Аппаратные ресурсы, задействованные в реализации:

GPIOC (12), GPIOB (5 - 10).

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

LCD\_Drv.h и LCD\_Drv.c

LCD\_Task.h и LCD\_Task.c

LCD\_Delay.s

**Аппаратно-независимая часть**

LCD\_Task.h.

LCD\_Task.h содержит объявление функции потока vLCDTask, которая необходима при создании потока в функции Create\_OS\_Objects() (main.c).

LCD\_Task.c.

LCD\_Task.c включает хедеры LCD\_Task.h (определение функции vLCDTask), LCD\_Drv.h (определение аппаратно-зависимых функций), CommonDefines.h (для взаимодействия с FreeRTOS) и InterDefines.h (тут определены типы данных для сообщений).

LCD\_Task.c содержит в себе описание функции потока LCD thread - vLCDTask().

vLCDTask() инициализирует аппаратуру посредством вызова функции InitLCD(), которая будет описана ниже, и становится в бесконечный цикл ожидания сообщений от других потоков, посредством оператора while и вызова функции FreeRTOS xQueueReceive(), которая ожидает сообщений на очереди qM\_to\_LCD. После поступления сообщения, оно извлекается из очереди и дальше, основываясь на ID\_cmd, управлении передается одной из функций: SendString(), ShiftBlink(), ControlBlink().

**Аппаратно-зависимая часть (“драйвер“)**

LCD\_Drv .h.

LCD\_Drv .h содержит объявление функций InitLCD() (инициализация аппаратуры, связанной с LCD), SendString(), ShiftBlink(), ControlBlink(), ClearLCD() (функции управления экраном и выводом на экран), InitGPIO\_LCD(), SendFirstCommand() и Preinit\_LCD() (ответственны за инициализацию аппаратуры), SendCommand(), SendData() и CreateSigmaPattern() (ответственны за низкоуровневые операции с LCD), Delay\_asm() (функция-задержки, написанная на ассемблере, для реализации маленьких задержек, от 238 нс, при 72 МГц тактовой частоты) и дефайны для функций, работающих с аппаратурой LCD.

LCD\_Drv.c

LCD\_Drv.c включает хедер LCD\_Drv.h (определение аппаратно-зависимых функций).

LCD\_Drv.c реализует низко-уровненные функции для работы с LCD аппаратурой (библиотека LCD), функции инициализации LCD аппаратуры и функции, представляющие интерфейс драйвера ( SendString(), …).

*CreateSigmaPattern().*

Функция CreateSigmaPattern() создат в памяти LCD контроллера картинку символа, для отображения символа дисперсия.

*Delay\_asm().*

Данная функция написана на ассемблере, для большой точности измерения временных интервалов (задержек), и для интереса :-).

**Строка, указатель на которую приходит в сообщении, не должна быть расположена в стеке вызова функции, которая вызывается из любой функции-потока. Расположение в стеке возможно только, если поток, который посылает сообщение, имеет приоритет меньше чем приоритет у LCD thread. (в таком случае LCD thread получит управление сразу и указатель будет указывать на валидные данные)**

**Описание реализации потока Menu thread**

Аппаратные ресурсы, задействованные в реализации:

2kB SRAM. Must be rechecked.

Реализация потока содержит часть, зависящею от FreeRTOS и аппаратуры и не зависящею часть (меню). Реализация представлена четырьмя файлами: .c и .h для части зависящей и .c и .h для части не зависящей.

Упор делается на независимость реализации меню.

Файлы реализации:

Menu\_Drv.h и Menu\_Drv.c

Menu\_Task.h и Menu\_Task.c

**Зависящая часть**

Menu\_Task.h.

Menu\_Task.h содержит объявление функции потока vMTask, которая необходима при создании потока в функции Create\_OS\_Objects() (main.c), и объявление аппаратно-зависимых функций.

Menu\_Task.c.

Menu\_Task.c включает хедеры Menu\_Task.h (определение функции vMTask), Menu\_Drv.h (определение аппаратно-зависимых функций), CommonDefines.h (для взаимодействия с FreeRTOS) и InterDefines.h (тут определены типы данных для сообщений).

LCD\_Task.c содержит в себе описание функции потока Menu thread - vMTask() и реализацию аппаратно-зависимых функций.

*VMTask().*

VMTask() инициализирует, посредством функции InitView(), первоначальное самостояние меню, а значит и всего генератора-измерителя, и становится в бесконечный цикл ожидания сообщений от других потоков, посредством оператора while и вызова функции FreeRTOS xQueueReceive(), которая ожидает сообщений на очереди qPB\_to\_M. После прихода сообщения, оно передается функции PrepareMesg(), которая извлекает из него необходимую информацию (тип клавиатуры, код клавиши и тип события ). Затем эта информация передается state-machine, реализующей меню MenuStateMach().

*ChangeChar().*

Функция ChangeChar() меняет символ 'ъ' на символ, код которого равен 0xFF, и который будет отображен контроллером LCD, как символ дисперсии. (предварительно он туда загружен функцией CreateSigmaPattern()).

**Строка, указатель на которую передается в функцию ChangeChar(), не может располагаться в Flash-памяти, то есть быть const-антной, поскольку функция меняет значение строки. Располагайте или в стеке или в куче. На счет кучи в данном проекте смотрите дальше.**

**Не зависящая часть**

Menu\_Drv .h.

Menu\_Drv .h содержит объявление функций InitView() (инициализация состояния меню), MenuStateMach() (state-machine меню), Switch\_to\_EditMode(), EditParam() и ReDrawParam() , ReDrawZeroRow(), re\_draw\_body() (внутренние функции state-machine меню), объявление структур sParam, sSignalParam и sChannel (ответсвенны за реализацию state-machine меню), определение перемнных этих и других типов.

Menu\_Drv.c

Menu\_Drv.c включает хедер Menu\_Drv.h (определение аппаратно-зависимых функций).

Menu\_Drv.c реализуте функции InitView(), MenuStateMach(), Switch\_to\_EditMode(), EditParam(), ReDrawParam(), ReDrawZeroRow() и re\_draw\_body() и определяет строки-константы для отображения меню на экране.

*InitView().*

InitView() устанавливает текущий канал 1:

CurChannel = 0;

режим редактирования выключен:

EditMode = 0;

режим измерения включен:

is\_meas\_mode = 1;

Заполнят поля массива структур Channel, для каждого канала задает сигналы, их параметры (доступность, значения) и другую вспомогательную информацию.

Затем перерисовывает верхнюю строку (ReDrawZeroRow()) и остальные строки (re\_draw\_body()) LCD экрана, выключает мигание курсора blink\_control(), посылает сообщения потоку, ответственному за генерацию сигналов (Oscillation thread), о необходимости настроить каналы на генерацию, заданных выше сигналов (preInitDAC()) и включает измерения (посылает сообщение потоку Measuring thread, ответственному за измерения) (meas\_control()).

*MenuStateMach().*

Каждый раз, когда Poliing button thread посылает сообщение нашему потоку, сообщая таким образом, что произошло какое-то событие с клавиатурой, вызывается функция MenuStateMach(), которая управляет состоянием меню. Первым делом она прекращает свое дальнейшее выполнение, если меню (а значит и весь генератор), находится в режиме измерений и текущее событие не является событием о нажатии кнопок '+' или '-'. Затем идет switch по номеру кнопки, для которой пришло сообщение. Для кнопок с кодами 2-5 сперва выходим из режима редактирования:

EditMode = 0,

а затем вызываем функцию Switch\_to\_EditMode(), с параметрами соответствующими коду кнопки. Для кнопок с кодом 6 и 7, в режиме редактирования — вызываем функцию EditParam() с параметром, соответствующими коду, иначе — управляем режимом измерения (вкл./выкл. и изменение состояния переменной lock\_send\_message\_to\_calc\_thread) (для кнопки с кодом 6); посылаем сообщение потоку SD writing thread. А он уже следит нужно ли включить запись и разрешить посылку сообщений себе или выключить и запретить посылку сообщений себе. (посредством переменной lock\_send\_message\_to\_sd\_thread) (для кнопки с кодом 7).

**Описание реализации потока Measuring thread**

Аппаратные ресурсы, задействованные в реализации:

DMA1 ch 1, GPIOC (0-3), ADC1 (10-13), ADC\_ExternalTrigConv\_T3\_TRGO, ADC\_DMA, DMA1\_Channel1\_IRQn, TIM3, T3\_TRGO, 8kB SRAM.

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

Meas\_Drv.h и Meas\_Drv.c

Meas\_Task.h и Meas\_Task.c

**Аппаратно-независимая часть**

Meas\_Task.h.

Meas\_Task.h содержит объявление функции потока vMeasTask(), которая необходима при создании потока в функции Create\_OS\_Objects() (main.c).

Meas\_Task.c.

Meas\_Task.c включает хедеры Meas\_Task.h (определение функции vMeasTask()), Meas\_Drv.h (определение аппаратно-зависимых функций), CommonDefines.h (для взаимодействия с FreeRTOS).

Meas\_Task.c содержит в себе описание функции потока Measuring thread - vMeasTask().

*vmeasTask().*

Функция vmeasTask() становится в бесконечный цикл ожидания сообщений от других потоков, посредством оператора while и вызова функции FreeRTOS xQueueReceive(), которая ожидает сообщений на очереди queu\_to\_meas. Функция контролирует текущее состояние связки TIM-ADC-DMA и запускает/останавливает эту цепочку, то есть измерение параметров каналов генерации, исходя от сообщения и текущего состояния. Запуск/останов выполянется посредством вызова функций start\_meas()/stop\_meas().

**Аппаратно-зависимая часть (“драйвер“)**

Meas\_Drv .h.

Meas\_Drv .h содержит объявление функций start\_meas() (запуск аппаратуры ответственной за измерения) и stop\_meas() (останов аппаратуры ответственной за измерения), определение глобального указателя p\_beg\_adc\_buff (используется в обработчике прерывания DMA1\_Channel1\_IRQHandler() ) и некоторые дефайны.

Meas\_Drv.c

Meas\_Drv.c включает хедер Meas\_Drv.h (определение аппаратно-зависимых функций).

Meas\_Drv.c выделяет, статически, память под циклический double-buffer gl\_adc\_buff для хранения измеренных значений и реализует функции start\_meas() и stop\_meas().

*start\_meas().*

Функция start\_meas() устанавливает глобальный указатель p\_beg\_adc\_buff на циклический double-buffer gl\_adc\_buff, настраивает и запускает цепочку TIM3-ADC1-DMA1.

*stop\_meas().*

Функция stop\_meas() останавливает цепочку TIM3-ADC1-DMA1. (после останова можно использовать аппаратуру в других целях).

**Описание реализации потока Oscillation thread**

Аппаратные ресурсы, задействованные в реализации:

DAC (1, 2), DMA2 ch 3, DMA2\_Channel3\_IRQn, TIM4, TIM4\_TRGO\_event, 4kB SRAM.

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

Osc\_Drv.h и Osc\_Drv.c

Osc\_Task.h и Osc\_Task.c

**Аппаратно-независимая часть**

Osc\_Task.h.

Osc\_Task.h содержит объявление функции потока vOscTask(), которая необходима при создании потока в функции Create\_OS\_Objects() (main.c).

Osc\_Task.c.

Osc\_Task.c включает хедеры Osc\_Task.h (определение функции vOscTask()), Osc\_Drv.h (определение аппаратно-зависимых функций), CommonDefines.h (для взаимодействия с FreeRTOS).

Osc\_Task.c содержит реализацию функцию vOscTask().

*vOscTask().*

Функция vOscTask() инициализирует аппаратуру, ответственную за работу связки DAC-TIM-DMA (InitDAC\_TIM\_DMA()) и становится в бесконечный цикл ожидания сообщений от других потоков, посредством оператора while и вызова функции FreeRTOS xQueueReceive(), которая ожидает сообщений на очереди qTo\_Osc. После прихода сообщения, проверяется откуда оно пришло (по полю Sig\_Type). Если из обработчика прерываний DMA2\_Channel3\_IRQHandler(), то вызывается функция ReFill(), которая сменяет текущую заполняемую область циклического double-buffer DAC\_Buff, измененяет указатель pDAC\_Buff и заполняет текущую область циклического double-buffer DAC\_Buff отсчетами текущих сигналов обоих каналов. Иначе, то есть от потока Menu thread, вызывается функция ReCalc(), которая заполняет следующую область циклического double-buffer DAC\_Buff пересчитанными, согласно полученным данным из сообщения, отсчетами сигнала для ОДНОГО канала.

**Аппаратно-зависимая часть (“драйвер“)**

Osc\_Drv .h.

Osc\_Drv .h включает хедер InterDefines.h (тут определены типы данных для сообщений).

Osc\_Drv .h содержит объявление функций InitDAC\_TIM\_DMA() (запуск аппаратуры ответственной за генерацию), ReFill(), ReCalc() (реализация double-buffer и обслуживание запросов), и sweep\_control() (управление качением частоты), выделение, статическое, памяти под циклический double-buffer DAC\_Buff, некоторые структуры и переменные, массив указателей на функции, ответственных за вычисления отсчетов конкретных сигналов, переменные для DDS и некоторые дефайны.

Osc\_Drv.c

Osc\_Drv.c включает хедер Osc\_Drv.h (определение аппаратно-зависимых функций).

Osc\_Drv.c реализует InitDAC\_TIM\_DMA(), ReFill(), ReCalc() и sweep\_control().

*InitDAC\_TIM\_DMA().*

Данная функция заполняет массив указателей на функции, ответственных за вычисления отсчетов конкретных сигналов, адресами соответствующих функций. Заполняет обе части циклического double-bufferа отсчетами zero-сигнала для двух каналов, вызывая 4 раза функцию ReCalc(), устанавливает текущую половину заполняемого буфера (CurFillArea = 1) , для коррктного старта генерации. Затем происходит настройка и запуск аппаратуры, ответственной за генерацию сигналов.

*ReFill().*

Данная функция (вызывается по событию заполнение половины циклического буфера) меняет указатель на половину циклического буфера, которая должна быть заполнена новыми отсчетами и вызывает функции для заполнения этой половины буфера отсчетами двух каналов.

*ReCalc().*

Данная функция (вызывается по событию от потока Menu thread) устанавливает текущий канал (CurDAC\_Ch = pOscParam->Ch\_num;), исходя из сообщения от Menu thread, и вызывает функцию для заполнения отсчетами текущей половины циклического буфера для установленного канала. Внутри этой функции происходит пересчет модельки сигнала благодаря first\_entry[CurDAC\_Ch] = 0.

*sweep\_control().*

Данная функция управляет качением частоты генерируемых сигналов. Следит за направлением изменения частоты, удерживает частоту в заданных рамках :-), и загружает в регистр частоты FREQ\_REG новое, вычисленной, значение.

**Описание реализации потока Polling button thread**

Аппаратные ресурсы, задействованные в реализации:

GPIOC(7-9), GPIOA(8-12).

Реализация потока содержит аппаратно зависимую часть и часть не зависящею от аппаратуры. Реализация представлена четырьмя файлами: .c и .h для части зависящей от аппаратуры и .c и .h для части не зависящей от аппаратуры.

Файлы реализации:

PolBut\_Drv.h и PolBut\_Drv.c

PolBut\_Task.h и PolBut\_Task.c

**Аппаратно-независимая часть**

PolBut\_Task.h.

PolBut\_Task.h содержит объявление функции потока vPBTask(), которая необходима при создании потока в функции Create\_OS\_Objects() (main.c) и Send\_PolBut\_Message() (отправка сообщения потоку Menu thread).

PolBut\_Task.c.

PolBut\_Task.c включает хедеры PolBut\_Task.h (определение функции vPBTask() и Send\_PolBut\_Message() ), PolBut\_Drv.h (определение аппаратно-зависимых функций), CommonDefines.h (для взаимодействия с FreeRTOS).

PolBut\_Task.c содержит реализацию функцию vPBTask().

*vPBTask().*

Функция vPBTask() инициализирует аппаратуру, ответственную за работу опроса клавиатуры и становится в бесконечный цикл ожидания получения управления от FreeRTOS, посредством оператора while и вызова функций FreeRTOS xTaskGetTickCount() и vTaskDelayUntil(). После получения управления функция вызывает GetKeyPadState() для проверки, произошло ли какое-либо событие на клавиатуре и, если произошло, посылает сообщение потоку Menu thread, посредством вызова Send\_PolBut\_Message().

*Send\_PolBut\_Message().*

Формирует сообщение для посылки потоку Menu thread.

**Аппаратно-зависимая часть (“драйвер“)**

PolBut\_Drv .h.

PolBut\_Drv .h включает определение структуры sKeyMesg и sButtonDescr и определение функций InitPad\_Task(), GetKeyPadState(), initGPIO\_Pads(), GetSst\_Key(), Key\_State\_machine() и выделение памяти под массив sButtonDescr[8].

PolBut\_Drv.c

PolBut\_Drv.c включает хедер PolBut\_Drv.h (определение аппаратно-зависимых функций).

PolBut\_Drv.c реализует InitPad\_Task(), GetKeyPadState(), initGPIO\_Pads(), GetSst\_Key() и Key\_State\_machine().

*InitPad\_Task().*

Вызывает initGPIO\_Pads() и заполняет поля структур массива ButtonDescr\_K[].

*initGPIO\_Pads().*

Настраивает аппаратуру, связанную с опросом клавиатуры.

*GetKeyPadState().*

Вызывает функцию GetSst\_Key() для проверки состояния на линиях ввода/вывода и, если нажата только ОДНА кнопка, вызывает для всех восьми кнопок функцию Key\_State\_machine(), которая следит за их состоянием.

*GetSst\_Key().*

Проверяет состояние линий ввода/вывода.

*Key\_State\_machine().*

Реализует state-machine для опроса кнопок. Следит за состоянием кнопки (нажата, отжата и зажата).

Таблица распределения кучи под стеки потоков

|  |  |
| --- | --- |
| Имя потока | Размер стека (в байтах) |
| PBTask | 256\*4 |
| MTask | 800\*4 |
| LCDTask | 256\*4 |
| OscTask | 512\*4 |
| CalcTask | 256\*4 |
| SDTask | 256\*4 |
| MeasTask | 256\*4 |
| Общее потребление памяти: 10368 байт | |

Таблица распределения памяти под глобальные переменные

|  |  |
| --- | --- |
| Тип и имя переменой | Размер (в байтах) |
| sChannel Channel[2] | 2048 |
| unsigned short gl\_adc\_buff[8192] | 16384 |
| uint32\_t DAC\_Buff[1024] | 4096 |
| Общее потребление памяти: 22528 байт | |

Таблица распределения RAM

|  |  |
| --- | --- |
| Тип памяти | Размер (в байтах) |
| Общий стек (под main()) | 800 |
| Куча под стеки потоков | 10368 |
| глобальные переменные | 22528 |
| Куча под очереди, семафоры и служебную информацию FreeRTOS + не учтенные глобальные переменые |  |
| Общее потребление памяти: байт | |

**Main**

*main().*

Данная функция получает управление после возникновения прерывания reset, настройки flash-памяти и частоты работы ядра контроллера.

Выводит на LCD строку загрузки "Loading...", производит инициализацию некоторых глобальных переменных, которые должны быть инициализированы до запуска FreeRTOS, настраивает аппаратуру ответственную за мониторинг ножек контролера (для старта записи измерений на sd- карточку), создает объекты FreeRTOS (потоки, семафоры и мютексы) и запускает планировщик. Ели запуск произошел успешно, то эта функция не получит управление назад. Дальнейший ход программы будет зависеть от FreeRTOS, приоритетов потоков и внешних событий.

**Обработчики прерывания**

*DMA2\_Channel3\_IRQHandler().*

Обслуживает прерывание о заполнение половины буфера от DMA2 Channel 3. Используется для генерации сигналов. Посылает сообщение, с полем Sig\_Type = 7, в голову очереди потоку Oscillation thread. В голову, потому что потоку Oscillation thread может посылать сообщения еще и Menu thread, а реакция на сообщение из обработчика прерывания должна быть максимально быстрой.

*DMA1\_Channel1\_IRQHandler().*

Обслуживает прерывание о заполнение половины буфера от DMA1 Channel 1. Используется для измерения параметров генерируемых сигналов. Посылает сообщение потоку Calculate thread, если разрешено, и потоку SD write control thread, если разрешено.

*TIM5\_IRQHandler().*

Обслуживает прерывание о переполнение счетчика таймера TIM5. Используется потоком SD write control thread. Вызывает функцию disk\_timerproc() для нужд FATFs.

*EXTI4\_IRQHandler()/EXTI9\_5\_IRQHandler().*

Обслуживают прерывание о изменении состоянии ножек контроллера. Используются для запуска записи измеренных данных на sd-карточку, без нажатия кнопки. Блокирует посылку сообщений потоку SD write control thread, из обработчика прерывания DMA1\_Channel1\_IRQHandler, и посылает потоку SD write control thread событие SD\_EVENT.