|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_Компьютерные системы и сети\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***ПОДСИСТЕМА ФОРМИРОВАНИЯ ПАРАМЕТРОВ САМОЛЁТА ДЛЯ ПРОВЕДЕНИЯ ТЕСТОВЫХ ИСПЫТАНИЙ***

Студент \_\_\_ИУ6-81\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Веинский В. А.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ерёмин О. Ю.**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2019 г.*

**АННОТАЦИЯ**

В настоящей выпускной квалификационной работе бакалавра описан процесс разработки подсистемы формирования параметров самолёта для проведения тестовых испытаний.

Произведен анализ различных систем для тестирования летательных аппаратов, изучен процесс проведения испытаний и программное обеспечение, необходимое для этого. По результатам анализа определены основной функционал и некоторые технические решения, особенности разрабатываемой подсистемы. Выбран способ хранения данных.

Разработаны технологии задания параметров, отправки данных на удалённый компьютер, тестирования подсистемы. Разработан интерфейс программного обеспечения.

**ABSTRACT**

*In this final qualifying work of the bachelor describes the process of developing a subsystem of formation of the parameters of the aircraft for testing.*

*The analysis of various systems for testing aircraft, studied the process of testing and software required for this. Based on the results of the analysis, the basic functionality and some technical solutions, features of the developed subsystem are determined. The method of data storage is selected.*

*The technologies of setting parameters, sending data to a remote computer, testing the subsystem are developed. The software interface has been developed.*

**РЕФЕРАТ**

Записка 64с., 19 рис., 20 табл., 12 источников

ПАРАМЕТРЫ САМОЛЁТА, СТЕНДОВЫЕ ИСПЫТАНИЯ, ОТПРАВКА ДАННЫХ, ЗАДАНИЕ ПАРАМЕТРОВ

Объектом разработки выпускной квалификационной работы бакалавра является подсистема формирования параметров самолёта для проведения тестовых испытаний, с помощью которой пользователь может просматривать и задавать текущие параметры самолёта, чтобы смоделировать его поведение при различных условиях внешней среды и при наличие каких-либо сбоев в аппаратной или программной частях самолёта.

Актуальность данной темы заключается в том, что каждый запускаемый в производство самолёт должен пройти ряд технических испытаний, быть безопасным и вести себя в соответствии с требованиями, установленными при разработке. Любое проведение испытаний не может обойтись без программного обеспечения, которое предоставит возможность испытателю изменять параметры самолёта в соответствии с тестом, который необходимо провести.

Документация включает в себя расчетно-пояснительную записку, графическую часть, приложения с руководством пользователя и листингом программы.

В результате работы была разработана подсистема, включающая в себя приложение для хранения, задания и отправки параметров самолёта.

Задачи, решаемые в процессе разработки:

- анализ предметной области;

- разработка структурной схемы программного обеспечения;

- выбор способа отправки данных;

- разработка эргономичного интерфейса;

- разработка формы хранения данных.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc11399047)

[1 Анализ методов тестирования летательных аппаратов 8](#_Toc11399048)

[1.1 Виды испытаний 8](#_Toc11399049)

[1.2 Лётные испытания 9](#_Toc11399050)

[1.3 Наземные испытания 14](#_Toc11399051)

[1.4 Анализ программного обеспечения, для проведения стендовых испытаний 17](#_Toc11399052)

[2 Разработка подсистемы формирования параметров 20](#_Toc11399053)

[2.1 Выбор модели жизненного цикла 20](#_Toc11399054)

[2.2 Анализ функциональных возможностей подсистемы 22](#_Toc11399055)

[2.3 Выбор программных средств для разработки 26](#_Toc11399056)

[2.4 Особенности проектируемой подсистемы 28](#_Toc11399057)

[2.5 Разработка пользовательского интерфейса программы 29](#_Toc11399058)

[2.5.1 Выбор типа интерфейса 29](#_Toc11399059)

[2.5.2 Выбор типа диалога 29](#_Toc11399060)

[2.5.3 Выбор формы диалога 30](#_Toc11399061)

[2.5.4 Разработка графа состояний интерфейса 30](#_Toc11399062)

[2.5.5 Разработка форм интерфейса 32](#_Toc11399063)

[2.6 Разработка структурной схемы приложения 34](#_Toc11399064)

[2.7 Разработка диаграммы классов 36](#_Toc11399065)

[2.8 Разработка диаграммы компоновки 39](#_Toc11399066)

[2.9 Алгоритм работы программы 41](#_Toc11399067)

[2.10 Тестирование подсистемы 46](#_Toc11399068)

[2.10.1 Функциональное тестирование 47](#_Toc11399069)

[2.10.2 Оценочное тестирование подсистемы 49](#_Toc11399070)

[3 Технологические решения 53](#_Toc11399071)

[3.1 Разработка технологий хранения данных 53](#_Toc11399072)

[3.1.1 Структура файла хранения параметров 53](#_Toc11399073)

[3.1.2 Структура файла хранения описания таблиц 56](#_Toc11399074)

[3.1.3 Файл инициализации set\_parametrs.ini 57](#_Toc11399075)

[3.2 Разработка технологии тестирования подсистемы 58](#_Toc11399076)

[ЗАКЛЮЧЕНИЕ 61](#_Toc11399077)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 62](#_Toc11399078)

[ПРИЛОЖЕНИЕ А. Техническое задание 64](#_Toc11399079)

[ПРИЛОЖЕНИЕ Б. Руководство пользователя 73](#_Toc11399080)

[ПРИЛОЖЕНИЕ В. Фрагмент исходного кода подсистемы 78](#_Toc11399081)

[ПРИЛОЖЕНИЕ Г. Копии листов графической части 92](#_Toc11399082)

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

ПО – программное обеспечение

ПК – персональный компьютер

ЛА – летательный аппарат

КИС – контрольно-испытательная станция

СУОСО – система управления обще-самолётным оборудованием

РМП – рабочее место пилота

РМИ – рабочее место испытателя

IDE – Integrated Development Environment (интегрированная среда программирования)

# ВВЕДЕНИЕ

С каждым годом количество новой авиационной техники растёт, разрабатываются новые модели самолётов. Требования к качеству материалов и надёжности программной и аппаратурной составляющих летательных устройств увеличиваются. Чтобы проверить, удовлетворяет ли разработанный самолёт требованиям эксплуатации и безопасности, необходимо провести испытания. Испытания бывают различных видов. Особенно сильно в программном обеспечении (далее ПО), не связанным непосредственно с ПО самолёта, нуждаются наземные испытательные стенды.

К таким стендам относятся контрольно-испытательные станции (далее КИС), на которых возможно тестирование математической модели самолёта (симуляция) и реальных самолётных систем. Разрабатываемая подсистема подходит для обоих вариантов испытательных стендов и необходима для их базового функционирования.

Подобные подсистемы уже существуют и используются для тестирования различных моделей самолётов, одна для каждого нового стенда пишется новое ПО в соответствии с требованиями к месту проведения и объекту испытаний. Назначения и различные особенности летательного средства вносят свои корректировки в разрабатываемое для проведения испытаний ПО, поэтому для каждого нового самолёта необходимо как минимум внести серьёзные доработки в уже имеющееся ПО, но чаще его пишут заново.

Разрабатываемая подсистема планируется для использования на стенде МС-21, поэтому разрабатывается с учётом особенностей площадки и самого самолёта.

# Анализ методов тестирования летательных аппаратов

Испытание [1] — процесс, неразрывно связанный с созданием любых летательных аппаратов (далее ЛА) на всех этапах, начиная с составления технического задания, когда возникает необходимость провести экспериментальные проверки и исследования новых научных и технических проблем, и кончая запуском производства и последующими модификациями серийных ЛА (рисунок 1).

Программа испытаний состоит из двух частей — наземной и летной. Наземная отработка объединяет работы, которое проводятся на всех уровнях, — от элемента до комплексной системы. Летная отработка ЛА является завершающим этапом. Затраты на лётные испытания составляют около половины трудовых и материальных ресурсов, расходуемых при разработке ЛА, а их продолжительность достигает до 50% всей длительности цикла создания ЛА. При отработке образца может совершаться до двух-трех тысяч испытательных полетов.

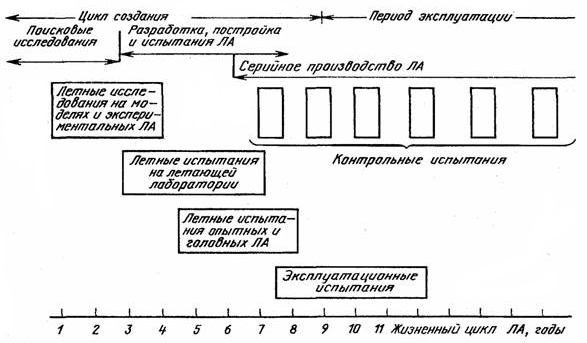


Рисунок 1 – Испытания на различных стадиях создания ЛА

## Виды испытаний

Весь объем испытаний делится, как правило, на несколько этапов (категорий):

* заводские (предварительные) испытания, которые выполняются головным разработчиком с участием соисполнителей и НИИ отрасли;
* государственные испытания, проводимые заказчиком с участием специалистов промышленности;
* эксплуатационные испытания, которые проводятся испытательными организациями заказчика совместно с эксплуатирующими подразделениями и с участием представителей промышленности.

## Лётные испытания

Летные испытания регламентируются типовыми программами, которые на ряду с общими характеристиками для различных видов ЛА отражают специфические особенности, определяемые их назначением.

Общий объем летных испытаний регламентируют следующие программные документы: типовая программа испытаний ЛА данного класса, общая программа испытаний данного ЛА, частные рабочие программы испытаний отдельных экземпляров опытных образцов и их функциональных систем, программы специальных испытаний, задание на полет [2]. Перед летными испытаниями ставятся следующие цели:

* исследование особенностей аэродинамики, маневренности, пилотирования вблизи критических режимов;
* определение специфики компоновки и систем управления ЛА;
* изучение особенностей работы двигателя и его систем в натурных условиях;
* оценка эффективности и особенностей работы пилотажно-навигационных комплексов и систем бортового оборудования;
* отработка систем безопасности полета и средств спасения экипажа;
* предварительная оценка работоспособности и стыковки специальных бортовых систем и их взаимодействия с наземными средствами управления.

Испытания нуждаются в мощной производственной базе, способной строго по графику поставлять необходимое число испытываемых объектов. Эта же производственная база должна достаточно быстро и полно осуществлять необходимую доводку ЛА и его систем по материалам испытаний. Доводка ЛА и его систем осуществляется на летно-испытательных и доводочных базах при головных ОКБ, либо на предприятиях-изготовителях.

Каждый новый (опытный) летательный аппарат проходит ряд испытаний с момента его создания и до внедрения в серийное производство. Вначале на летающих лабораториях проводится отработка наиболее ответственных установок и агрегатов аппарата. Затем в процессе заводских испытаний проводят доводку до состояния, обеспечивающего безопасность выполнения полетов, ведут отработку и оценку систем бортового и наземного оборудования, определяют основные летно-технические и эксплуатационные характеристики. Далее летательный аппарат поступает на государственные испытания для выявления и регламентации условий, обеспечивающих наибольшую надежность, безопасность и простоту пилотирования, а также для определения соответствия техническому заданию его реальных летно-технических данных и основных эксплуатационных характеристик. При этом решается вопрос о запуске ЛА в серийное производство.

Затем проводятся эксплуатационные испытания опытной серии лета тельных аппаратов и испытания первых аппаратов серий. Основная задача эксплуатационных испытаний ЛА состоит в выявлении всех наиболее характерных особенностей его наземной и летной эксплуатации, пригодности и совершенства вспомогательного оборудования и технических средств, в выявлении и регламентации условий, обеспечивающих наибольшую регулярность и экономичность полетов на новом летательном аппарате в зависимости от назначения. В ходе эксплуатационных испытаний проверяются и уточняются:

* регламенты технического обслуживания;
* руководства и инструкции по летной эксплуатации и технике пилотирования;
* трудоемкость обслуживания летательного аппарата, потребная численность технического персонала, способы эксплуатации и обслуживания ЛА.

Летные испытания опытного самолета по сравнению с испытаниями серийного имеют следующие особенности. Во-первых, на опытном самолете есть специальная экспериментальная аппаратура, позволяющая получить значительно больше информации о летных характеристиках самолета и работе бортовых систем и планера. Во-вторых, в процессе испытаний опытного самолета допустимы проверки режимов, требующих от летчика исключительного мастерства, чрезмерного внимания и большого физического напряжения. В силу этого, опытные самолеты нередко оборудуются специальными средства ми спасения, отсутствующими на серийных самолетах. В-третьих, существует возможность замен и доработок отдельных элементов летательного аппарата по результатам летных испытаний.

Летные испытания позволяют определить важнейшие летные характеристики самолета — максимальную скорость, практический потолок, дальность и продолжительность полета, наиболее выгодную скорость набора высоты, вертикальную скорость и время набора высоты при оптимальной скорости. Летные характеристики (летные данные) определяют технические возможности самолета.

Не менее важными факторами, определяемыми в летных испытаниях, являются характеристики устойчивости и управляемости летательных аппаратов. Оценка устойчивости и управляемости должна производиться в полном диапазоне допустимых скоростей полета, высот, чисел Маха, углов атаки и скольжения, центровок, при полной и малой тяге двигателя, с автоматикой и без нее.

Наиболее ответственными и сложными летными испытаниями являются полеты с целью определения предельных скоростных на поров и предельных чисел Маха летательного аппарата; при этом ограничениями являются общая прочность или жесткость конструкции, вибрация, устойчивость и управляемость, особенность работы силовой установки и других систем, аэродинамический нагрев. К специальным летным испытаниям относятся испытания на сваливание в штопор. Весьма важными являются летные испытания, проводимые с целью определения характеристик маневренности летательного аппарата. Маневренность лета тельного аппарата в целом не может быть охарактеризована каким-либо одним обобщающим критерием. Для оценки маневренных качеств необходимо находить из летных испытаний те характеристики, которые определяют основные параметры маневров.

Основные факторы, исследуемые в процессе летных испытаний. Летные испытания проводятся и для определения взлетно-посадочных характеристик лета тельных аппаратов. При этом определяются длина и время разбега, скорость отрыва, длина и время пробега, посадочная скорость и другие характеристики. Помимо определения взлетно-посадочных характеристик, соответствующих условиям нормальной эксплуатации, при летных испытаниях по возможности проводят полеты с имитацией аварийных ситуаций, на пример посадки однодвигательного самолета с отказавшим двигателем, взлета и посадки многодвигательного самолета с одним отказавшим двигателем и др.

Летные испытания проводятся для отработки всех бортовых систем лета тельного аппарата в условиях нормальной эксплуатации и с имитацией отказов. Так, например, при испытаниях топливной системы определяют “границу высотности” системы (т. е. ограничения по кавитации), надежность подачи горючего к двигателю при всех возможных режимах работы двигателя (стационарных и переходных) и всех возможных режимах полета, включая резкие маневры.

Летные испытания серийных самолетов. Основная особенность летных испытаний серийных летательных аппаратов по сравнению с испытаниями опытных экземпляров заключается в повторении полетов на аппарате одного и того же типа. Однако большой диапазон скоростей и других летных параметров вызывает значительные различия в характеристиках управляемости и других характеристиках однотипных летательных аппаратов, хотя при конструктивно-технологической отработке прилагаются максимальные усилия к обеспечению полного единообразия.

Смысл летных испытаний на серийном заводе состоит прежде всего в выявлении правильности регулировочных работ с целью доведения всех параметров до пределов, ограниченных техническими условиями. Кроме того, при летных испытаниях выявляются и устраняются дефекты и отказы готовых изделий, а также дефекты и отказы, появляющиеся в результате нарушения технологических процессов на серийном заводе.

Первый испытательный полет серийного самолета начинается с наземного осмотра. При этом проверяются подвижность и полнота отклонения рулей и триммеров, действие закрылков, воздушных тормозов, управление стабилизатором, давление в пневмо и гидросистемах. Необходимо также проверить исправность механизма открывания и закрывания фонаря, качество видимости через стекла фонаря. Пилотажные приборы и приборы контроля двигателя, а также топливная система проверяются при работающих двигателях. При рулении проверяется работа тормозов, для этого нужно развить предельную скорость руления, а затем резко затормозить и еще раз провести общую проверку систем самолета. Во время взлета необходимо проконтролировать работу двигателей, процесс уборки шасси и закрылков. Во время набора высоты проверяются обороты двигателя, давление в системе подачи топлива, устойчивость и управляемость.

Затем проводятся проверки параметров устойчивости и управляемости при различных маневрах и на максимальной горизонтальной скорости, при работе воздушных тормозов, на режиме потери скорости. Осуществляются замер давления на ручку управления при пикировании до максимальной скорости и проверка при этом работы рулей (испытания на пикирование проводятся только для легких самолетов), а также замер давления на ручку управления при выходе из пикирования на максимальной скорости.

Основными задачами второго полета являются определение времени набора практического потолка, пикирование с этой высоты до достижения скорости, соответствующей максимально допустимому числу Маха, замер скоростей горизонтального полета на высотах, больших, чем при первом полете.

Для пассажирских самолетов программа летных испытаний определяется на основании действующих норм летной годности, которые регламентируют все основные требования к летным характеристикам пассажирских самолетов. Так, проверка соответствия характеристик устойчивости и управляемости самолета согласно нормам, должна производиться на всех основных режимах полета. Методика проверки должна предусматривать выполнение полетов с рядом контрольных маневров и имитациями отказа двигателей, а также имитациями отказов одной из гидросистем управления рулями и других элементов системы управления.

Предельные по условиям прочности режимы полета согласно нормам, должны быть проверены в летных испытаниях, если их достижению не препятствуют ограничения по условиям аэродинамики или конструктивные особенности самолета, делающие невозможным достижение этих режимов без чрезмерных усилий летчика и особого мастерства пилотирования.

## Наземные испытания

Наземные испытания проводятся как при подготовке ЛА к первому вылету, так и в процессе текущей (межполётной) подготовки. Под наземными испытаниями понимается проверка и отработка штатных систем, установленных на борту ЛА. Их преимущество заключается в использовании полного комплекта аппаратуры со всеми связями и коммуникациями реальной кабины экипажа и всех линий управления. Перед наземными испытаниями ставятся следующие цели:

* проверка работоспособности агрегатов и систем, устранение дефектов, снятие контрольных характеристик функционирования;
* на стройка регулируемых параметров объектов испытаний и сопряженных с ними систем;
* анализ особенностей функционирования систем, обнаруженных в полетах;
* градуировка измерительной аппаратуры, установленной на борту ЛА.

Созданное диагностическое оборудование, специальные имитаторы, подвижные стенды позволяют воспроизводить в наземных испытаниях соответствующие внешние условии и имитировать режим полета.

Проверка и испытания всех систем самолетов не должны влиять на ресурс элементов оборудования и гарантийный ресурс самолета в целом. Для некоторых видов аппаратуры бортового оборудования небольшая предварительная наработка необходима и полезна. В частности, для электронной высокочастотной аппаратуры устанавливается обязательная предварительная наработка, называемая тренировкой. Тренировку можно применять к механическим и гидравлическим системам. В процессе тренировки происходят прирабатывание отдельных элементов механизмов и самоподстройка контуров. После тренировки повторно проверяют и испытывают соответствующие элементы оборудования.

Основная часть контрольной оснастки и стендов КИС благодаря своей универсальности аналогична контрольному оборудованию, применяемому в цехе окончательной сборки.

В особую группу оборудования КИС входят испытательные стенды, моделирующие условия работы бортовых систем самолета. На таких стендах имитируются нормальные и аварийные режимы работы, а также регистрируются переходные процессы. Результаты испытаний (мера достоверности и объективность контроля) зависят от полноты воспроизводимых рабочих режимов системы, степени автоматизации воспроизведения внешних условий работы системы, изменения их различных параметров, от измерения и регистрации самих результатов.

Моделирующие испытательные стенды позволяют:

* проверять теоретические расчеты, принятые при разработке новой аппаратуры;
* определять оптимальные схемы и параметры для проектируемой системы путем исследования в реальных условиях работы агрегатов;
* проводить контрольные испытания системы для определения отклонения ее параметров от заданных значений;
* проводить ресурсные испытания или испытания до первого отказа;
* воспроизводить аварийные режимы в связи с рекламациями от эксплуатирующих организаций;
* моделировать конструктивные и производственные дефекты с целью выявления их влияния на работоспособность и надежность системы и т. д.

Одной из наиболее трудоемких и ответственных операций при на земной отработке является проверка системы управления. Система проверяется для режимов ручного и автоматического управления. Определяются углы отклонения органов управления, переда точные числа, работоспособность всех агрегатов системы.

При контроле шасси проверяется кинематика всех движущихся агрегатов и деталей, за меряется время выпуска и уборки закрылков, контролируется работа механизмов поворота переднего колеса и тормозов.

При проверке топливной системы контролируется прежде всего ее герметичность, для чего на шесть-восемь часов система подвергается воздействию избыточного давления порядка 15...20 Па.

Контроль системы высотного оборудования заключается в проверке работоспособности агрегатов и в определении температур и расходов воздуха в различных точках магистралей. Проводятся также проверки гидравлической, пневматической, электрической, радиолокационной и других систем.

После отработки, контроля и регулирования отдельных систем самолет транспортируют на девиационную площадку, где проверяют по пеленга тору и корректируют показания магнитных, гиромагнитных приборов и радиокомпасов. Для списывания девиации компасов самолет устанавливают на поворотном круге. Далее определяют полетную массу и положение центра тяжести самолета, взвешивая его в двух или более вариантах загрузки (пустой самолет с несъемным оборудованием и полностью загруженный).

## Анализ программного обеспечения, для проведения стендовых испытаний

Чтобы разобраться, какое ПО необходимо для работы стенда рассмотрим схему модуля управления стендом (рисунок 2), на которой отображены рабочие места сотрудников и всё необходимое для проведения испытаний оборудование [3].

Основные модули:

* рабочее место пилота (далее РМП) представляет из себя имитацию кабины пилота. В ней находятся один или два испытателя, которые исполняют роль пилота и контролируют движение самолёта, информацию на приборной доске и непосредственные действия электроники в различных ситуациях;
* рабочее место испытателя (далее РМИ) представляет из себя один из 7 компьютеров, каждый из которых отвечает за контролирование и тестирование определённой системы самолёта. Один из компьютеров отведён для ведущего испытателя – с него возможно просмотреть состояние любой из систем самолёта. Так же именно на этих компьютерах осуществляется задание параметров, которые влияют на каждую из систем;
* на сервер регистрации данных приходит информация о состоянии параметров с самолётных блоков, где с помощью специальной программы они считываются и становятся доступны для просмотра. Просмотр осуществляется с любого компьютера в сети с помощью программы для удалённого доступа к рабочему столу;
* персональный компьютер (далее ПК) управления принимает информация ото всех РМИ, обрабатывает информацию, генерирует сигнала и отправляет их платы для дальнейшей передачи на самолётные блоки.

На основании проанализированной информации о процессе работы стенда можно сделать следующие выводы:

* иметь возможность использовать разрабатываемую подсистему должен каждый из испытателей, работающих непосредственно с заданием параметров тестирования, поэтому необходимо будет разместить программу на каждом компьютере РМИ, что позволит любому испытателю вносить изменения в интересующие его параметры, оставаясь на своём рабочем месте и имея возможность легко работать с другими программами, необходимыми для проведения испытаний, не отвлекаясь на перемещение между компьютерами для задания параметров самолёта;
* важно добиться универсальности, чтобы программа не зависела от конкретных параметров и подсистем самолёта, к которым они относятся, чтобы добиться возможности использовать практически одинаковые дистрибутивы на всех компьютерах;
* необходимо предусмотреть возможность настройки, какому именно удалённому компьютеру отправлять файлы, и в какую именно директорию. Проанализировать формат файла, используемого для генерации сигналов на ПК управления: именно этот файл должен приходить на удалённый ПК;
* стоит учитывать требования, которые предъявляются к методике проведения испытаний, что позволит лишний раз упростить и уменьшить затрачиваемое время испытателей на работу с программой, что очень важно, так как позволит сэкономить драгоценное время, а соответственно и сильно ускорить весь цикл разработки и ввода в эксплуатацию.

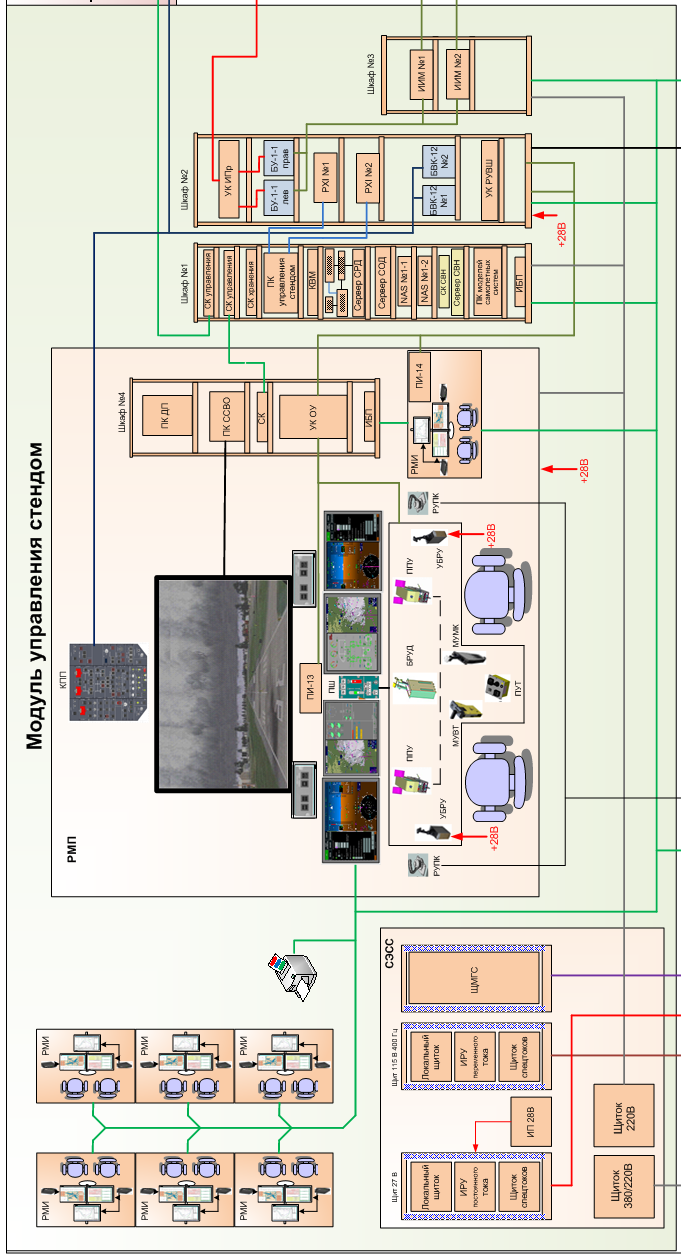


Рисунок 2 - Схема модуля управления стендом

# Разработка подсистемы формирования параметров

## Выбор модели жизненного цикла

После анализа требований, предоставленных к разрабатываемой подсистеме, была выбрана модель её жизненного цикла:

Спиральная модель (рисунок 3) жизненного цикла сочетает в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции [4]. Позволит быстрее создать работоспособный продукт, чтобы можно было скорее начать стендовые испытания, а соответственно и ускорит выпуск самолёта. Повторные циклы позволят модифицировать программу под новые тесты. Необходимо стремиться к тому, чтобы каждая итерация была результативна, а каждая версия работоспособна.

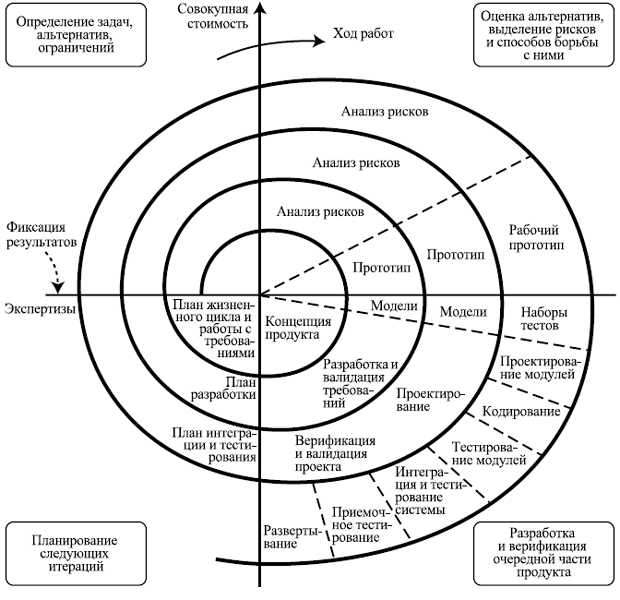


Рисунок 3 – Спиральная модель разработки

Преимущества спирального подхода:

* позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;
* допускает изменение требований при разработке программного обеспечения, что характерно для большинства разработок, в том числе и типовых;
* в модели предусмотрена возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной модели, и в то же время разрешены итерации по всем фазам этой же модели;
* позволяет получить более надежную и устойчивую систему. По мере развития программного обеспечения ошибки и слабые места обнаруживаются и исправляются на каждой итерации;
* эта модель разрешает пользователям активно принимать участие при планировании, анализе рисков, разработке, а также при выполнении оценочных действий;
* уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта;
* обратная связь по направлению от пользователей к разработчикам выполняется с высокой частотой и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества.

Подход к проектированию целесообразно выбрать объектный, так как все части интерфейса являются объектами, а решаемую задачу можно декомпозировать на отдельные модули, которые удобно представить объектами.

Выбранный жизненный цикл позволит в кратчайшие сроки получить работоспособную версию программы (чтобы ввести стенд в эксплуатацию), легко вносить изменения, которые могут понадобится из-за изменения программы работы стенда, постепенно расширять функционал подсистемы, а также настроить её для удобства эксплуатации испытателями.

## Анализ функциональных возможностей подсистемы

Для разработки пользовательского интерфейса необходимо определить и проанализировать все возможные действия пользователя в программе. Для этого построим диаграмму вариантов использования, которая продемонстрирует все возможности разрабатываемой подсистемы. Предполагается наличие всего одного типа пользователя. Им является испытатель, который и будет проводить необходимые для проведения тестов действия. К этим действиям относятся: внесение значений параметров самолёта, настройка, запуск и остановка испытаний. Более подробно набор функций системы для пользователя рассмотрен на рисунке 4. В таблицах 1 – 10 описаны варианты использования, а также альтернативные действия.

Таблица 1 – Описание для варианта использования – Запуск приложения

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь запускает приложение. | 2. Происходит чтение .ini файла.  3. Восстановление настроек программы, в соответствии с теми, что были при прошлом завершении работы.  4. Открытие файла, который был при завершении работы в прошлый раз. |

Таблица 2 – Описание для варианта использования – Загрузить файл

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь нажал на кнопку «Загрузить файл».  3. Пользователь выбирает файл, с которым собирается работать. | 2. Программа открывает диалоговое окно для выбора файла.  4. Программа загружает информация из файла и выводит на экран для обработки. |

Альтернатива:

2. Если на в файл внесены изменения, которые не были сохранены, то перед открытием нового файла пользователю предложат сохранить предыдущий.

Таблица 3 – Описание для варианта использования – Сохранить файл

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь нажал на кнопку «Сохранить». | 2. Программа выполняет внесение в файл проведённых пользователем изменений. |

Альтернатива:

2. Если никаких изменений не было внесено, то программа файл не обновляет.

Таблица 4 – Описание для варианта использования – Сохранить файл как

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь нажал на кнопку «Сохранить как».  3. Пользователь выбирает путь и имя файла. | 2. Программа открывает диалоговое окно для выбора пути и имени сохраняемого файла.  4. Создаётся указанный пользователем файл и все текущие изменения записываются в него. |

Альтернатива:

4. Если указанный пользователем файл уже существует, то программа предлагает варианты дальнейших действий: изменить имя файла, перезаписать или отменить процедуру создания файла.

Таблица 5 – Описание для варианта использования – Установить

длительность

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь вводит длительность проведения теста (в секундах) | 2. Программа запоминает значение |

Альтернатива:

2. Если указанные пользователем данные некорректны (например, буквы вместо чисел), то программа сообщает об этом.

2. Если введённые данные отсутствуют, то программа сообщает об этом.

Таблица 6 – Описание для варианта использования – Выбор типа исходных

данных

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь ставит или снимает галочку напротив поля «Модельные данные» | 2. Программа определяет, откуда необходимо брать параметры |

Таблица 7 – Описание для варианта использования – Запустить тест

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь нажимает на кнопку «Старт» | 2. Программа проверяет наличие корректного времени тестирования.  3. Проверяет наличие открытого файла (в случае отсутствия галочки «Модельные данные»)  4. Запускается проведение испытания |

Альтернатива:

3. Если не открыт файл, то выдаётся сообщение, что невозможно начать испытание из-за отсутствия данных.

Таблица 8 – Описание для варианта использования – Остановить тест

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь нажимает на кнопку «Останов» | 2. Программа останавливает проведение теста. |

Альтернатива:

2. Если тест ещё не запущен, то кнопка его остановки недоступна.

2. Если после нажатия не удалось отправить команду об остановке испытания, то выводится соответствующее сообщение, а также в лог файл записываются возможные причины возникновения этой ошибки.

Таблица 9 – Описание для варианта использования – Просмотр исходных

параметров

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь запускает программу и загружает файл.  3. Пользователь выбирает устройство.  5. Пользователь выбирает канал. | 2. Программа формирует списки устройств, параметров принадлежащим данным устройствам и распределенным по каналам.  4. Программа показывает таблицу, содержащую все каналы данного устройства.  6. Программа показывает таблицу, содержащую все параметры и их значения данного канала. |

Альтернатива:

1. При запуске программы файл уже может быть загружен (из предыдущей сессии).

2. При неудачном чтении файла, выводится сообщение об этом.

Таблица 10 – Описание для варианта использования – Запись параметров

|  |  |
| --- | --- |
| Типичный ход событий | |
| Действие пользователя | Отклик системы |
| 1. Пользователь выбирает параметр, который хочет изменить.  3. Пользователь выбирает один из способов задания значения параметра.  5. Пользователь задаёт параметр в соответствии с выбранным способом. | 2. Программа выводит информацию о данном параметре.  4. Для выбранного способа задания параметра появляется возможность внести изменения  6. Программа сохраняет внесённые пользователем изменения в таблицу. |

Альтернатива:

3. Если пользователь не выбирает какой-либо способ задания параметра, то по умолчанию остаётся тот же, что и был.

6. При некорректных данных, выводится сообщение об ошибке.

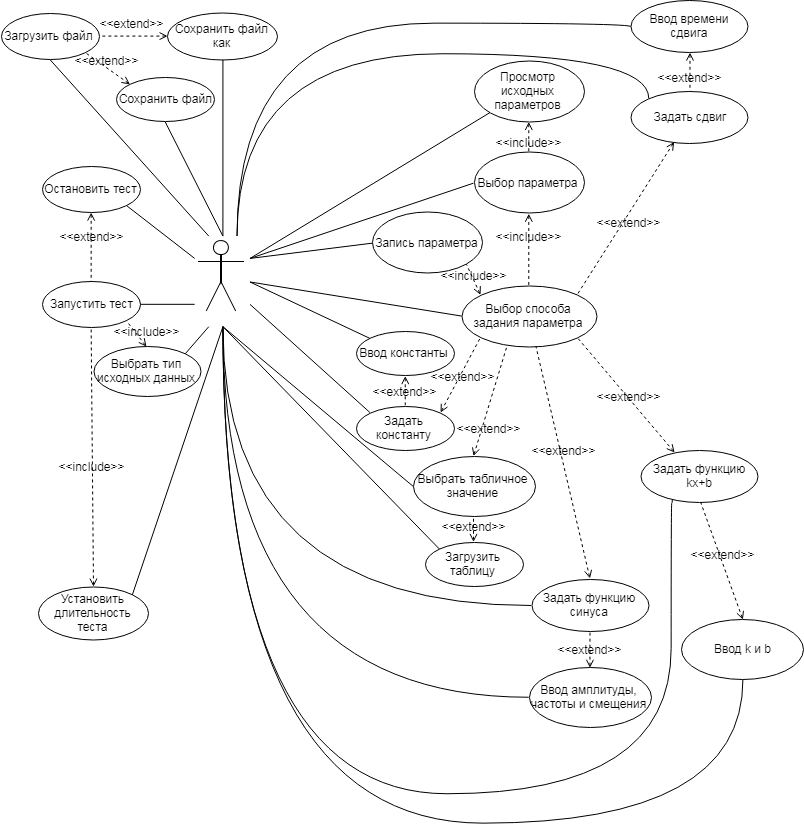


Рисунок 4 – Диаграмма вариантов использования

## Выбор программных средств для разработки

После определения всех функций подсистемы, которые необходимо реализовать, необходимо определиться с программными средствами, используемыми при разработке.

На рынке программных средств существует множество решений от популярных производителей. Для разработки сложной подсистемы очень важно выбрать удобное и функциональное решение, которое наиболее упростит и ускорит процесс разработки. Наиболее важными критериями в данном проекте являются возможности решить большинство задач в одной среде разработке, а также наличие лицензии у предприятия или полная бесплатность программы.

Microsoft Visual Studio [5] — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд сопутствующих инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe).

Qt Creator [6] (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная IDE для разработки на С, С++ и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

C++ Builder [7] — программный продукт, инструмент быстрой разработки [приложений](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), [интегрированная среда программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (IDE), система, используемая программистами для разработки программного обеспечения на языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

C++ Builder объединяет в себе комплекс [объектных библиотек](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ([STL](https://ru.wikipedia.org/wiki/Standard_Template_Library), [VCL](https://ru.wikipedia.org/wiki/VCL), [CLX](https://ru.wikipedia.org/wiki/CLX), [MFC](https://ru.wikipedia.org/wiki/MFC) и др.), [компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), [редактор кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%BA%D0%BE%D0%B4%D0%B0) и многие другие компоненты. Цикл разработки аналогичен [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)). Большинство компонентов, разработанных в Delphi, можно использовать и в C++ Builder без модификации.

C++ Builder содержит инструменты, которые при помощи [drag-and-drop](https://ru.wikipedia.org/wiki/Drag-and-drop) действительно делают [разработку визуальной](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), упрощает программирование благодаря встроенному [WYSIWYG](https://ru.wikipedia.org/wiki/WYSIWYG) — [редактору интерфейса](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0&action=edit&redlink=1) и пр.

Eclipse [8] — [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [модульных](https://ru.wikipedia.org/wiki/Plug-in_Development_Environment) [кроссплатформенных](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) приложений. Развивается и поддерживается [Eclipse Foundation](https://ru.wikipedia.org/wiki/Eclipse_Foundation).

Наиболее известные приложения на основе [Eclipse Platform](https://ru.wikipedia.org/wiki/Eclipse_Platform) — различные «Eclipse [IDE](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)» для разработки ПО на множестве языков.

Таблица 11 – Сравнительная характеристика сред разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Microsoft Visual Studio | Qt Creator | C++ Builder | Eclipse |
| Наличие бесплатной версии | Нет | Нет | Нет | Да |
| Наличие лицензии | Нет | Нет | Да | Да |
| Наличие опыта работы | Да | Нет | Да | Нет |
| Наличие возможности разработки интерфейса | Нет | Да | Да | Нет |
| Удобство работы (по 10 балльной шкале) | 9 | 7 | 8 | 6 |

Вывод: в ходе проведённого исследования доступных сред разработки программного обеспечения на языке С++, было принято решение использовать C++ Builder, потому что он обладает встроенными возможностями для создания графической части и предприятие, для которого производится разработка, обладает лицензией на эту программы.

## Особенности проектируемой подсистемы

Поскольку разрабатываемая подсистема входит в состав системы испытательного стенда летательных аппаратов, то необходимо продумать взаимодействие с другим программным обеспечение. Программы, которые взаимодействуют с разрабатываемой определяют многие технологические и конструкторские решения, на пример: формат данных, способ их хранения и передачи на удалённый компьютер. Поэтому некоторые из принятых решений могут быть не самыми эффективными или удобными в угоду взаимодействия с уже существующим ПО.

## Разработка пользовательского интерфейса программы

### Выбор типа интерфейса

Эргономичность интерфейса и его удобство для пользования являются весомым параметром оценки качества программного обеспечения. Грамотно разработанный интерфейс приложения зачастую определяет его успешность.

Исходя из основных задач программы, и ее специфики, целесообразно применить процедурно-ориентированный подход к разработке интерфейса. Данных подход дает возможность пользователю, выполнять определенные действия, задавать необходимые данные, и в итоге получать требуемый результат.

Существует три типа процедурно-ориентированных интерфейсов [9]: примитивный, меню и со свободной навигацией. Разумным выбором будет использование интерфейса со свободной навигацией (или графического пользовательского интерфейса). Данный тип интерфейса наиболее часто используется в оконных приложениях, и позволяет пользователю в интерактивном режиме взаимодействовать с элементами управления и ввода данных.

### Выбор типа диалога

Задача пользовательского интерфейса заключается в обеспечении взаимодействия пользователя и компьютера. Как и любое общение, данное взаимодействие осуществляется посредством диалога. Диалог между компьютером и пользователем обеспечивает обмен информацией между участниками диалога. Существуют разные типы диалога: управляемый пользователем и управляемый программой. Если диалог управляется пользователем, то процесс обмена информацией зависит от пользователя. Этот тип диалога позволяет реализовывать различные пользовательские сценарии. Если диалог управляется программой, то процесс обмена информации строго линеен, и задается программой.

Разрабатываемое программное обеспечение предполагает достаточно большое количество сценариев использования, поэтому наиболее подходящим типом диалога будет управляемый пользователем.

### Выбор формы диалога

Форма диалога описывает «язык», на котором пользователь и компьютер обмениваются информацией. Как и любой другой язык, данный язык имеет синтаксис и семантику. Различные виды синтаксиса и семантики определяют три формы диалога: фразовая, директивная и табличная.

Проанализировав данные формы диалога, и применив их особенности к разрабатываемому программному обеспечению, было решено использовать преимущественно табличную форму диалога, дабы избежать некорректного ввода некоторых параметров, а также упростить пользователю задачу. Однако некоторые поля все же требую свободного ввода, поэтому в целом, форму диалога, разрабатываемого ПО можно считать комбинированной.

### Разработка графа состояний интерфейса

На основании разработанной диаграммы вариантов использования (пункт 2.2), а также технического задания можно понять, что должно входить в интерфейс разрабатываемой программы, а именно: главная форма, в которой и решается основная задача – обработка параметров самолёта, а также форма для загрузки и сохранения внешних файлов, содержащих эти самые параметры. Роль второй выполняет стандартный проводник Windows. Запуск и завершение работы программы осуществляется из главной формы приложения. Подобное отображения состояний интерфейса позволит наглядно представить необходимые компоненты для разработки самих форм. Рассмотрим состояния интерфейса при взаимодействии с пользователем на графе состояния интерфейса программы (рисунок 5).



Рисунок 5 – Граф состояния интерфейса

С1 – Запуск программы;

С2 – Определение типа обрабатываемых данных (модельные или нет);

С3 – Установка времени проведение теста;

С4 – Нажатие на кнопку «Старт»;

С5 – Нажатие на кнопку «Останов»;

С6 – Выбор канала;

С7 – Выбор параметра;

С8 – Выбор типа значения параметра;

С9 – Нажатие на кнопку «Загрузить»;

С10 – Нажатие на кнопку «Сохранить»;

С11 – Нажатие на кнопку «Сохранить как»;

С12 – Выбор пути файла;

С13 – Выбор имени файла;

С14 – Подтверждение выбора;

С15 – Задание параметра;

С16 – Переключение на вкладку «Настройки»;

С17 – Закрытие программы.

Также на главной форме располагаются окна, в которых происходит основной рабочий процесс, а именно взаимодействие с параметрами проведения испытания. Для настройки ПО существует вкладка «Настройки»

### Разработка форм интерфейса

Как было определено при создании графа состояния интерфейса, необходимо создать всего одну форму (рисунок 6), так второй будет являться стандартный проводник Windows.

В верхней части формы отображается полный путь до текущего открытого файла, кнопки для работы с файлом (загрузить, сохранить, сохранить как), поле для установки длительности теста, поле для установки модельных данных, кнопка завершения работы и кнопки для начала и остановки теста.

Для наличия возможности просмотра данных на этой форме необходимо отобразить две таблицы:

1. Для выбора устройства и канала
2. Для выбора параметра в этом канале

После выбора параметра из соответствующего ему канала появляется возможность выбрать его тип и задать значение. Также при выборе параметра на экране отображается информация о нём: название, единица измерения, максимальное и минимальное значения.

RadioButton служит для выбора всех возможных способов задания параметров, а также все поля, необходимые для задания параметра выбранным способом. Для параметров, заданных табличными значениями или синусом присутствует возможность просмотра графика изменения сигнала во времени.

Настройка параметров программы вынесена в отдельную вкладку. Здесь можно внести изменения в соответствии с теми данными, что прописаны в файле инициализации. Т. е. просто визуализирует .ini файл, но добавляет возможность просмотра и изменения тех параметров программы, что не указаны в файле и берутся по умолчанию.

Во время проведения теста осуществляется отображение таймера с текущим значением, что позволяет визуально контролировать время проведения испытания и отлавливать момент, когда таймер выйдет и испытание остановится.

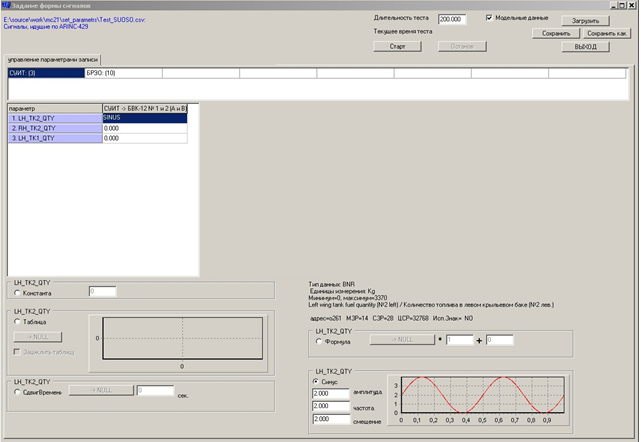


Рисунок 6 – Главная форма приложения

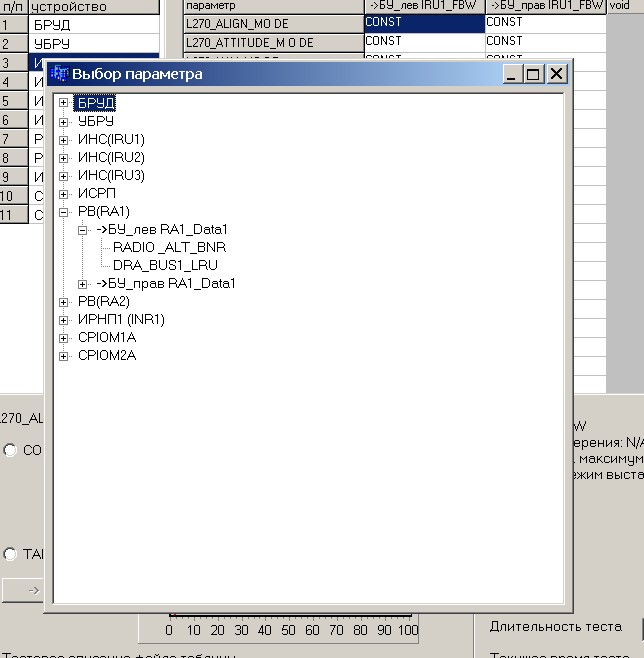


Рисунок 7 – Пример управления ссылочными параметрами

В списке, представленном в виде дерева <устройство>->»канал»-><параметр> (рисунок 7). Для обеспечения однозначности при разборе ссылочного параметра используется свойство параметра «уникальный идентификатор». В списке присутствуют только элементы, которые имеют тип данных BNR или DW. В данном окне выбирается ссылочный элемент – двойным щелчком «мыши». При применении выбранного параметра выполняется проверка на недопустимость рекурсии. Данная проверка выполняется по полной цепочке ссылочных параметров.

При нажатии на кнопку «старт» сначала выполняется проверка на несохраненные данные. Выдается запрос на сохранение, если такие данные найдены. Затем выполняется копирование всех зависимых файлов в указанную в файле инициализации директорию. Затем в зависимости от состояния параметра, указывающего на наличие модельных данных, отсылается команда на простой старт или на старт с учетом модельных данных (в случае установленного ModelCheckBox часть параметров берётся из рефлективной памяти).

## Разработка структурной схемы приложения

После уточнения интерфейса выполняется декомпозиция предметной области задачи разрабатывается структурная схема будущего продукта и описывается взаимодействие его функциональных элементов. Разрабатываемая схема позволит в дальнейшем шаге составить диаграмму классов, отражающую все выделенные в этом пункте модули.

Структурная схема, отражающая состав и взаимодействие по управлению частей разрабатываемого продукта. При объектной декомпозиции такими частями являются объекты, однако в текущей разработке объекты тесно переплетены с интерфейсом, поэтому несколько разных задач могут быть реализованы одним классом. В разрабатываемой схеме проведём декомпозицию на значимые модули, а их реализация будет описана в последующих пунктах. Структурная схема приведена на рисунке 8.

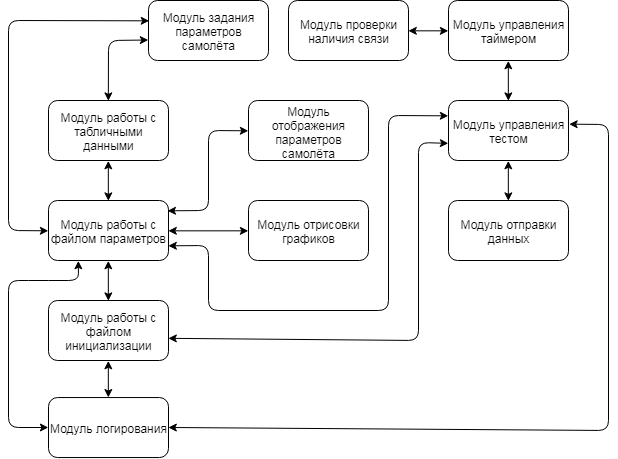


Рисунок 8 – Структурная схема

Вышеуказанная диаграмма показывает взаимодействие модулей предметной области. Рассмотрим каждый из них подробнее и опишем полученные особенности:

* для записи логов используется отдельный модуль. В него будут записываться действия, связанные с прохождением теста, с изменением файлов параметров и настроек программы. Результат работы этого модуля позволит в случае возникновения проблем или ошибок устранить их в кратчайшие сроки;
* графики, отображающие поведение параметров во времени, рисуются для параметров, заданных функцией синуса или табличными данными. Позволяет наглядно изучить поведение параметра с течением времени. Данные берутся из модуля для хранения параметров.
* присутствует модуль для проверки наличия связи с удалённым компьютером (осуществляет опрос по таймеру).

## Разработка диаграммы классов

Перейдём к реализации модулей, рассмотренных в пункте 2.6 модулей. Поскольку был выбран объектный подход, необходимо составить диаграмму классов (рисунок 9), которая будет наглядно отображать какие именно задачи выполняются тем или иным элементом программы.

Перед построением диаграммы стоить заметить, что для написания интерфейса программы используется встроенная в C++ Builder библиотека Windows Forms. Полностью отображать иерархию объектов внутри этой библиотеки нет смысла, поэтому первым будет объект, являющийся главной формой разрабатываемого приложения [10]. Для других элементов, используемых из этой библиотеки, также не будет описана иерархия и все они будут указаны, как поля основного класса.

Также необходимо учесть, что некоторые модули выполняют очень значимые задачи, однако реализовывать их в отдельных классах не всегда представляется удобным и идёт на пользу разработке. Поэтому некоторые из них будут выполняться совместно в одном объекте. Текущий факт несколько усложняет восприятие и понятность разработки в целом, однако значительно ускоряет её и снижает сложность, что имеет большую значимость в конкретно этой разработке.

Дополнительно стоить иметь в виду, что для реализации некоторых модулей использованы универсальные классы, обладающие большим функционалом, чем необходимо. Этот факт намного упростит дальнейшую модификацию подсистемы и позволит легко вносить какие-либо изменения. Это очень важное замечание, поскольку дальнейшая модификация подсистемы очень вероятна, учитывая изменяющиеся нормы к безопасности самолётов и сопутствующего оборудования. Чрезмерные усилия в этом стремлении могут погубить любой интересный и перспективный проект.

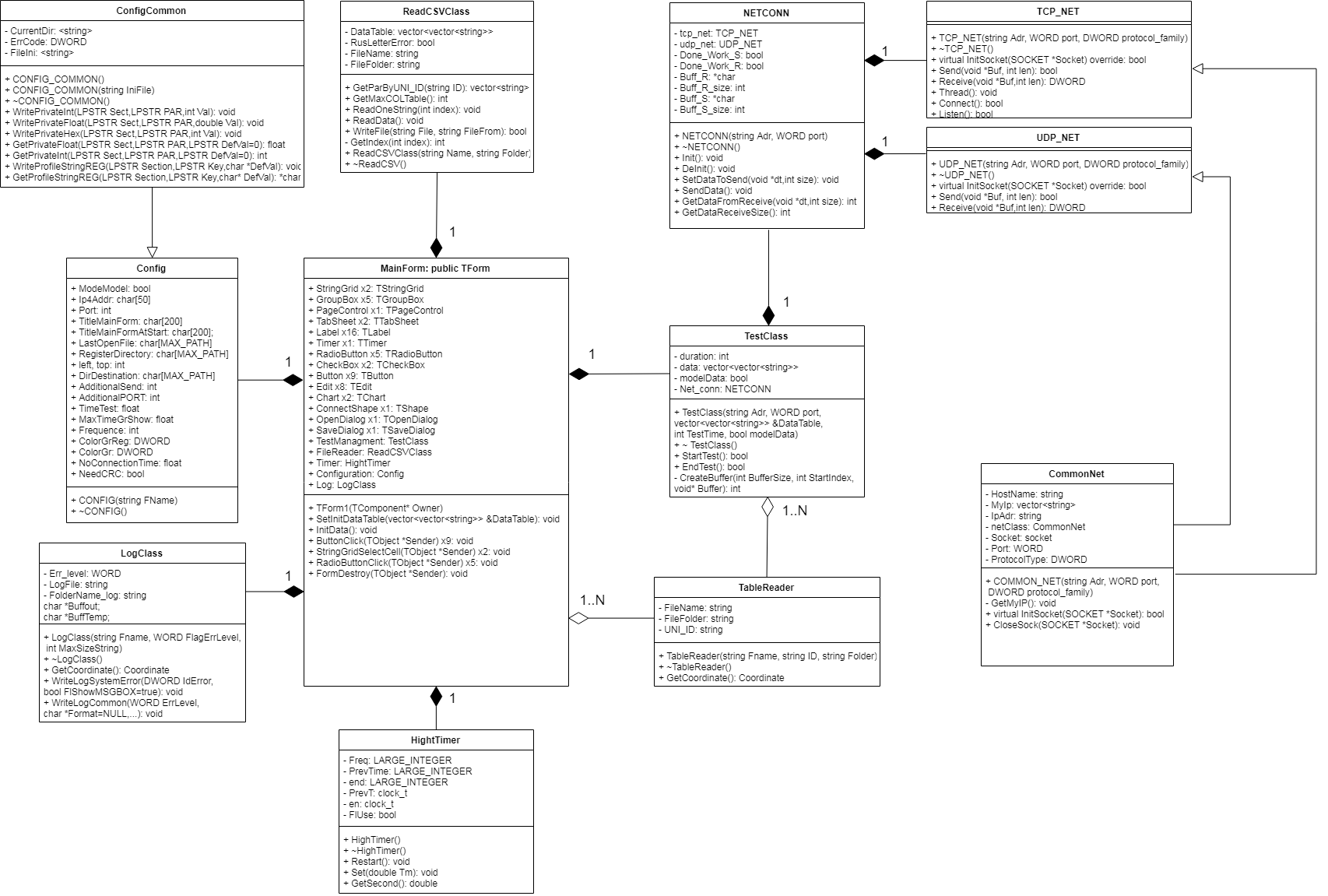


Рисунок 9 – Диаграмма классов

Как уже было сказано раньше главным является класс MainForm. Он унаследован от библиотечного класса TForm. Содержит в себе все остальные объекты. Время жизни этого объекта определяется временем работы всей программы. В него интегрирован модуль проверки наличия связи, модуль задания параметров самолёта, модуль отображения и модуль отрисовки. Это связано с тем, что объекты интерфейса, которые необходимы для взаимодействия с пользователем, тесно связаны с методами, выполняющими поставленные задачи и располагаются эти объекты интерфейса именно внутри разрабатываемого класса.

Для работы с файлом инициализации разработан специальный класс ConfigCommon, который позволяет осуществлять запись и чтение параметров программы. Он является универсальным. Класс для работы с файлом инициализации разрабатываемой программы называется Config и унаследован от вышеуказанного. В нём хранятся текущие параметры программы, которые сохранены в файле инициализации.

Для записи логов разработан класс LogClass. Он осуществляет запись действий пользователя в указанный файл. В дальнейшем данные из этого файла могут помочь в случае некорректной работы программы или некорректных действий пользователя, а также восстановить работоспособное состояние/конфигурацию подсистемы.

NETCONN – класс отвечающий за отправку и приём данных по сети. В данной подсистеме используется только отправка данных, поэтому универсальность разработанного класса упростит модификацию, связанную с необходимостью приёма данных. Содержит в себе два объекта, реализующих работу сетевыми данными по протоколам TCP/IP и UDP.

Классы TCP/IP\_NET и UDP\_NET унаследованы от одного виртуального класса и реализуют одинаковые функции, но по-разному. Используется либо один из них на выбор, либо оба по очереди (если при работе по первому протоколу произошёл сбой, то осуществляется попытка передачи по второму).

TestClass реализует проведение испытаний. Осуществляет формирование всех пакетов для отправки. Запускает и останавливает тест. Содержит в себе объект класса NETCONN, реализующий отправку сформированных массивов данных.

HightTimer класс для расчёта времени проведения теста. Разработан отдельно от интерфейса, чтобы добиться большей обособленности класса проведения испытаний от интерфейса.

## Разработка диаграммы компоновки

После синтеза диаграммы классов, необходимо решить, как выделенные классы будут размещать в файлах проекта. Разобраться в этом поможет диаграмма компоновки (рисунок 10).

Поскольку языком разработки является С++, то описанные в пункте 2.7 классы удобно хранить каждый в отдельном файле, причём для каждого класса будет выделено 2 файла: файл расширения .h, в котором находится описание класса (его поля и методы) и файл расширения .cpp, в котором находится реализация соответствующего класса. Также в разработанной диаграмме отмечены вспомогательные файлы: файл инициализации, файл для записи логов, файл для хранения параметров самолёта и файл для хранения табличных значений (подробнее про выбор способа хранение указанных данных можно прочитать в технологической части работы). Подключение одного класса осуществляется в хедере второго файла.

Распределение проекта на несколько файлов упрощает дальнейшую разработку и модификацию, а также делает работу более наглядной для других людей. В дополнение к вышеуказанному стоит заметить, что реализация кода не в одном файле способствует большей независимости отдельных модулей и повышает читаемость.

Подробная структура файлов, в которых хранится информация, в разработке текущей диаграммы не важна [11], но ознакомиться ней можно в пункте 3.1 настоящей работы.

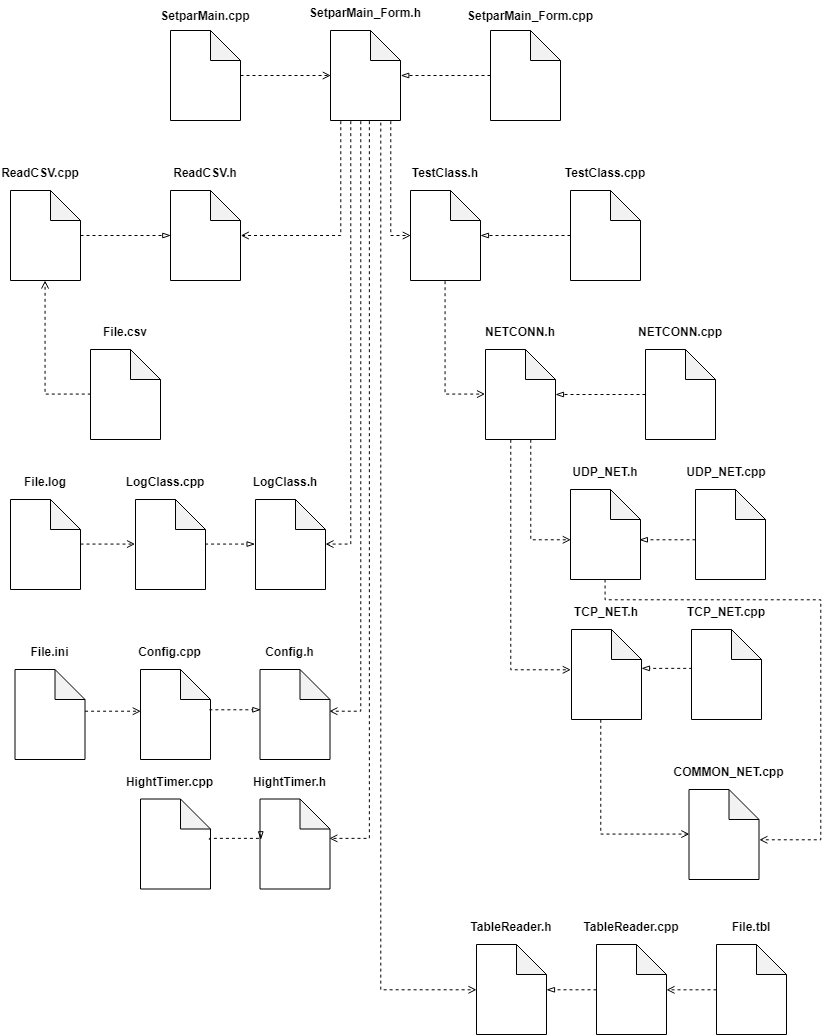


Рисунок 10 – Диаграмма компоновки

Выполнено оптимальное разбиение на файлы. Их не очень большое количество, поэтому в их множестве запутаться не получится, однако каждый из них играет большую роль в улучшении и структурировании кода всей программы.

## Алгоритм работы программы

Далее опишем общий алгоритм работы программы (рисунок 11). В нём отразится поведение программы при последовательных действиях пользователя, соответствующих инструкции проведения испытания. На данном уровне детализации алгоритм отображает только основные действия программы, однако далее каждое из них будет рассмотрено подробнее.



Рисунок 11 – Общий алгоритм работы программы

Первым действием после запуска программы является инициализация её компонентов. Подробнее процесс инициализации рассмотрен на рисунке 12. На данном этапе выполняется чтение файла инициализации, создание формы, попытка чтения файла параметров и наполнение таблиц, отображающих устройства, каналы и параметры считанными из файла данными.



Рисунок 12 – Схема алгоритма инициализации компонентов

Далее у испытателя есть выбор: загрузить новый файл параметров, внести изменения в текущие параметры или же запустить тест. Рассмотрим процесс задания параметров (рисунок 13).

Здесь пользователю доступны 4 варианта:

* выбрать канал;
* выбрать параметр;
* изменить параметр;
* сохранить изменения.



Рисунок 13 – Схема алгоритма задания параметров самолёта

Опишем механизм проведения тестов (рисунок 14). Для начала стоит отметить, что возможности пользователя сильно разняться в зависимости от того запущен тест или нет. Если тест не запущен, то пользователь имеет возможность выполнять настройку теста и его запуск, а также остальные действия, указанные в схеме основного алгоритма. Если при нажатии на кнопку запуска теста указаны некорректные параметры теста или какие-то ошибки с чтением файла, то пользователю выводится соответствующие сообщение.

Если тест запущен, то осуществляется ожидание окончания таймера для отправки команды завершения теста, также эта возможность доступна при нажатии на кнопку (досрочное завершение).

Стоит заметить, что выход из программы во время проведения теста недоступен. В случае аварийного выхода из программы, подсистема, с которой выполняется общение по сети не пострадает, так как в ней предусмотрен механизм отключения даже без сигнала от удалённого компьютера.



Рисунок 14 – Схема алгоритма проведения тестирования

Обобщим всё указанное выше и поэтапно опишем стандартный вариант пользования программой:

1. При загрузке программы выполняется чтение файла инициализации.
2. Создается основная форма и устанавливается ее местоположение на экране в соответствии с координатами, прописанными в файле инициализации.
3. Создаются потоки для работы по сети, с удаленной программой, для указанных в файле инициализации адреса и портов.
4. Выполняется попытка чтения последнего загруженного файла, прописанного в файле инициализации.
5. При успешном чтении файла с данными (CSV) формируются списки устройств, параметров принадлежащим данным устройствам и распределенным по каналам.
6. На форме заполняется таблица содержащая список устройств.
7. По выбранному элементу в первой таблице формируется вторая таблица, с распределением параметров по каналам. Для каждого параметра соответственно прописывается условное обозначение формы сигнала. Для таблично заданных пишется “TABL”, для смещения по времени – «FORMUL\_TIME», для заданных через kx+b – «FORMUL\_DAT», для синусоиды «SINUS». Для данных имеющих тип BNR или DW – вместо «CONST» записывается фактическое значение параметра, для всех остальных типов записывается «CONST».
8. По выбранному элементу второй таблицы, заполняется блок задания данных для конкретного параметра. В соответствии с формой сигнала заполняются те или иные параметры. При заполнении данных с описателем «TABL» выполняется чтение файла с таблицей и формирование графика, записанного в указанном файле. Для описателя «SINUS» строится график по заданным параметрам. Для частоты меньше 1 – строится на одну секунду по временной шкале, в противном случае график строится на один период.
9. Применение выполняется по факту изменения выбранной ячейки в таблице - "наименования устройств" или таблице-"названий параметров", также при изменении параметра активизируется кнопка «Сохранить».
10. Для выбора другого параметра или канала для данного устройства перемещаемся по второй таблице.
11. Для выбора другого устройства перемещаемся по первой таблице.
12. При нажатии на кнопку «старт», выполняется проверка на измененные, но не сохраненные данные, затем выполняется поиск всех зависимых файлов таблиц для текущего набора данных. После этого выполняется копирование текущего набора данных и зависимых файлов таблиц, в место, указанное в файле инициализации. После успешного копирования на удаленный компьютер, адрес которого указан в файле инициализации, отсылается пакет, указывающий на необходимость начала генерации сигналов. Если отмечен или не отмечен checkbox «Модельные данные», то отсылается соответствующая команда. При невозможности копирования файлов на экран выдается предупреждение, почему невозможно скопировать файлы и сообщение что процесс «старта» не выполнен.
13. При закрытии программы выполняется проверка на изменение исходных данных и в случае если этот так выполняется запрос на сохранение. Также, если тест не закончен, то на удаленный компьютер отсылается пакет, указывающий на необходимость остановки теста.
14. После сохранения данных обновляется файл инициализации в части: последнего открытого файла и положения основной формы на экране, а также времени «накопительного отображения» графиков на вкладке «отображение».

## Тестирование подсистемы

Проведём два типа тестирования (разработка технологии тестирования приведена в подразделе 3.2). Сначала проверим подсистему на наличие ошибок, после приступим к оценочному тестированию, целью которого является проверка соответствия основным требованиям (удобство использования, удобство эксплуатации и др.).

Для проверки подсистемы на наличие ошибок используем функциональное тестирование. В таблице 12 приведено общее описание проведённых тестов.

Таблица 12 – Перечень всех тестов подсистемы формирования параметров

|  |  |  |
| --- | --- | --- |
| Виды тестов | Количество тестов | Цель тестирования |
| Функциональное тестирование | 3 | Проверка реакции подсистемы на ввод корректных и не корректных данных |
| Оценочное тестирование | 3 | Проверка на соответствие основным требованиям |
| Тестирование на корректность работы с взаимосвязанными подсистемами | 1 | Проверка корректности взаимодействия с программой на удалённом компьютере |

### Функциональное тестирование

Проведём функциональное тестирование для задания параметров тестирования. Проверим работу с различными файлами параметров, и введёнными данными, а затем проверим полученный результат. Разработанные тесты представлены в таблицах 13-17.

Таблица 13 – Функциональное тестирование процедуры запуска теста

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Длительность теста | Файл | Ожидаемый результат | Полученный результат |
| 1 | Ничего не введено | Загружен читаемый файл | Сообщение с просьбой указать корректное время | Сообщение с просьбой указать корректное время |
| 2 | Введено некорректное время | Загружен читаемый файл | Сообщение с просьбой указать корректное время | Сообщение с просьбой указать корректное время |
| 3 | Введено корректное время | Файл не загружен | Сообщение с просьбой загрузить файл | Сообщение с просьбой загрузить файл |
| 4 | Введено корректное время | Загружен нечитаемый файл | Сообщение с просьбой проверить файл и загрузить новый | Сообщение с просьбой проверить файл и загрузить новый |
| 5 | Введено некорректное время | Файл не загружен | Сообщение о некорректности времени и файла | Сообщение о некорректности времени и файла |

Таким образом осуществляется защита от отправки некорректных данных и защита от указания неверных параметров тестирования.

В таблице 14 расположены результаты функционального тестирования процедуры редактирования параметров.

Таблица 14 – Функциональное тестирование процедуры редактирования

параметров

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Тест идёт | Выбор параметра | Тип параметра | Значение | Ожидаемый результат | Полученный результат |
| 1 | Нет | Параметр не выбран | Невозможно выбрать | Невозможно изменить | Поля изменения  недоступны | Поля изменения  недоступны |
| 2 | Нет | Параметр выбран | Любой | Вне диапазона мин макс | Сообщение о неверном значении параметра | Сообщение о неверном значении параметра |
| 3 | Нет | Любой | Таблица или сдвиг | Нечитаемый файл | Сообщение о неверном файле | Сообщение о неверном файле |
| 4 | Да | Недоступно | Недоступно | Недоступно | Корректная работа | Корректная работа |
| 5 | Нет | Любой | Любой | Корректные данные | Корректная работа | Корректная работа |

Предусмотрена защита от ввода неверных параметров, нечитаемых файлов.

Протестируем механизм загрузки/сохранения файлов (таблица 15).

Таблица 15 – Функциональное тестирование механизма загрузки/сохранения

файлов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Загружаемый файл | Сохранение | Ожидаемый результат | Полученный результат |
| 1 | Нечитаемый файл | Невозможно сохранить файл | Сообщение о некорректности загружаемого файла | Сообщение о некорректности загружаемого файла |
| 2 | Файл неверного формата | Невозможно сохранить файл | Сообщение о некорректности загружаемого файла | Сообщение о некорректности загружаемого файла |
| 3 | Не выбран | Невозможно сохранить файл | Недоступен основной функционал программы | Недоступен основной функционал программы |
| 4 | Корректный файл | Несуществующая директория | Предложение создать директорию | Предложение создать директорию |
| 5 | Корректный файл | Недоступная директория | Сообщение о невозможности сохранить файл по указанному пути | Сообщение о невозможности сохранить файл по указанному пути |

Предусмотрена защита от загрузки некорректных файлов, а также уведомление пользователя о невозможности сохранения файла.

При тестировании подсистемы методом черного ящика не было выявлено существенных нарушений. Несмотря на это, можно доработать программу для лучшего контроля ошибок пользователей, например:

* указывать причины при недоступности некоторых элементов (например, при наведении мыши);
* продумать отображение ошибок, возникающих при работе смежных подсистем;
* добавить возможность наглядной проверки связи для избегания ошибок при передаче данных (уже реализовано).

### Оценочное тестирование подсистемы

В настоящем разделе приведено тестирование трех видов:

* удобства использования;
* удобства эксплуатации;
* конфигурации оборудования.

Подробное описание назначения оценочного тестирования приведено в подразделе 3.2. Приступим к тестированию, целью которого является тестирование подсистемы на соответствие основным требованиям.

Программа была дана на использование 4 людям, которым предстоит работать с разработанной программой. Они осведомлены о её предназначении и разбираются в предметной области (что необходимо для её эксплуатации), а также ознакомлены с руководством пользователя. Сразу стоит заметить, что тестирование проводилось осведомлёнными людьми. Поскольку подсистема предполагается для эксплуатации в довольно узком направлении, то тестировать её обычными пользователями нет никакого смысла. Поэтому в тестирующую группу вошли люди, непосредственно которым предстоит работать с разрабатываемой программой. Они провели тестирование по каждому из критериев и выставили оценки. Результаты тестирования представлены в таблице 16.

Таблица 16 – Тестирование удобства эксплуатации

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерий | Варианты ответа | 1 человек | 2 человек | 3 человек | 4 человек | Итого |
| Наличие справочной информации | Да/Нет | Да | Да | Да | Да | 100% |
| Читаемость текста | Хорошо/ Плохо | Хорошо | Хорошо | Хорошо | Хорошо | 100% |
| Размер шрифта | Приятный/ Неприятный | Приятный | Неприятный | Приятный | Приятный | 75% |
| Расположение полей ввода и кнопок | Удобное/ Неудобное | Удобное | Неудобное | Удобное | Удобное | 75% |
| Цветовая гамма | Приятная/ Неприятная | Приятная | Неприятная | Приятная | Приятная | 75% |
| Понятность назначения кнопок выбора | Понятно/  Не понятно | Не понятно | Понятно | Понятно | Понятно | 75% |
| Понятность назначения кнопок | Понятно/ Не понятно | Понятно | Понятно | Понятно | Понятно | 100% |

По итогам тестирования подсистема удобна в эксплуатации на 85%. При этом программу можно улучшить, добавив более наглядное описание и справочную информацию по каждому параметру, однако это требует модификации архитектуры подсистемы.

Результаты оценочного тестирования по оставшимся критериям представлены в таблице 17.

Таблица 17 – Оценочное тестирование

|  |  |
| --- | --- |
| Критерий | Результат тестирования |
| Тестирование удобства использования | Сервис полностью соответствует техническому заданию, выполняет все заявленные функции. |
| Тестирование  конфигурации  оборудования | Проверка была осуществлена на 4 устройствах. Все они являются компьютерами РМИ. Именно на них установлена подсистема и будет проводится дальнейшая работа.  Конфигурации машин одинаковые. |

Также было необходимо разработать автоматические тесты для проверки корректности выполнения задач функциями. Однако большинство из них работают только с исходными данными (с данными из файла), поэтому был сформирован наиболее универсальный набор данных, способствующий проверке большинства классов подсистемы.

Также для проверки разработанной подсистемы была создана программа имитирующая приём данных. Она необходима для проверки сетевого взаимодействия.

Проверка передачи файлов и команд:

* 1. Запускаем Setpar\_main.exe
  2. Запускаем udp\_receive\_test.exe
  3. В Setpar\_main.exe нажимаем «старт», на экране приложения udp\_receive\_test видим в том числе:

IpAdress: 127.0.0.1

Send Port 0x2490 (32 bytes)

Receive Port 0x2454 [12 bytes]

……….  
RECEIVE:

Int : 3

Dbl : 200.00

Текущий результат показывает, что пришла команда '3' – запуск с модельными данными.

Если снять checkbox «Модельные данные» в Setpar\_main.exe, то придет команда «1».

RECEIVE:

Int : 1

Dbl : 200.00

Текущий результат показывает, что должны быть сформированы данные только из файла «set\_parametrs.csv».

При нажатии на кнопку «стоп» на экране отобразится   
RECEIVE:

Int : 2

Dbl : 200.00

Текущий результат показывает, что пришла команда «стоп».

Переходим в директорию c:\mera\_ini\_files\ там после нажатия кнопки «старт» должны появиться файлы «set\_parametrs.csv» и «test.tbl».

Если выбрать первый параметр, то можно увидеть его графическое отображение (рисунок 15).

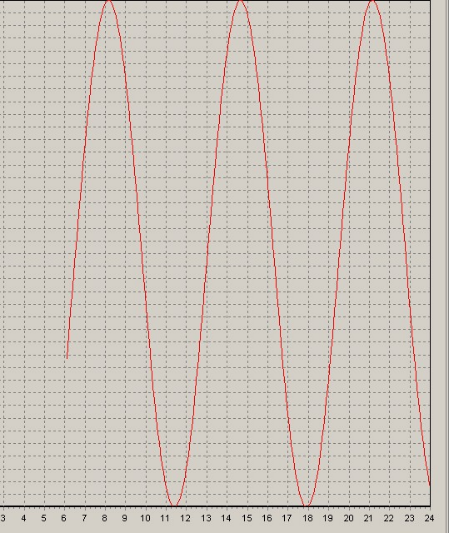


Рисунок 15 – Отображение графика параметра при тестировании

Если построить график этого же параметра в Excel, то можно увидеть точно такую же синусоиду

Проверка назначаемых параметров выполняется визуально:

1. На форме задания параметров выполняем какие угодно назначения, допустимые для данного набора данных
2. Сохраняем файл.
3. Заново открываем этот же файл и убеждаемся, что внесенные изменения присутствуют.

# Технологические решения

## Разработка технологий хранения данных

В качестве исходных данных разрабатываемая подсистема принимает 3 сущности: данные описания сигналов, данные описания таблиц, а также данные инициализации. В качестве выходных данных выступают логи.

В качестве данных инициализации было принято решение использовать стандартный .ini файл, что позволит с легкостью настроить программу для работы на различных компьютерах и с различными файлами.

Данные описания сигналов используются не только в данной подсистеме, но и в подсистеме формирования сигналов. Она работает с .csv файлами со специальным внутренним форматом. Поэтому было принято решение использовать именно данный способ хранения данных, чтобы иметь возможность с минимальными затратами создать и отправить сформированный массив данных. Также подобный тип хранения описания сигналов позволяет легко и без постороннего ПО внести в структуру хранения данных описание новых сигналов.

Аналогичная ситуация обстоит и с данными, описывающими таблицы. Существует уже установленный формат, который используется другими подсистемами, поэтому в разрабатываемом ПО он будет таким же.

Для хранения логов также выбран формат файла, что упрощает и запись логов, и их доступность в случае непредвиденной ситуации.

Подробное описание формата и хранимых данных указано ниже.

### Структура файла хранения параметров

В данном случае описываются только те поля, которые читаются в программе (рисунок 17). Общая структура файла составлена следующим образом: за основу взята таблица из документации по КСУ, к ней дописаны необходимые для работы программы колонки и строки.

Первая строка – это описание данного набора задаваемых сигналов, выводится на форме как справочная информация

Вторая и третья строки описывают заголовки колонок:

* «Устройство» (например): БРУД или УБРУ, по данному названию формируется список устройств и определяется принадлежность того или иного канала определенному устройству;
* «Канал» – название канала (отображается в названии колонки канала), по данному параметру происходит определение, в какую колонку определить текущий параметр;
* «UNI\_ID\_CHAN» – уникальный идентификатор канала, по данному параметру определяется по какой цифровой линии будет выдаваться информация. В связи с возможностью задания русских букв, а также с тем что именно такой же идентификатор необходимо использовать на удаленном компьютере, выполняется дополнительная проверка на наличие русских букв. В случае обнаружения выдается предупреждение как на рисунке 16;

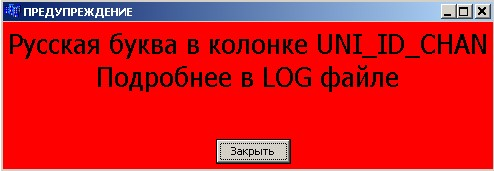


Рисунок 16 – Предупреждение о наличии русских букв

* «Тип сигнала» – описывается тип сигнала, в программе предусмотрен анализ двух типов «ARINC» и «DISKR»;
* «Форма сигнала» – данная колонка описывает форму сигнала, в программе выполняется анализ на следующие значения:
  1. «TABL» - форма сигнала задана табличной функцией,
  2. «CONST» - форма сигнала постоянное значение,
  3. «FORMUL\_TIME» - форма сигнала это смещенный по времени другой сигнал,
  4. «FORMUL\_DATE» - форма сигнала это уравнение заданное как kx+b, где x – это другой сигнал,
  5. «SINUS» - форма сигнала имеет синусоидальную форму, с доступными параметрами сдвига по вертикали, амплитуды и частоты.
* параметры сигнала- данная колонка описывает параметры сигнала, интерпретация данной колонки зависит от «формы сигнала»:
  1. Значение = «CONST» - это значение постоянного сигнала,
  2. Значение = «TABL» - это имя файла \*.tbl, в котором форма сигнала задана табличной функцией от времени,
  3. Значение = «FORMUL\_TIME» - запись имеет вид «PAR\_ID0566,1.000000», где PAR\_ID0566 – это уникальный идентификатор параметра (UNI\_ID\_PAR), через «,» к нему дописано положительное смещение по времени, заданное в секундах,
  4. Значение = «FORMUL\_DATE» - запись имеет вид «PAR\_ID0566,1.000000,0.0000», где PAR\_ID0566 – это уникальный идентификатор параметра (UNI\_ID\_PAR), через «,» к нему дописано значение коэффициента «k», далее через «,» дописано значение коэффициента «b»,
  5. Значение = «SINUS» - запись имеет вид 50.000, 200.000000, 1.000000, где 50 – это сдвиг по вертикали, 200 – это амплитуда сигнала, а 1.0000 - это частота сигнала.
* UNI\_ID\_PAR – уникальный идентификатор параметра, предназначен для ссылок на данный параметр;
* Название параметра – имя параметра, по данной колонке формируется список параметров, передаваемых в конкретный канал (цифровую линию);
* Описание параметра – используется как первая часть комментария к параметру;
* Тип параметра - тип передаваемых данных, в зависимости от значения, по-разному интерпретируются значения, заданные как «CONST». Используется как часть комментария к описанию параметра:
  1. BCD – двоично десятичное число (задается как целое),
  2. ISO 5 – символ (задается как целое),
  3. BNR – число с плавающей точкой, в таблице на форме отображается как число вместо надписи «CONST»,
  4. DW – битовое значение, может иметь значение только 0 или 1.
* Ед. измерения – используется как часть комментария к описанию параметра BCD, ISO 5, BNR, DW;
* Физ. диап., Мин – используется как часть комментария к описанию параметра;
* Физ. диап., Макс – используется как часть комментария к описанию параметра;
* Примечания – используется как вторая часть комментария к параметру.

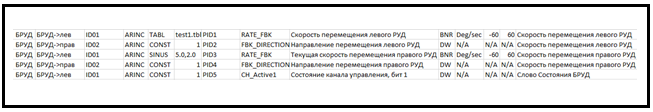


Рисунок 17 – Пример хранения параметров самолёта в файле

### Структура файла хранения описания таблиц

1 строка – это описатель файла, выводится на форме как справочная информация по таблице (рисунок 18).

Число1; Число2

Число1- это время в секундах

Число2 – значение параметра на указанное время.

Промежуточные значения считаются по линейной интерполяции.

Количество строк произвольное.

Первый параметр должен идти по возрастанию. В программе возрастание первого параметра не проверяется.

Проще всего таблицу подготавливать в Excel, там же можно сразу смотреть и график изменения параметра.

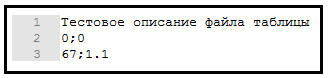


Рисунок 18 – Пример хранения параметров табличных данных

### Файл инициализации set\_parametrs.ini

[DEFAULT]

LastOpenFILE=E:\source\work\mc21\set\_parametrs\set\_par.csv – последний открываемый файл (рисунок 19). Поддерживается как абсолютный, так и относительный путь – если записать его вручную, т.к. в автоматическом режиме записывается всегда абсолютный путь.

[POSITION]

LEFT=199 левый край основный формы

TOP=49 – верхний край основной формы

[NET]

DirDestination=\\127.0.0.1\guest – директория куда будут скопированы файлы с заданием формы сигнала, а также все файлы, содержащие таблицы

Ip4Addr=127.0.0.1 – адрес удаленного компьютера на котором запускается программа имитации сигналов

PORT=0x2454 – порт для связи с удаленной программой по протоколу UDP

AdditionalPORT=0x2490 – порт для получения информации по параметрам для регистрациив файлы CSV и отображения графиков на экране, адрес используется Ip4Addr

AdditionalSend=1 – флаг указывающий на возможность принятия данных моделирования по сети (1-есть, 0 -нет). В зависимости от этого флага скрывается вкладка «отображение»

[INTERFACE]

TimeTest=200.000000 – время выполнения теста

MaxTimeGrShow=24.000000 - масимальное время жизни графика (т.е. длина временной оси от минимума до максимума)

Frequence=10 - интервал обновления графиков на вкладке «отображение», в мсек

RegisterDirectory=E:\source\work\mc21\set\_parametrs\ - место сохранение файлов регистрации, это имя директории куда будут складываться файлы в формате CSV

ColorGrReg=0x0000ff цвет графиков со статусом "REGISTRATION"

ColorGr=0xff0000 - цвет остальных графиков

TitleMainForm – здесь задается заголовок основной формы

TitleMainFormAtStart – здесь задается часть заголовка основной формы, если при нажатии кнопки «старт» отмечен checkbox (модельные данные).

ChName=М11.3W\_02 – задает имя физического канала из колонки [UNI\_ID\_CHAN]

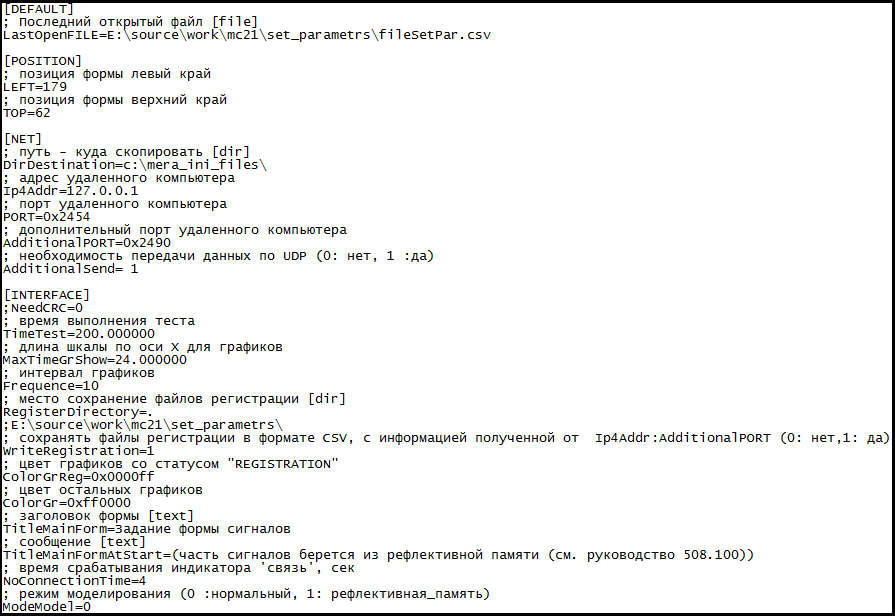


Рисунок 19 – Пример хранения данных в файле инициализации

## Разработка технологии тестирования подсистемы

Тестирование системы – проверка соответствия системы заявленным характеристикам, сравнение ожидаемого результата с полученным.

Несмотря на доступ к исходному коду было принято решение проведения тестирования методом функционального тестирования [12] («чёрный ящик»), без структурного тестирования («белый» или «серый ящик»). Таким образом, можно достичь наибольшей схожести использования программы реальным человеком и выявить самые явные и неожиданные недоработки подсистемы.

Из-за наличия доступа к исходному коду и знания возможных «слабых» мест подсистемы, было принято решение использовать в функциональном тестировании подход «предположение о тестировании». Более подробное описание способа проверки системы приведено в таблице 18.

Таблица 18 – Описание подхода функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Данные ввода | Ожидаемый результат | Полученный результат |
| Порядковый номер теста | Некорректные данные, удовлетворяющие предположенной ошибке | Результат, который должна выдать подсистема, в соответствии с техническим заданием и основным требованиям | Реальный результат, предоставляемый подсистемой |

Для проверки системы на соответствие основным требованиям необходимо провести оценочное тестирование, которое также называют «тестированием системы в целом». Для данной подсистемы целесообразно провести тестирование трёх видов, описанных ниже, так как они позволяют оценить его основные потребительские характеристики:

* тестирование удобства использования - последовательная проверка соответствия сервиса и документации на него основным положениям технического задания;
* тестирование удобства эксплуатации - анализ психологических факторов, возникающих при работе с программным обеспечением; это тестирование позволяет определить на сколько удобен интерфейс, на сколько правильно подобрано цвето-звуковое сопровождение и т. п.;
* тестирование конфигурации оборудования - проверка работоспособности сервиса на разном оборудовании.

Для проверки системы на удобство использования необходимо использовать шаблон, указанный в таблице 19. Так как он позволяет получить более объективное мнение пользователей о разработанном сервисе.

Таблица 19 – Шаблон для тестирования удобства использования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Варианты ответа | 1 человек | N человек | Итого |
| В каждой строке необходимо вписать один из наиболее важных критериев, по которым контрольная группа может высказать свое мнение, например «Расположение полей ввода и кнопок» | Описание вариантов ответа для контрольной группы, например, «Удобно/Не удобно» | Ответ конкретного человека по данному критерию | Ответ конкретного человека по данному критерию | Сумма  положительных мнений делится на N и умножается на 100% |

В таблице 20 описаны необходимые действия для тестирования удобства эксплуатации и конфигурации оборудования.

Таблица 20 – Способ тестирования удобства использования и конфигурации

оборудования

|  |  |
| --- | --- |
| Критерий | Способ |
| Тестирование удобства использования | Проверка сервиса на соответствие техническому заданию. |
| Тестирование  конфигурации  оборудования | Проверка корректности работы системы на N устройствах. |

Целью всех этих проверок является поиск несоответствий техническому заданию, проверка работоспособности подсистемы на компьютерах различной конфигурации, а также удобства использования.

# ЗАКЛЮЧЕНИЕ

В соответствии с техническим заданием была разработана подсистема формирования параметров самолёта для проведения тестовых испытаний в качестве выпускной квалификационной работы.

Подсистема способствует автоматизации и упрощению процесса наземного испытания летательных аппаратов. Анализ существующей системы, которая нуждалась в подобной разработке, позволил предусмотреть возможные проблемы и дополнительные требования, способствующие улучшению процесса задания параметров. Интерфейс программы наглядно отображает возможности подсистемы. Проверка вводимой информации позволяет обеспечить частичную корректность данных, передаваемых смежным подсистемам.

Разработанная подсистема имеет дальнейшие перспективы для развития и модификаций. На полезности системы сильно отразится добавление возможности приёма и отображения уведомлений от программы на удалённом компьютере, добавление возможности создания и редактирования файлов, содержащих табличные значения. Также пойдёт на пользу интегрирование подсистемы отображения состояния параметров в режиме реального времени и разработка механизма изменения параметров во время проведения теста.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Теоретические основы испытаний и экспериментальная отработка сложных технических систем / Л.Н. Александровская, В.И. Круглов, А.Г. Кузнецов и др.: Учеб. пособие. М.: Логос, 2003. - Гл. 3. - С.40-64.
2. Михаил Григорьевич Котик, Алексей Васильевич Павлов, Игорь Михайлович Пашковский, Николай Григорьевич Щитаев // Летные испытания самолётов, Издательство «Машиностроение», Москва, 1998
3. Испытательный стенд МС-21 [Электронный ресурс] // URL: http://mc-21.wikidot.com/news:439 (дата обращения 15.03.2019).
4. Модели жизненного цикла программного обеспечения [Электронный ресурс] // Хабр. URL: https://habr.com/ru/post/111674/ (дата обращения 11.02.2019).
5. Microsoft Visual Studio Community Edition [Электронный ресурс] // Wikipedia – Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio (дата обращения: 02.05.2019).
6. Qt Creator [Электронный ресурс] // Wikipedia – Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Qt\_Creator (дата обращения: 01.05.2019).
7. C++ Builder [Электронный ресурс] // Wikipedia – Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/C%2B%2B\_Builder (дата обращения: 05.05.2019).
8. Eclipse [Электронный ресурс] // Wikipedia – Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Eclipse (дата обращения: 05.05.2019).
9. Проектирование графического интерфейса пользователя [Электронный ресурс] // Хабр. URL: https://habr.com/ru/post/208966/ (дата обращения 11.02.2019).
10. Отношения классов – от UML к коду [Электронный ресурс] // Хабр. URL: https://habr.com/ru/post/150041/ (дата обращения 10.03.2019).
11. Иванова Г.С. Технология программирования. М. : Изд-во КНОРУС, 2011.
12. Тестирование. Фундаментальная теория [Электронный ресурс] // Хабр. URL: https://habr.com/ru/post/279535/ (дата обращения 11.02.2019).

# ПРИЛОЖЕНИЕ А

Техническое задание

Листов 7

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технический университет

имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатики и управления»

Кафедра «Компьютерные системы и сети»

УТВЕРЖДАЮ

Зав. кафедрой ИУ6,

д.т.н., проф.\_\_\_\_\_\_\_Пролетарский А.В.

" \_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

ПОДСИСТЕМА ФОРМИРОВАНИЯ ПАРАМЕТРОВ САМОЛЁТА ДЛЯ ПРОВЕДЕНИЯ ТЕСТОВЫХ ИСПЫТАНИЙ

Техническое задание

Листов 8

Исполнитель студ. гр. ИУ6-81Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. А. Веинский**

(Подпись, дата) (И.О.Фамилия)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Ю. Еремин**

(Подпись, дата) (И.О.Фамилия)

Москва, 2019

1 Введение

Настоящее техническое задание распространяется на разработку подсистемы формирования параметров самолёта для проведения наземных испытаний, предназначенной для функционирования испытательного стенда.

Актуальность данной темы заключается в том, что испытателю необходим графический интерфейс, чтобы иметь возможность вносить изменения в текущие параметры самолёта. Подобное программное обеспечение необходимо на каждом испытательном стенде и индивидуально дорабатывается под каждый самолёт.

2 Основания для разработки

Подсистема формирования параметров самолёта для проведения тестовых испытаний разрабатывается на основании инициативы исполнителя выпускной квалификационной работы.

3 Назначение разработки

Основное назначение подсистемы формирования параметров самолёта для проведения тестовых испытаний заключается в повышении надёжности разрабатываемых самолётов, путём увеличения удобства системы тестирования и предоставления возможности проведения более полноценных и наглядных тестов. Испытатель получает возможность вносить изменения в текущие параметры самолёта и управлять проведением испытаний на удалённом компьютере.

4 Исходные данные, цели и задачи

4.1. Исходные данные

Исходными данными для разработки являются описание функционального и эксплуатационного назначения.

4.2. Цель работы

Целью работы является разработка программной подсистемы формирования параметров самолёта для проведения тестовых испытаний.

4.3. Решаемые задачи

4.3.1. Выбор подхода, технологии, методов, стандартов и средств разработки.

4.3.2. Анализ требований технического задания с точки зрения выбранной технологии и уточнение требований к подсистеме: техническим средствам, внешним интерфейсам.

4.3.3. Исследование предметной области – анализ существующих систем тестирования летательных аппаратов, изучение методик, оборудования и программного обеспечения для проведения стендовых испытаний.

4.3.4. Определение архитектуры подсистемы: разработка её структуры и процессов обслуживания, анализ смежных подсистем.

4.3.5. Анализ требований технического задания и разработка спецификаций проектируемого программного обеспечения.

4.3.6. Разработка структуры программного обеспечения.

4.3.7. Проектирование компонентов программного продукта.

4.3.8. Реализация компонентов программного продукта с использованием выбранных средств и функциональное тестирование.

4.3.9. Сборка программного обеспечения и его комплексное тестирование.

4.3.10. Оценочное тестирование программного обеспечения, в частности тестирование удобства использования.

5 Требования разрабатываемой ПОДсистеме

5.1. Требования к функциональным характеристикам

5.1.1. Выполняемые функции

5.1.1.1. Для пользователя:

* просмотр параметров самолёта;
* выбор типа значения параметра самолёта;
* изменение значения параметра самолёта;
* загрузка файла параметров;
* сохранение внесённых изменений;
* запуск и остановка проведения испытания.

5.1.2. Исходные данные:

* Файл инициализации set\_parametrs.ini;
* Файлы описания параметров. Формат данного файла соответствует стандартному формату CSV, для MS Excel;
* Файлы описания таблиц \*.tbl.

5.1.3. Результаты:

* файл описания параметров;
* файл, содержащий логи \*.log;
* запуск программы генерации сигналов на удалённом компьютере.

5.2. Требования к надежности

5.2.1. Предусмотреть контроль вводимой информации.

5.2.2. Предусмотреть защиту от некорректных действий пользователя.

5.2.3. Предусмотреть проверку загружаемых файлов на предмет совместимости с разработанным программным обеспечением.

5.3. Условия эксплуатации

5.3.1. Условия эксплуатации в соответствие с СанПиН 2.2.2/2.4.1340-03.

5.3.2. Обслуживание

5.4. Требования к составу и параметрам технических средств

5.4.1. Программное обеспечение должно функционировать на устройствах с операционной системой Windows.

5.4.2. Минимальная конфигурация технических средств:

5.4.2.1. Тип процессора Intel Core i3 и новее

5.4.2.2. Объем ОЗУ 2 Гб

5.5. Требования к программной совместимости

5.5.1. Программное обеспечение должно работать под управлением операционной системы Windows 7.

5.5.3. Язык разработки приложения: C++.

5.6. Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

5.7. Требования к транспортировке и хранению

Требования к транспортировке и хранению не предъявляются.

5.8. Специальные требования

Сгенерировать установочную версию программного обеспечения.

6 Требования к документации

6.1. Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

6.2. В состав сопровождающей документации должны входить:

6.2.1. Расчетно-пояснительная записка на 60-80 листах формата А4 (без приложений 6.2.2, 6.2.3, 6.2.4 и 6.2.5).

6.2.2. Техническое задание (Приложение A).

6.2.3. Руководство пользователя (Приложение Б).

6.2.4. Исходный текст программы (Приложение В).

6.2.5. Копии листов графической части (Приложение Г).

6.3. Графическая часть должна быть выполнена на 6 листах формата А1 (копии формата А4 включить в качестве приложения к расчетно-пояснительной записке):

6.3.1. Схема модуля управления стендом.

6.3.2. Диаграмма вариантов использования.

6.3.3 Структурная схема подсистемы формирования параметров.

6.3.4 Граф состояния интерфейса, формы интерфейса.

6.3.5. Диаграмма классов.

6.3.6. Схема алгоритма работы программы.

7 Технико-экономические показатели

Выполнить технико-экономическое обоснование разработки.

8 Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок,  даты, % | Отчетность |
| 1. | Анализ предметной области, выбор технического решения | 01.01.2019-15.02.2019  10% | Техническое задание |
| 2. | Анализ возможных способов использования программы | 16.02.2019-28.02.2019  10% | Диаграмма вариантов использования |
| 3. | Проектирование интерфейса программы | 01.03.2019-01.04.2019  35% | Формы интерфейса, диаграмма состояний интерфейса |
| 4. | Проектирование программного обеспечения | 02.04.2019-16.04.2019  15% | Схема структурная, диаграмма классов, схема алгоритма работы программы |
| 5. | Тестирование системы | 17.04.2019-30.04.2019  10% | Тесты, результаты тестирования. |
| 6. | Оформление проектной документации | 01.05.2019-19.05.2019  10% | Остальная проектная документация. |
| 7. | Подготовка доклада и предзащита | 20.05.2019- 01.06.2019  5 % | Доклад |
| 8. | Защита проекта | 100 % |  |

9 Порядок контроля и приемки

9.1. Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

9.2. Порядок защиты

Защита осуществляется перед государственной экзаменационной комиссией (ГЭК).

9.3. Срок защиты

Срок защиты определяется в соответствии с планом заседаний ГЭК.

10 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.

# ПРИЛОЖЕНИЕ Б

Руководство пользователя

Листов 4

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

**Руководство пользователя**

**Подсистемы формирования параметров самолёта для проведения тестовых испытаний**

Студент гр. ИУ6-81 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Веинский В.А.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКРБ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ерёмин О. Ю.**

(Подпись, дата) (И.О.Фамилия)

**СОДЕРЖАНИЕ**

**1 Общее сведения о подсистеме …………………… ………………………79**

**2 Описание запуска……………………………………………………………79**

**3 Работа в программе …………………………………………………………79**

**1 Общие сведения о подсистеме**

Основная задача подсистемы формирования параметров самолёта для проведения тестовых испытаний редактировать параметры систем самолёта и контролировать проведение испытаний.

**2 Описание запуска**

При запуске программы происходит инициализация рабочей формы в соответствии с настройками при прошлом запуске (если не были проведены вручную изменения в файле инициализации). В случае успешной загрузки последнего используемого файла, на экране пользователя появляется список устройств и каналов, содержащих параметры и их значения.

Рисунок 1 – Пример инициализированной таблицы каналов

**3 Работа в программе**

После загрузки файла параметров пользователь имеет следующие возможности:

* выбрать один из каналов для просмотра и/или редактирования параметров;
* установить параметры проведения теста;
* начать проведение испытания.

Рисунок 2 – Таблица параметров выбранного канала

Рисунок 3 – Окно управления проведением теста

При редактировании параметров доступны следующие типы параметров:

* константное значение;
* табличное значение;
* функция синуса;
* функция прямой;
* сдвиг по времени.

Рисунок 4 – Окно задания параметров

После запуска испытания, возможно контролирование времени его проведения, а также досрочное завершение.

Рисунок 5 – Окно управления ходом тестирования

# ПРИЛОЖЕНИЕ В

Фрагменты исходного кода

Листов 15

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

**Фрагменты исходного кода**

**Подсистемы формирования параметров самолёта для проведения тестовых испытаний**

Студент гр. ИУ6-81 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Веинский В.А.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКРБ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ерёмин О.Ю.**

(Подпись, дата) (И.О.Фамилия)

Листинг 1 – Модуль логирования

|  |
| --- |
| #include <windows.h>  #include "WriteLog.h"  #include <dos.h>  #include <conio.h>  #include <string.h>  Log \*my\_LOG;  Log::Log(char \*Fname,WORD FlagErrLevel,int MaxSizeString, bool DeletePrev)  {  if(!FlagErrLevel) Err\_level=ERR\_CRIT|ERR\_OTHER|ERR\_WARN4|ERR\_INFO4;  else Err\_level=FlagErrLevel|ERR\_CRIT|ERR\_OTHER;  sprintf(Name\_log,Fname);  if(DeletePrev) remove(Name\_log);  Buffout=new char[MaxSizeString];  BuffTemp=new char[MaxSizeString];  WriteLogCommon(ERR\_INFO1,"sx","Уровень подробности LOG файла:",Err\_level);  my\_LOG=this;  }  // =========================================  void Log::Print2Str(FILE \*to,char \*\_1st,char \*\_2nd)  {  fprintf(to,"%s ",\_1st);  fprintf(to,"%s\n",\_2nd);  }  // =========================================  void Log::WriteLogCommon(WORD Err,LPSTR Format,...)  {  if((Err & Err\_level)!=Err) return;  if(!Buffout) return;  SYSTEMTIME ST;  GetLocalTime(&ST);    out\_log=fopen(Name\_log,"rt+");  if(!out\_log) out\_log=fopen(Name\_log,"wt");  fseek(out\_log,0,SEEK\_END);  fprintf(out\_log,"%04d.%02d.%02d %02d:%02d:%02d.%03d ",  ST.wYear,ST.wMonth,ST.wDay,ST.wHour,ST.wMinute,ST.wSecond,ST.wMilliseconds);  char \*TextInfo;  if(Err&ERR\_CRIT) TextInfo="ERR\_CRITICAL";  else if(Err&ERR\_OTHER) TextInfo="ERROR\_OTHER\_";  else if(Err&ERR\_WARN4) TextInfo="WARNING\_\_\_\_\_";  else if(Err&ERR\_INFO4) TextInfo="INFORMATION\_";  if(!Buffout) fprintf(out\_log,"%s\n","Buffout память не выделена");  if(!BuffTemp) fprintf(out\_log,"%s\n","BuffTemp память не выделена");  Buffout[0]=0;  va\_list ap;  va\_start(ap,Format);  long N\_var;  LPSTR S\_var;  double F\_var;  char C\_var;  while(1)  {  switch(\*Format++)  {  case 0:  va\_end(ap);  Print2Str(out\_log,TextInfo,Buffout);  fclose(out\_log);  return;  case 'c':  C\_var=va\_arg(ap,char);  sprintf(BuffTemp,"%c",C\_var);  break;  case 'x':  N\_var=va\_arg(ap,long);  sprintf(BuffTemp," 0x%08x ",N\_var);  break;  case 'd':  N\_var=va\_arg(ap,long);  sprintf(BuffTemp," %d ",N\_var);  break;  case 'f':  F\_var=va\_arg(ap,double);  sprintf(BuffTemp," %.3f ",F\_var);  break;  case 'p':  N\_var=va\_arg(ap,long);  sprintf(BuffTemp," 0x%08x ",N\_var);  break;  case 's':  S\_var=va\_arg(ap,LPSTR);  sprintf(BuffTemp," %s ",S\_var);  break;  default:  sprintf(BuffTemp," ???");  }  strcat(Buffout,BuffTemp);  }  }  // =========================================  void Log::WriteLogSystemError(DWORD IdError,bool FlShowMSGBOX)  {  if(!IdError) IdError=GetLastError();  if(!IdError) IdError=WSAGetLastError();  if(!IdError) return;  LPVOID lpMsgBuf;  if(FormatMessage(FORMAT\_MESSAGE\_ALLOCATE\_BUFFER|FORMAT\_MESSAGE\_FROM\_SYSTEM|FORMAT\_MESSAGE\_IGNORE\_INSERTS,  NULL,  IdError,  MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT), // Default language  (LPTSTR) &lpMsgBuf,0,NULL))  { WriteLogCommon(ERR\_CRIT,"ssd",lpMsgBuf,"код ошибки",IdError);  if(FlShowMSGBOX)  MessageBox(NULL,(LPCTSTR)lpMsgBuf,"SYSTEMERROR",MB\_OK|MB\_ICONINFORMATION|MB\_SYSTEMMODAL);  LocalFree(lpMsgBuf);  }  }  // ========================================= |

Листинг 2 – Модуль чтения файла инициализации

|  |
| --- |
| #ifndef SET\_PARAMETERS\_CONFIG\_HEADER  #define SET\_PARAMETERS\_CONFIG\_HEADER  #include "config\_common.h"  #define CNT\_SUOSO\_DESCR 4  typedef struct  {  char MaskNamePar[100];  char ChName[40];  }SUOSO\_Str;  class CONFIG:public CONFIG\_COMMON  {  public:  bool ModeModel; // исппользовать данные из модели (это признак что надо отсылать MODE\_START\_MODEL)  char Ip4Addr[50];  int Port;  char TitleMainForm[200];  char TitleMainFormAtStart[200];  char LastOpenFile[MAX\_PATH];  char RegisterDirectory[MAX\_PATH];  int left,top;  char DirDestination[MAX\_PATH];  int AdditionalSend;  int AdditionalPORT;  float TimeTest;  float MaxTimeGrShow; // время накопительного отображения графиков  int Frequence; // интервал задержки в мсек для рисования графиков  bool WriteRegistration; // создавать файлы CSV с регистрацией параметров  bool ShowPageSUOSO;  // bool ShowPageIMM;  DWORD ColorGrReg;  DWORD ColorGr;  SUOSO\_Str SUOSO\_dscr[CNT\_SUOSO\_DESCR];  float NoConnectionTime; // время в сек, через которое если нет приема счетчика, будет показана нзапись "НЕТ СВЯЗИ"  bool NeedCRC;  CONFIG(char \*FName):CONFIG\_COMMON(FName)  {  char TempBf[20];  GetPrivateProfileString("INTERFACE","NeedCRC","1",TempBf,20,FileIni);  NeedCRC=atoi(TempBf);  NoConnectionTime=GetPrivateFloat("INTERFACE","NoConnectionTime","4");  ModeModel=GetPrivateFloat("INTERFACE","ModeModel","0");  GetPrivateProfileString("INTERFACE","TitleMainFormAtStart","",TitleMainFormAtStart,200,FileIni);  WritePrivateProfileString("INTERFACE","TitleMainFormAtStart",TitleMainFormAtStart,FileIni);  GetPrivateProfileString("INTERFACE","TitleMainForm","",TitleMainForm,200,FileIni);  WritePrivateProfileString("INTERFACE","TitleMainForm",TitleMainForm,FileIni);  GetPrivateProfileString("NET","DirDestination","NULL",DirDestination,MAX\_PATH,FileIni);  GetPrivateProfileString("NET","Ip4Addr","127.0.0.1",Ip4Addr,50,FileIni);  Port=GetPrivateInt("NET","Port","0x4556");  GetPrivateProfileString("DEFAULT","LastOpenFILE","",LastOpenFile,MAX\_PATH,FileIni);  GetPrivateProfileString("INTERFACE","RegisterDirectory","",RegisterDirectory,MAX\_PATH,FileIni);  if(!strlen(RegisterDirectory))  {  GetCurrentDirectory(\_MAX\_PATH,RegisterDirectory);  WritePrivateProfileString("INTERFACE","RegisterDirectory",RegisterDirectory,FileIni);  }  int len=strlen(RegisterDirectory)-1;  while(len && (RegisterDirectory[len]=='\\' || RegisterDirectory[len]=='/'))  { RegisterDirectory[len]=0; len--; }  left=GetPrivateInt("POSITION","LEFT","10");  top=GetPrivateInt("POSITION","TOP","10");  AdditionalSend=GetPrivateInt("NET","AdditionalSend","0");  AdditionalPORT=GetPrivateInt("NET","AdditionalPORT","0x4560");  TimeTest=GetPrivateFloat("INTERFACE","TimeTest","100");  MaxTimeGrShow=GetPrivateFloat("INTERFACE","MaxTimeGrShow","100");  Frequence=GetPrivateInt("INTERFACE","Frequence","10");  if(Frequence<=0) Frequence=1000;  if(MaxTimeGrShow<1) MaxTimeGrShow=1;  ShowPageSUOSO=GetPrivateInt("INTERFACE","ShowPageSUOSO","0");  //ShowPageIMM=GetPrivateInt("INTERFACE","ShowPageIMM","0");  WriteRegistration=GetPrivateInt("INTERFACE","WriteRegistration","0");  ColorGrReg=GetPrivateInt("INTERFACE","ColorGrReg","0x0000ff");  ColorGr=GetPrivateInt("INTERFACE","ColorGr","0xff0000");  for(int i=0;i<CNT\_SUOSO\_DESCR;i++)  {  char SectNm[20];  sprintf(SectNm,"SUOSO\_STR%d",i+1);  GetPrivateProfileString(SectNm,"Name","NoNаme",SUOSO\_dscr[i].MaskNamePar,100,FileIni);  GetPrivateProfileString(SectNm,"ChName","NoNаme",SUOSO\_dscr[i].ChName,40,FileIni);  }  char Bf[100];  GetPrivateProfileString("SUOSO\_STR5","Name","NoNаme",Bf,100,FileIni);  if(strcmp(Bf,"NoNаme"))  MessageBox(NULL,"ТТребуется заменить описание SUOSO\_STRx в ini файле","Еррор",MB\_OK|MB\_APPLMODAL);  }  ~CONFIG()  {  WritePrivateProfileString("DEFAULT","LastOpenFILE",LastOpenFile,FileIni);  WritePrivateInt("POSITION","LEFT",left);  WritePrivateInt("POSITION","TOP",top);  WritePrivateInt("INTERFACE","ModeModel",ModeModel);  WritePrivateFloat("INTERFACE","TimeTest",TimeTest);  WritePrivateFloat("INTERFACE","MaxTimeGrShow",MaxTimeGrShow);  }  };  #endif |

Листинг 3 – Модуль отправки данных

|  |
| --- |
| #ifndef MY\_NET\_CLASS\_HEADER  #define MY\_NET\_CLASS\_HEADER  #define EXIT\_CODE\_THREAD 0x5050  #define ANSWER\_IDENT "OK\_IN\_NET"  #define my\_min(a,b) (((a)>(b))? (b):(a))  #define MY\_NEW(Where,What) { Where=new What; }  #define MY\_DELETE(What) { delete What; What=NULL; }  #define MY\_DELETE\_ARR(What) { delete []What; What=NULL; }  #include "high\_Timer.h"  #include <WinSock.h>  #include <Windows.h>  #include <wsipx.h>  /// только прием  #define NET\_OPT\_RECEIVE 0x0001  /// только передача  #define NET\_OPT\_SEND 0x0002  /// прием и передача  #define NET\_OPT\_BOTH (NET\_OPT\_SEND|NET\_OPT\_RECEIVE)  /// использование нестандартного протокола обмена (вначале идет счетчик пакетов)  #define NET\_PAR\_USE\_NON\_STANDART\_PROTOCOL 0x0004  /// не вызывается sleep(1) для потока RECEIVE  #define NET\_SLEEP\_RECV0 0x1000  /// не вызывается sleep(1) для потока SEND  #define NET\_SLEEP\_SEND0 0x2000  /// поставить поток RECEIVE на отдельное ядро  #define NET\_AFFINITY 0x4000  /// socket открывается в блокирующем режиме  #define NET\_BLOCKMODE 0x8000  /// для соединения по TCP выставляет CONNECT, a также используется при NET\_PAR\_USE\_NON\_STANDART\_PROTOCOL для IPX  #define NET\_OPT\_SERVER 0x0008  /// для соединения по TCP выставляет LISTEN, a также используется при NET\_PAR\_USE\_NON\_STANDART\_PROTOCOL для IPX  #define NET\_OPT\_CLIENT 0x0010  /// используется соединение по протоколу IPX  #define NET\_PAR\_USE\_IPX 0x0040  /// используется соединение по протоколу UDP  #define NET\_PAR\_USE\_UDP 0x0080  /// используется соединение по протоколу TCP  #define NET\_PAR\_USE\_TCP 0x0100  /// требуется детальное описание протоколов установленных в ОС, запись осуществляется в LOG файл (если он создается)  #define NET\_PAR\_NEED\_DESCRIPT\_PROTOCOL 0x0200  /// ТОЛЬКО ДЛЯ UDP!!! просмотр сообщения и всли UserFuncReceiveCheck вернул true, то чтение из буфера  #define NET\_PAR\_MSG\_PEEK 0x0400  // пакет icmp - пинг  #define NET\_PING 0x0800  #define NET\_ERROR 0x01  #define NET\_IDENT 0x02  #define NET\_DATA 0x04  #define NET\_END 0x08  /// максимальный размер пакета данных для приема или передачи  #define NET\_PACKET\_SIZE 26000  typedef bool (\*UserFuncReceive)(sockaddr\_in\*,void \*Buffer,int len);  typedef bool (\*UserFuncReceiveCheck)(sockaddr\_in\*,void \*Buffer,int len);  typedef struct  {  DWORD initOK :1; // [autoset] инициализация прошла успешно (internal)  DWORD Disconnected :1; // [autoset] в процессе работы был разрыв связи (internal)  }netFlags;  /\*\*  базовый класс для работы с сетевыми подключениями, в чистом виде не используется  содержит базовые переменные необходимые для IPX, UDP, TCP соединений  \*/  class COMMON\_NET  {  /// заполнение \*MyIp и cMyIp  void GetMyIP();  void Descript();  public:  /// собственное имя компьютера  char HostName[100];  /// список собственных адресов по которым идентифицируется этот компьютер  DWORD \*MyIp;  /// кол-во собственных адресов  int cMyIp;  /// пользовательская функция вызываемая при приеме данных, если из нее будет возврат 'true', то обработк приема дальше не выполняется  UserFuncReceive UserFn;  UserFuncReceiveCheck ChkFunction;  SOCKET Sock;  char Ip4Addr[20];  WORD Port;  WORD FlagOpt;  /// возможные значения NSPROTO\_IPX, IPPROTO\_UDP, IPPROTO\_TCP  DWORD protocol\_family;  /// возможные значения (protocol\_family==NSPROTO\_IPX)? AF\_IPX:AF\_INET;  DWORD protocol;  /// возможные значения (protocol\_family==IPPROTO\_TCP)? SOCK\_STREAM:SOCK\_DGRAM  DWORD protocol\_type;  sockaddr\_in AddrFrom;  /// автоматически выставляемые флаги  netFlags iFlag;  // ====================================================  /// конструктор базового класса  /// @param Adr - текстовый адрес в формате Ip4  /// @param port - порт через который будет идти обмен данными  /// @param Opt - набор флагов NET\_xxxxx из диапазона 0xffff  /// @param protocol\_family - NSPROTO\_IPX, IPPROTO\_UDP или IPPROTO\_TCP  COMMON\_NET(char \*Adr,WORD port,WORD Opt,DWORD protocol\_family);  ~COMMON\_NET() { MY\_DELETE(MyIp); CloseSock(&Sock); }  bool ErrorMessage(SOCKET \*s,char \*fl=\_\_FILE\_\_,int str=\_\_LINE\_\_);  void CloseSock(SOCKET \*Socket);  virtual bool InitSocket(SOCKET \*Socket);  DWORD PrepareReceive(DWORD CodeRet,void \*BufTo,void \*BufFrom,int AddrtoLen);  void SetUserFunction(UserFuncReceive Usr) { UserFn=Usr; }  void SetChkFunction(UserFuncReceiveCheck Chk) { ChkFunction=Chk; }  };  // =================================================================================  // =================================================================================  /\*\*  класс для работы с сетевыми подключениями по протоколу IPX  \*/  class IPX\_NET:public COMMON\_NET  {  int counter;  char MacAdr[6];  public:  IPX\_NET(char \*Adr,WORD port,WORD Opt,DWORD protocol\_family):  COMMON\_NET(Adr,port,Opt,protocol\_family) { InitSocket(&Sock); counter=0; }  ~IPX\_NET(){;}  virtual bool InitSocket(SOCKET \*Socket);  bool Send(void \*Buf,int len);  DWORD Receive(void \*Buf,int len);  };  // =================================================================================  // =================================================================================  /\*\*  класс для работы с сетевыми подключениями по протоколу TCP  \*/  class TCP\_NET:public COMMON\_NET  {  SOCKET SockTCP;  public:  TCP\_NET(char \*Adr,WORD port,WORD Opt,DWORD protocol\_family):  COMMON\_NET(Adr,port,Opt,protocol\_family) { InitSocket(&Sock); }  ~TCP\_NET(){ if(SockTCP) CloseSock(&SockTCP);}  virtual bool InitSocket(SOCKET \*Socket);  bool Send(void \*Buf,int len);  DWORD Receive(void \*Buf,int len);  void Thread();  bool Connect();  bool Listen();  };  // =================================================================================  // =================================================================================  /\*\*  класс для работы с сетевыми подключениями по протоколу UDP  \*/  class UDP\_NET:public COMMON\_NET  {  public:  UDP\_NET(char \*Adr,WORD port,WORD Opt,DWORD protocol\_family):  COMMON\_NET(Adr,port,Opt,protocol\_family) { InitSocket(&Sock); }  ~UDP\_NET() {;}  virtual bool InitSocket(SOCKET \*Socket);  bool Send(void \*Buf,int len,char \*Ip4To=NULL);  DWORD Receive(void \*Buf,int len);  };  // =================================================================================  // =================================================================================  struct ip // Структура заголовка IP  {  BYTE ip\_verlen; // Version and header length  BYTE ip\_tos; // Type of service  WORD ip\_len; // Total packet length  UINT ip\_id; // Datagram identification  WORD ip\_fragoff; // Fragment offset  BYTE ip\_ttl; // Time to live  BYTE ip\_proto; // Protocol  UINT ip\_chksum; // Checksum  IN\_ADDR ip\_src\_addr; // Source address  IN\_ADDR ip\_dst\_addr; // Destination address  BYTE ip\_data[1]; // Variable length data area  };  struct icmp // Структура заголовка ICMP  {  BYTE icmp\_type; // Type of message  BYTE icmp\_code; // Type "sub code" (zero for echos)  WORD icmp\_cksum; // 1's complement checksum  HINSTANCE icmp\_id; // Unique ID (the instance handle)  WORD icmp\_seq; // Tracks multiple pings  BYTE icmp\_data[1]; // The start of optional data  };  /\*\*  класс для работы с icmp - аналог команды PING  \*/  class ICMP\_NET:public COMMON\_NET  {  WORD InternetChksum(LPWORD lpwIcmpData, WORD wDataLength);  public:  ICMP\_NET(char \*Adr,WORD port,WORD Opt,DWORD protocol\_family):  COMMON\_NET(Adr,port,Opt,protocol\_family) { InitSocket(&Sock); }  ~ICMP\_NET(){;}  virtual bool InitSocket(SOCKET \*Socket);  bool DoPing(char \*Text,char \*Ip4To);  };  // =================================================================================  // =================================================================================  DWORD WINAPI Thread\_Net\_Work\_R(LPVOID param);  DWORD WINAPI Thread\_Net\_Work\_S(LPVOID param);  // =================================================================================  // =================================================================================  /\*\*  класс для работы с сетевыми подключениями по протоколу UDP  \*/  class NETCONN  {  WORD FlagOpt;  bool Done\_Work\_S; // флаг прекращения работы потока по передаче данных  bool Done\_Work\_R; // флаг прекращения работы потока по приему данных  HANDLE ThreadNet\_S;  HANDLE ThreadNet\_R;  // ---------------------------------------  public:  IPX\_NET \*ipx\_net;  TCP\_NET \*tcp\_net;  UDP\_NET \*udp\_net;  ICMP\_NET \*icmp\_net;  /// тип принятых данных  DWORD TypeReceive:4;  /// необходимо выполнить передачу  DWORD NeedSend:1;  /// был выполнен прием данных  DWORD Receiving:1;  /// буфер содержащий принятые данные  char \*Buff\_R;  /// длина принятых данных  int cBuff\_R;  /// буфер содержащий данные которые необходимо передать  char \*Buff\_S;  /// длина передаваемых данных  int cBuff\_S;  // ========================================  /// конструктор класса для работы с сетевыми подключениями  /// @param Adr - текстовый адрес в формате Ip4  /// @param port - порт через который будет идти обмен данными  /// @param Opt - набор флагов NET\_xxxxx из диапазона 0xffff  NETCONN(char \*Adr,WORD port,WORD Opt)  {  FlagOpt=Opt;  ThreadNet\_S=NULL;  ThreadNet\_R=NULL;  ipx\_net=NULL;  udp\_net=NULL;  tcp\_net=NULL;  icmp\_net=NULL;  MY\_NEW(Buff\_S,char[NET\_PACKET\_SIZE]);  MY\_NEW(Buff\_R,char[NET\_PACKET\_SIZE]);  cBuff\_S=cBuff\_R=0;  Receiving=false;  NeedSend=false;  DWORD Prot\_fam=(Opt&NET\_PAR\_USE\_IPX)? NSPROTO\_IPX:0;  if(!Prot\_fam) Prot\_fam=(Opt&NET\_PAR\_USE\_UDP)? IPPROTO\_UDP:0;  if(!Prot\_fam) Prot\_fam=(Opt&NET\_PAR\_USE\_TCP)? IPPROTO\_TCP:0;  if(!Prot\_fam) Prot\_fam=(Opt&NET\_PING)? IPPROTO\_ICMP:0;  if(Prot\_fam==NSPROTO\_IPX) MY\_NEW(ipx\_net,IPX\_NET(Adr,port,Opt,Prot\_fam));  if(Prot\_fam==IPPROTO\_TCP) MY\_NEW(tcp\_net,TCP\_NET(Adr,port,Opt,Prot\_fam));  if(Prot\_fam==IPPROTO\_UDP) MY\_NEW(udp\_net,UDP\_NET(Adr,port,Opt,Prot\_fam));  if(Prot\_fam==IPPROTO\_ICMP) MY\_NEW(icmp\_net,ICMP\_NET(Adr,port,Opt,Prot\_fam));  Init(Opt);  }  // ========================================  ~NETCONN()  {  DeInit();  delete ipx\_net;  delete tcp\_net;  delete udp\_net;  MY\_DELETE(Buff\_S);  MY\_DELETE(Buff\_R);  }  // ========================================  DWORD GetIp4ByHostName(char \*Host);  // ========================================  char\* GetIp4Str(DWORD Addr)  {  static char Ip[4\*3+4];  in\_addr \*str\_in=(in\_addr \*)&Addr;  strcpy(Ip,inet\_ntoa(\*str\_in));  return Ip;  }  // ========================================  void SetChkFunction(UserFuncReceiveCheck Chk)  {  if(udp\_net) udp\_net->SetChkFunction(Chk);  }  // ========================================  void SetUserFunction(UserFuncReceive Usr)  {  if(ipx\_net) ipx\_net->SetUserFunction(Usr);  if(udp\_net) udp\_net->SetUserFunction(Usr);  if(tcp\_net) tcp\_net->SetUserFunction(Usr);  }  // ========================================  DWORD SRV\_Thread\_Net\_Work\_R();  DWORD SRV\_Thread\_Net\_Work\_S();  // ========================================  int GetDataReceiveSize() { return cBuff\_R; }  // ========================================  void Init(WORD FlOpt)  {  if(FlOpt & NET\_PING) return;  DWORD ThreadId;  if(FlOpt & NET\_OPT\_SEND)  ThreadNet\_S=CreateThread(NULL,0,::Thread\_Net\_Work\_S,(LPVOID)this,0,&ThreadId); // ethernet  //SetThreadPriority(ThreadNet\_S,THREAD\_PRIORITY\_BELOW\_NORMAL);  if(FlOpt & NET\_OPT\_RECEIVE)  {  ThreadNet\_R=CreateThread(NULL,0,::Thread\_Net\_Work\_R,(LPVOID)this,0,&ThreadId); // ethernet  //SetThreadPriority(ThreadNet\_R,THREAD\_PRIORITY\_BELOW\_NORMAL);  SYSTEM\_INFO Info;  GetSystemInfo(&Info);  if(FlagOpt & NET\_AFFINITY)  {  DWORD Ret=SetThreadAffinityMask(ThreadNet\_R,1<<(Info.dwNumberOfProcessors-1));  if(my\_LOG)  my\_LOG->WriteLogCommon(ERR\_INFO1,"sd","Поток receive назначен на ядро",Info.dwNumberOfProcessors-1);  }  }  }  // ========================================  void DeInit()  {  SetUserFunction(NULL);  SetChkFunction(NULL);  if(ThreadNet\_S)  {  Done\_Work\_S=true;  DWORD ExitCode=0;  HighTimer ht;  while (ExitCode!=EXIT\_CODE\_THREAD)  {  if(ht.GetSecond()>1) break;  GetExitCodeThread(ThreadNet\_S,&ExitCode);  Sleep(1);  }  CloseHandle(ThreadNet\_S);  }  if(ThreadNet\_R)  {  Done\_Work\_R=true;  DWORD ExitCode=0;  HighTimer ht;  while (ExitCode!=EXIT\_CODE\_THREAD)  {  if(ht.GetSecond()>1) break;  GetExitCodeThread(ThreadNet\_R,&ExitCode);  Sleep(1);  }  CloseHandle(ThreadNet\_R);  }  }  // ========================================  void SetDataToSend(void \*dt,int size)  {  memcpy(Buff\_S,dt,size);  cBuff\_S=size;  NeedSend=true;  }  // ========================================  int GetDataFromReceive(void \*dt,int size)  {  if(!Receiving) return 0;  Receiving=false;  if(TypeReceive!=NET\_DATA) return 0;  memcpy(dt,Buff\_R,my\_min(cBuff\_R,size));  return cBuff\_R;  }  }; |

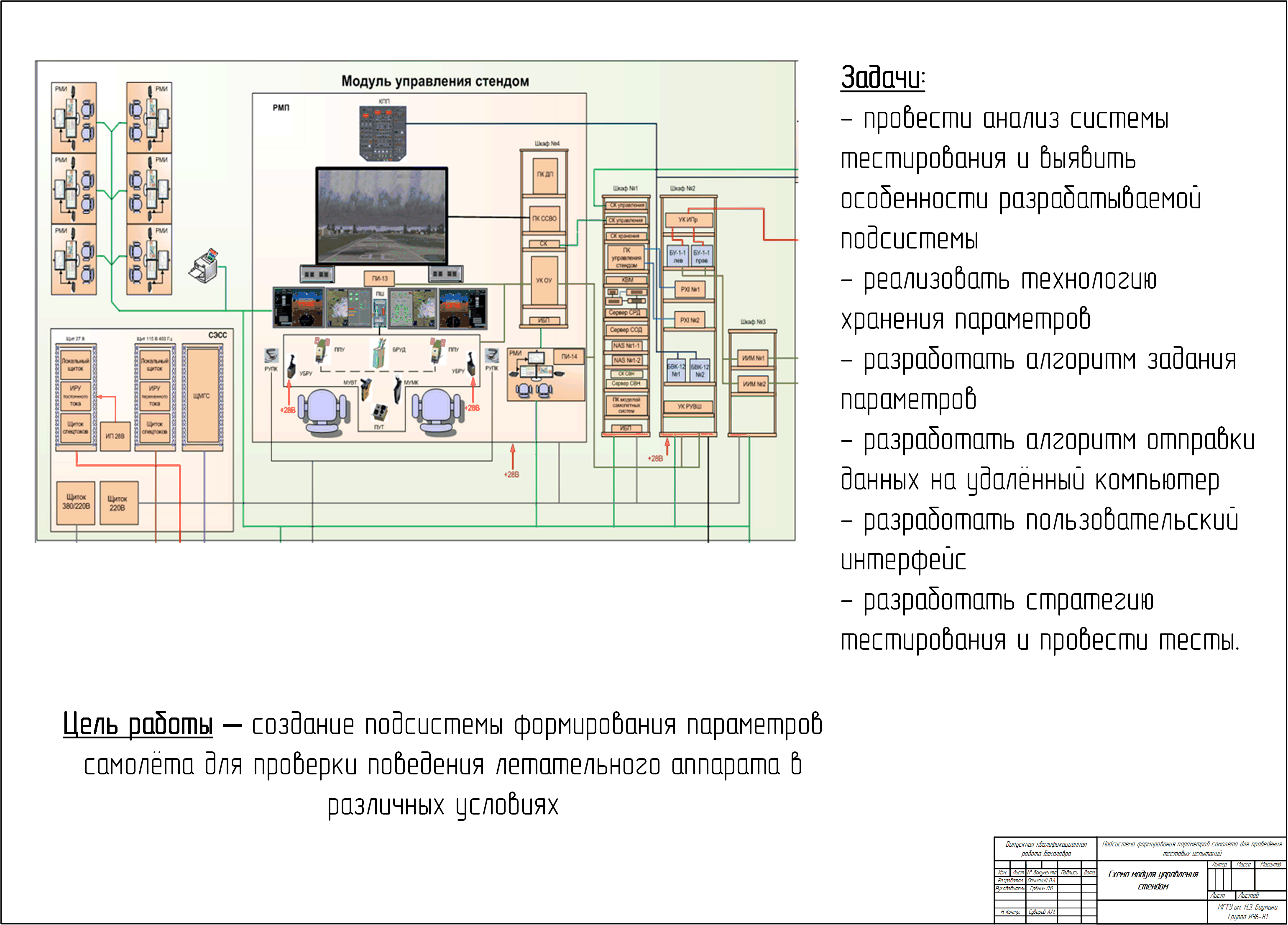
Листинг 4 – Модуль высокочастотного таймера

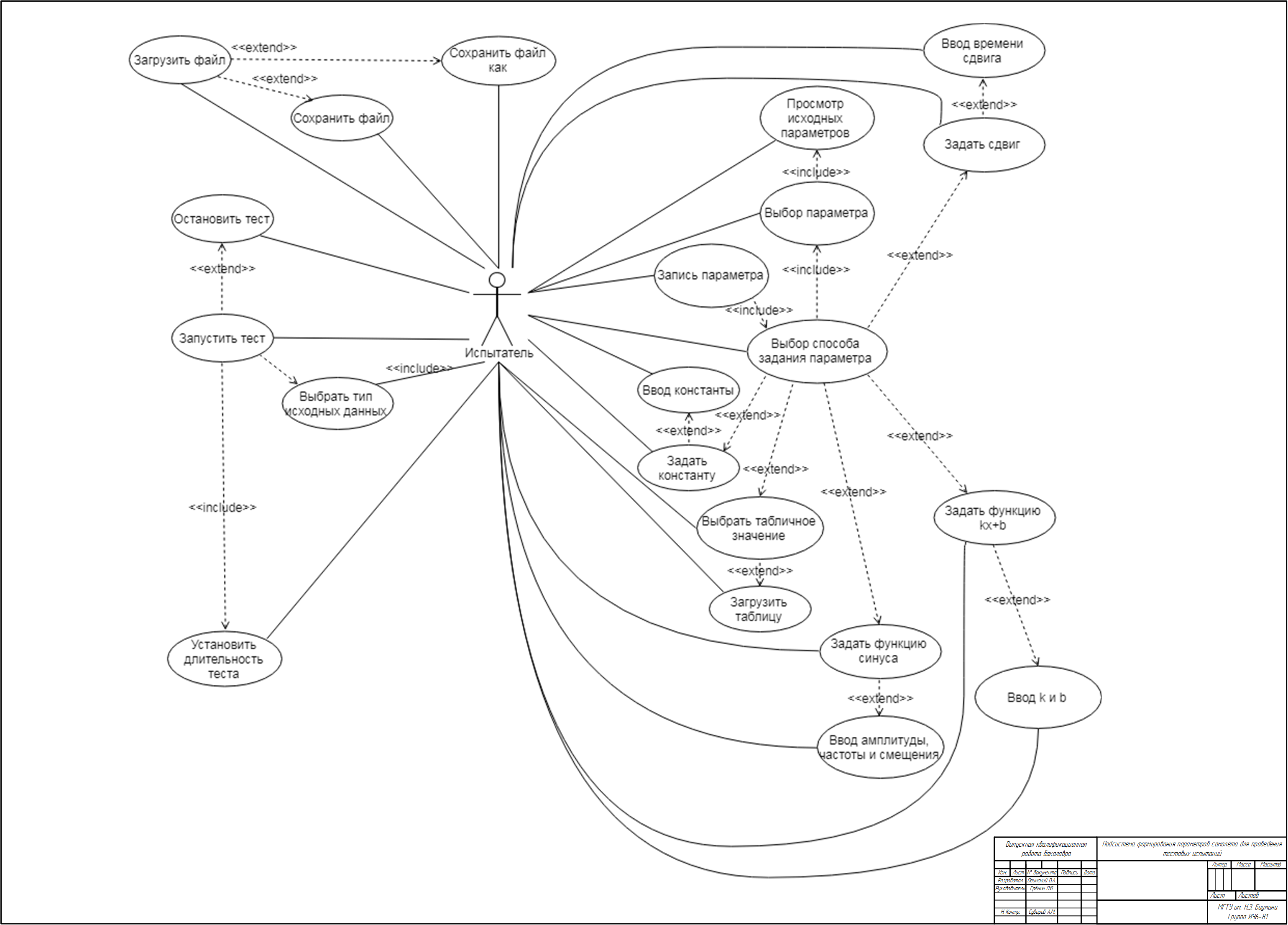
|  |
| --- |
| #ifndef \_\_HIGH\_TIMER\_HEADER  #define \_\_HIGH\_TIMER\_HEADER  #include <Windows.h>  #include <TIME.H>  class HighTimer  {  private:  LARGE\_INTEGER Freq;  LARGE\_INTEGER PrevTime;  LARGE\_INTEGER end;  clock\_t PrevT;  clock\_t en;  BOOL FlUse;  public:  // -------------------------------------------------  HighTimer()  {  FlUse=QueryPerformanceFrequency(&Freq);  if(FlUse) QueryPerformanceCounter(&PrevTime);  else PrevT=clock();  }  // -------------------------------------------------  ~HighTimer() {;}  // -------------------------------------------------  void Restart()  {  if(FlUse) QueryPerformanceCounter(&PrevTime);  else PrevT=clock();  }  // -------------------------------------------------  void Set(double Tm)  {  QueryPerformanceCounter(&PrevTime);  PrevTime.QuadPart-=Freq.QuadPart\*Tm;  }  // -------------------------------------------------  double GetSecond(bool Set=false)  {  double Tm;  if(FlUse)  {  QueryPerformanceCounter(&end);  Tm=((double)(end.QuadPart-PrevTime.QuadPart))/Freq.QuadPart; // время в секундах от предыдущего вызова  }  else  {  en=clock();  Tm=(double)(en-PrevT)/CLK\_TCK;  }  if(Set) Restart();  return Tm;  }  };  #endif |

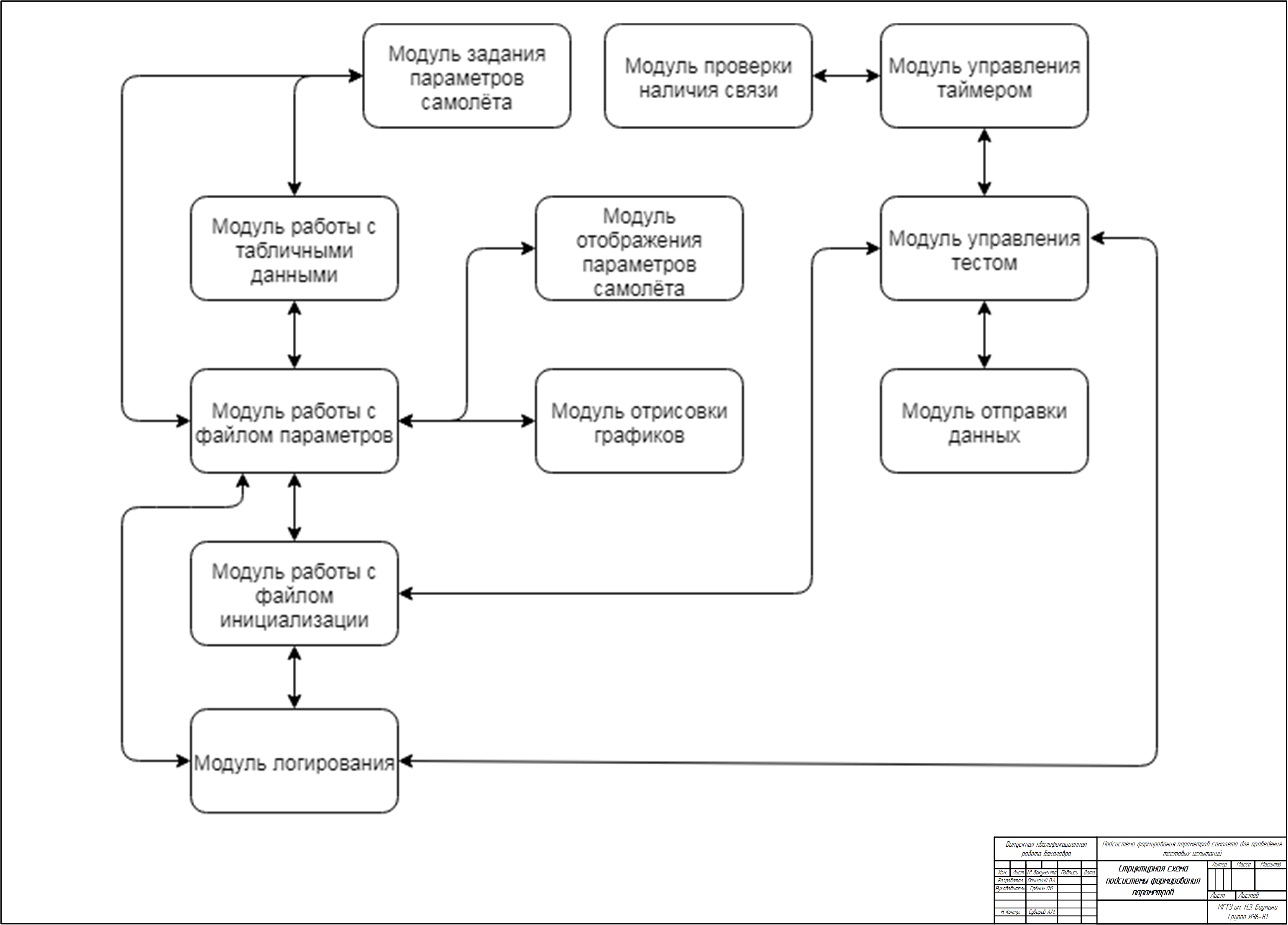
# ПРИЛОЖЕНИЕ Г

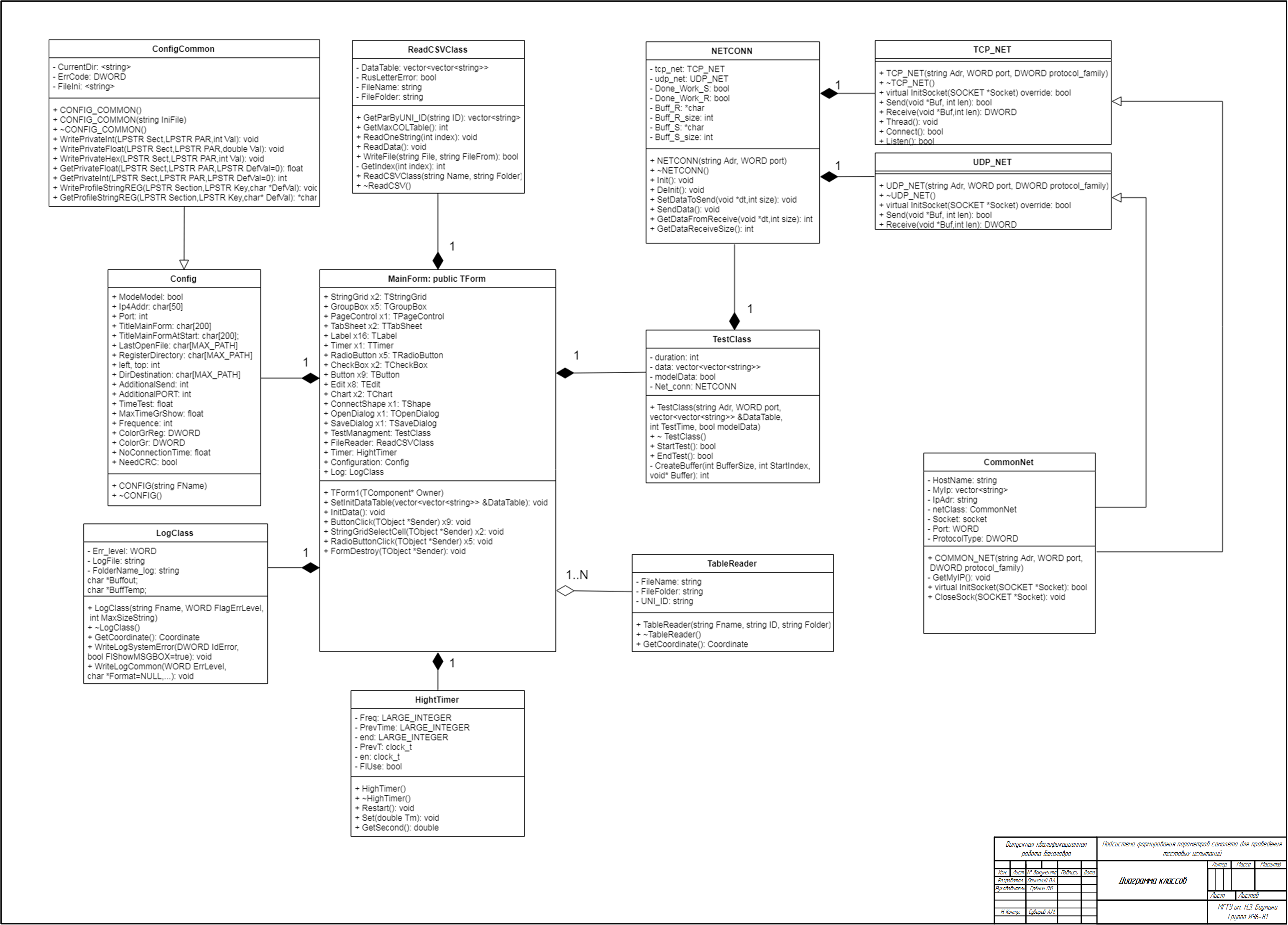
Копии листов графической части

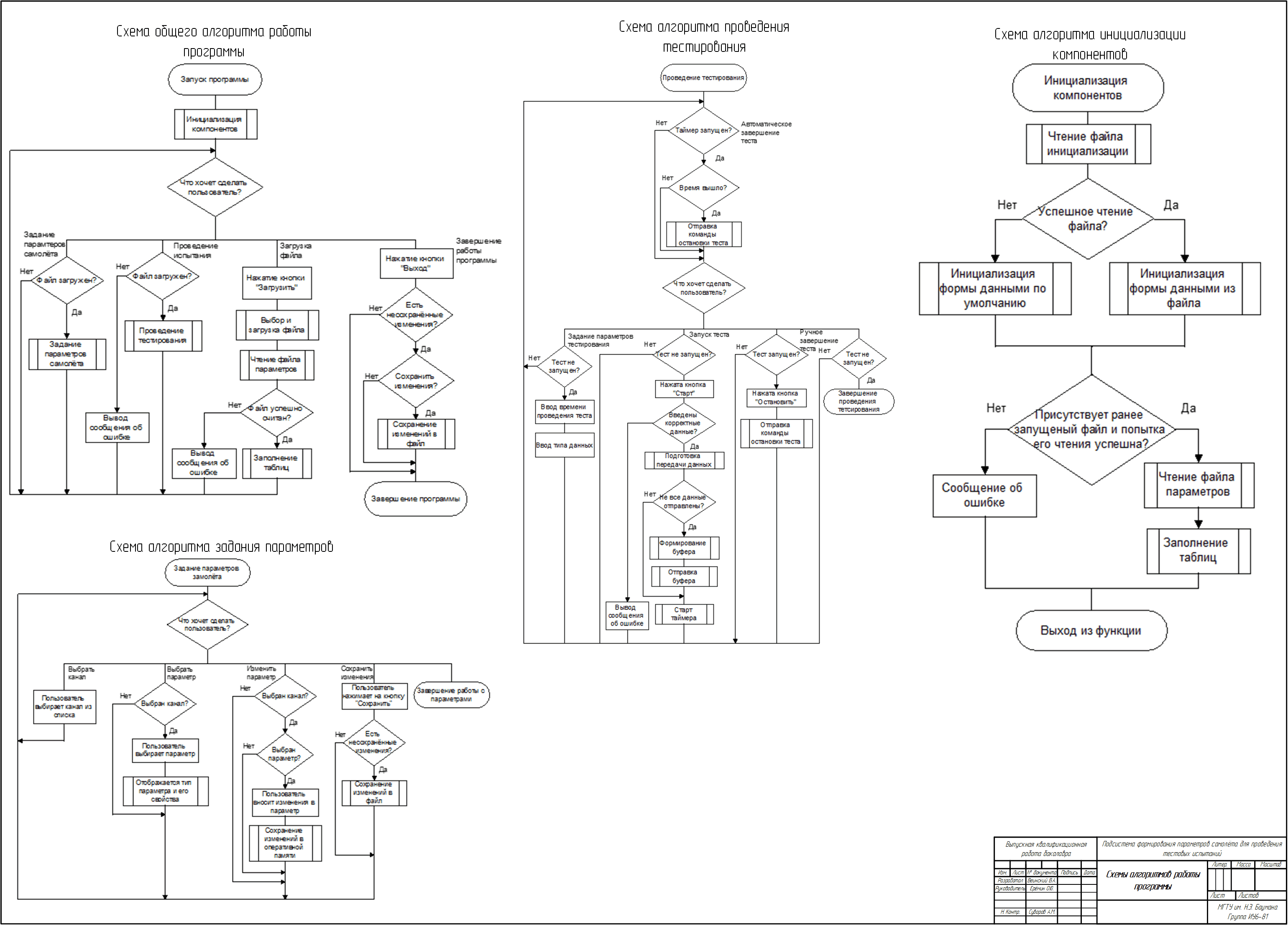
Листов 6

****

****

****

****

****