РЕФЕРАТ

Выпускная квалификационная работа 5 глав, 98 стр., 19 рисунков, 9 таблиц, 13 источников, 4 приложений.

ПОВЕРКА, КАЛИБРОВКА, ИЗМЕРИТЕЛЬНЫЙ КАНАЛ, ГЕНЕРАЦИЯ ОТЧЁТА.

Объектом исследования в рамках данной дипломной работы является система измерения стенда статических испытаний авиационного реактивного двигателя.

Цель работы: разработка программного модуля поверки\калибровки измерительных каналов с входным сигналом на модули MR-212 в составе системы измерения стенда статических испытаний. Оптимизация технологического процесса поверки\калибровки путём автоматизации таких процессов, как: балансировка нуля измерительных каналов, фиксация их значений, генерация отчётов о поверке\калибровке.

В результате дипломной работы были изучены принцип работы модулей для работы с тензометрическими датчиками MR-212, разработаны пргограммные модули поверки\калибровки и управления виртуальными тегами, предложина схема подключения измерительных каналов к поверочному оборудованию.

Практическая значимость работы заключается в получении программного модуля системы измерения, использование которого позволит минимизировать время, затрачиваемое на поверки\калибровку измерительных каналов с входным сигналом на модули MR-212 в составе системы измерения стенда статических испытаний.

Разработанные программные модули могут быть применены в дальнейших модификациях измерительной системы, а также в разработках новых plug-in’.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИК – измерительный канал

Испытательный стенд – это лабораторное оборудование, которое предназначено для специальных, контрольных, приёмочных испытаний разнообразных объектов.

НДС – напряженно деформированное состояние

ПВМ – персональная электронно-вычислительная машина

ПКМ – полимерный композитный материал

ПО – программное обеспечение

Тег – это программный объект, у которого есть ряд свойств, таких как имя, срока описания, тег обладает буфером для хранения данных.

ВВЕДЕНИЕ

Изучение сложных физических явлений, происходящих на различных участках тракта двигателя, и их взаимозависимости в ряде случаев требует длительного времени. [1] Вместе с тем в расчётных условиях не всегда представляется возможным правильно назначить действительные условия нагружения деталей – величины нагрузок, закономерность их проявления. Не всегда расчётная модель учитывает особенность форм детали. Не всегда учитываются все механизмы нагружения деталей в условии эксплуатации.

Поэтому наряду с предварительными инженерными расчётами приходится экспериментально проверять работу отдельных элементов, узлов, агрегатов и двигателей в целом. Для экспериментального определения напряженно деформированного состояния деталей и узлов авиационных двигателей применяют различные методы и средства, в которых используются различные принципы измерений. Среди них – методы, основанные на масштабном преобразовании деформаций поверхности детали с помощью тензометров и тензометрических преобразователей – тензометрирование. [2]

Целью моделирования является определение фактических внешних нагрузок, закономерность их проявлений, измерение деформаций и напряжений, возникающих при работе, и проверка соответствия напряженно-деформированного состояния расчётным данным.

Для моделирования нагрузок, соответствующих полетным и посадочным, на авиационный двигатель и крупногабаритные узлы из полимерного композитного материала с целью определения соответствия техническим требованиям по прочности, жесткости, устойчивости и несущей способности силовых элементов корпуса двигателя, а также для определения циклической долговечности узлов и агрегатов используется стенд статических испытаний. В измерительной системе стенда задействованы тензометрические измерительные каналы с входным сигналом на модули MR-212 в количестве 1024 штук.

Поверка и калибровка такого объёма ИК требует значительных временных ресурсов, также существует вероятность ошибки при формировании протоколов калибровки\поверки.

Актуальность темы заключается в необходимости минимизировать время, затрачиваемое на поверку\калибровку измерительных каналов системы измерения стенда статических испытаний, а также человеческий фактор.

Объектом исследования является система измерения стенда статических испытаний авиационного реактивного двигателя.

Предметом исследования является программный модуль для калибровки измерительных каналов с входным сигналом на модули MR-212.

Для достижения поставленной цели требуется решение следующих **задач**:

- выделить основные требования к программному модулю;
- проанализировать методику поверки измерительных каналов;
- спроектировать программный модуль поверки\калибровки;
- спроектировать программный модуль для тестирования;
- реализовать программный модуль поверки\калибровки;
- реализовать программный модуль для тестирования;
- протестировать программный модуль поверки\калибровки.

1 Исследовательский раздел

В данном разделе рассматриваются характеристика и анализ объекта исследования. Анализируется проблемма и предлагается метод её решения.

1.1 Характеристика и анализ объекта исследования

Стенд статических испытаний авиационного двигателя и крупногабаритных узлов из ПКМ предназначен для проведения испытаний корпусов и подвесок авиационного двигателя на соответствие техническим требованиям по прочности, жесткости, устойчивости и несущей способности силовых элементов корпуса двигателя при нагрузках, соответствующих полетным и посадочным, а также циклической долговечности корпусов двигателя и подвесок; для проведения статических испытаний мотогондолы.

Стенд состоит из трех основных систем:

– системы крепления;

– системы нагружения и управления нагружением;

– системы измерения.

Структурная схема потоков информации в системе измерений представлена на рисунке 1.

В состав системы измерения входит шестнадцать универсальных многоканальных измерительно-вычислительных комплекса MIC-236, в которые установлены по шестнадцать модулей для работы с тензометрическими датчиками MR-212, имеющих четыре канала. Структурная схема системы измерения представлена на рисунке 2.

Напряжения тензодатчиков по 128 групповым кабелям поступают на входы 256 четырехканальных измерительных модулей типа MR-212, входящих в состав 16 многоканальных измерительных комплексов MIC-236. Часть кабельных каналов и слотов комплексов MIC-236 используется модулями MR-114, MR-114C1, MR-405 и MR-406, которые используются в технологических целях специалистами изготовителями системы измерений при её настройке [3].

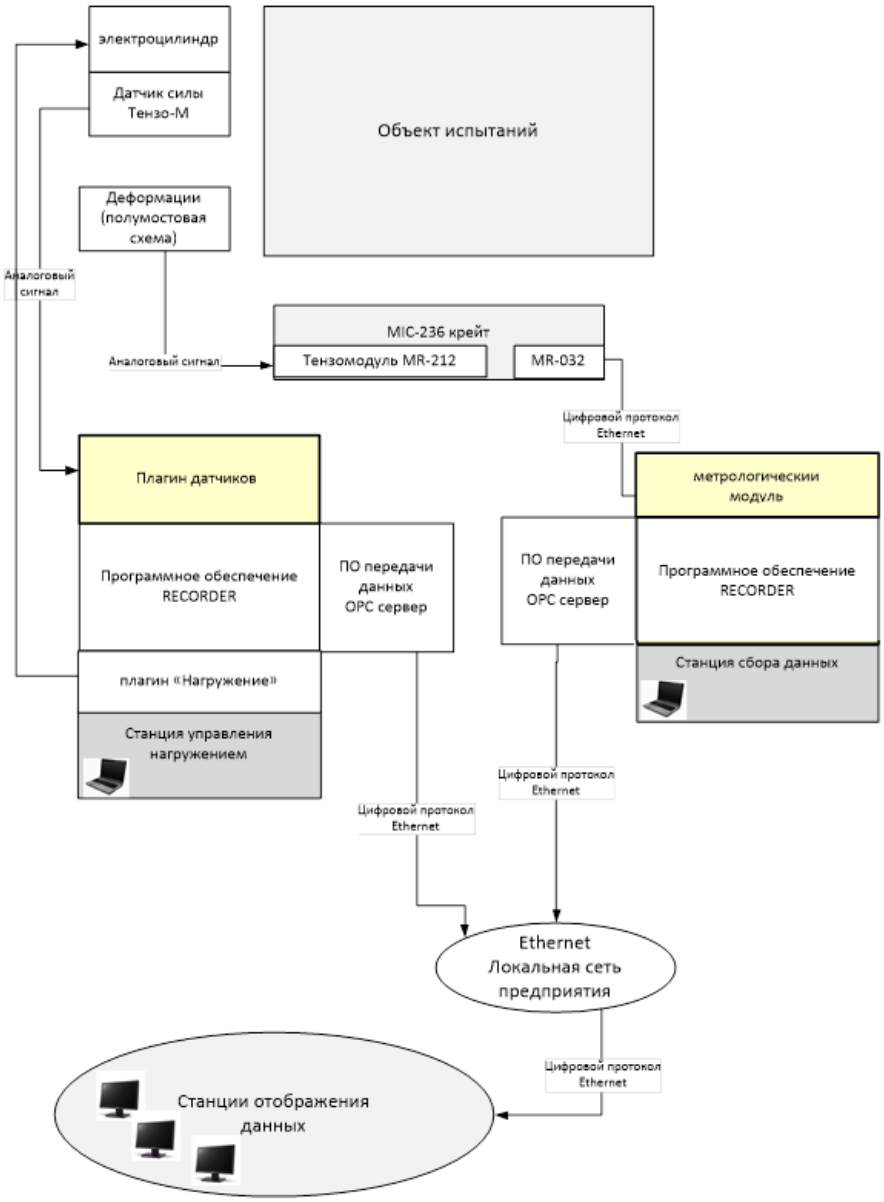

Рисунок 1 - Поток информации в системе измерений

Система измерений стенда статических испытаний обеспечивает регистрацию и обработку данных измерений параметров объектов испытаний, которые поступают с:

– 1024 (MR-212) тензодатчиков, измеряющих напряжения в деталях конструкции объекта испытаний с использованием полумостовых схем включения тензорезисторов;

– 20 датчиков контактного типа, регистрирующих механические перемещения элементов объекта испытаний;

– 23 лазерных датчиков, регистрирующих механические перемещения и деформации конструкции объекта испытаний;

– 25 тензорезисторных датчиков силы, регистрирующих прикладываемые усилия к объекту испытаний;

– 6 видеокамер, фиксирующих изображение объекта испытаний и окружающей обстановки при проведении испытаний.

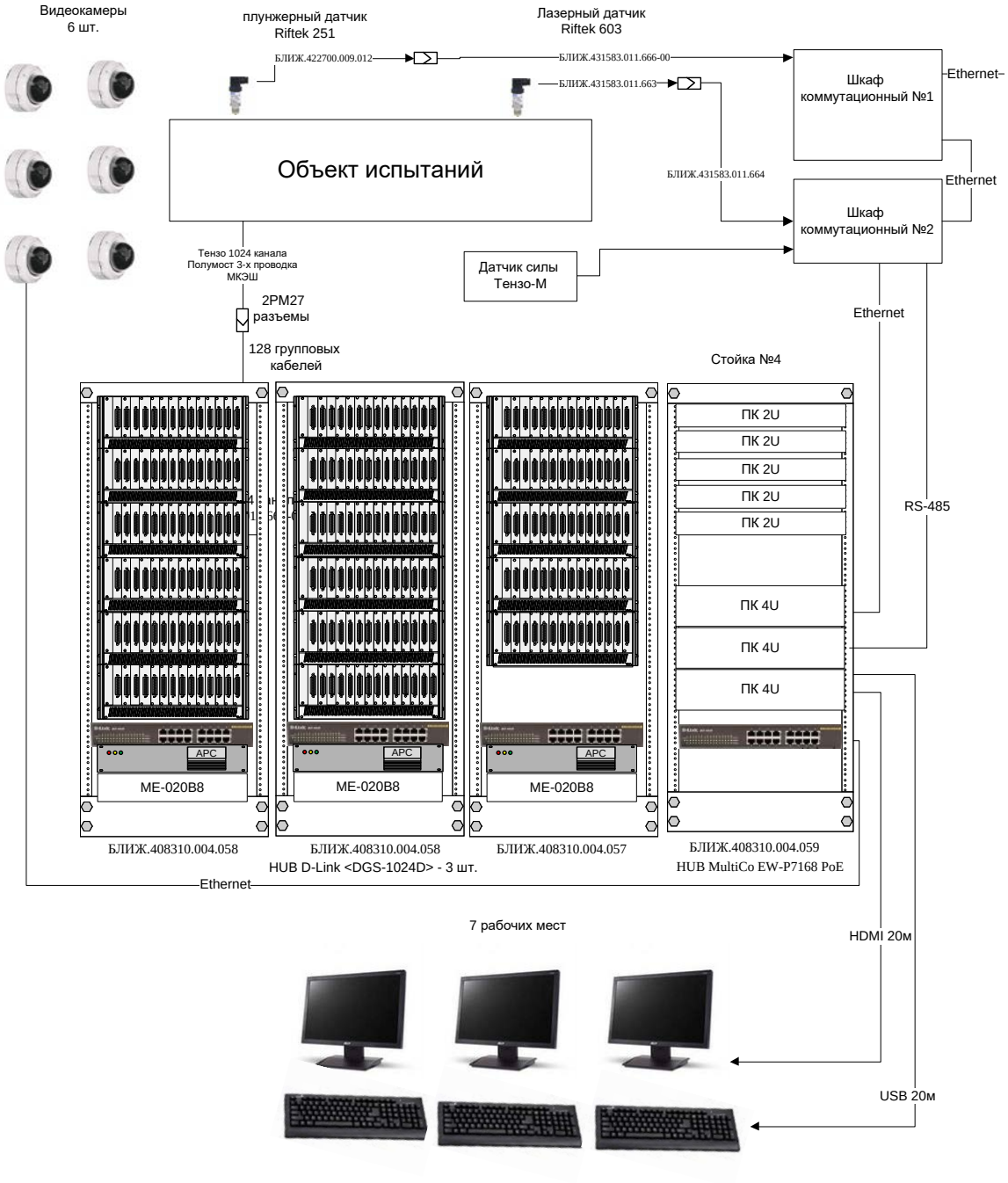

Рисунок 2 - Структурная схема системы измерения

1.1.1 Описание универсального многоканального магистрально-модульного измерительного комплекса MIC-236

Универсальный многоканальный комплекс серии RXI для регистрации и анализа измерительных данных. Базовый прибор для построения систем стендовых испытаний, многоканальных измерительных комплексов.

Комплекс может работать как с быстроменяющимися (вибрация, акустика), так и с медленноменяющимися (температура, давление, нагрузка, усилие, деформация) параметрами.

Технические характеристики измерительного комплекса MIC-236 [4] приведены в таблице 1.

Таблица 1 - Технические характеристики измерительного комплекса MIC-236

Наименование характеристики	**Значение параметра**
Количество слотов для установки измерительных и нормирующих модулей	16
Тип крейт-контроллера	MR-032
Интерфейса связи с ПЭВМ (Разъем RJ-45)	Ethernet
Интерфейс связи с устройством синхронизации (Разъем DB-9M)	IRIG-B
Время непрерывной работы, час	200
Наработка до отказа, час, не менее	10 000
Средний срок службы, лет	7
Габариты Ш×Г×В, мм	485×365×180
Масса (с полным комплектом модулей), кг	10
Частота питающей сети переменного тока, Гц	50±1
Напряжение питающей сети переменного тока, В	220 ± 22
Потребляемая мощность, ВА, не более	200

1.1.2 Описание модуля MR-212

Модули MR-212 предназначены для работы с мостовыми, полумостовыми тензодатчиками и одиночными тензорезисторами, включенными по схеме четвертьмоста, при проведении статических и динамических измерений. Для работы с модулем MR-212(рисунок 3) могут быть использованы тензодатчики сопротивлением от 100 до 1000 ом.

Основные области применения модулей MR-212:

– измерение механических нагрузок;

– измерение усилий (тяга, вес);

– высокоточные измерения линейных перемещений;

– измерение давлений при использовании тензометрических датчиков давления.

Рисунок 3 – Модуль MR212

Технические характеристики модуля MR-212 приведены в таблице 2[5].

Таблица 2 - Технические характеристики модуля MR-212

Наименование характеристики	**Значение параметра**
Количество независимых каналов	4
Диапазоны измерения	0…10мВ, -10…+10мВ, 0…20мВ, -20…+20мВ, 0…40мВ, -40…+40мВ, 0…80мВ, -80…+80Мв
Напряжение питания тензодатчиков (знакопеременный ток), В	2,5; 5
Частота сбора данных: - Режим “динамика” - Режим “статика”	 4800, 1200, 600, 300 50, 12.5, 6.25
Остаточное смещение нуля (после внутренней калибровки)	2 μV
Температурный дрейф смещения нуля	0,5 μV/°C
Временной дрейф смещения нуля	2,5 μV/1000ч
Основная приведенная погрешность измерения	0.05%
Дополнительная приведенная погрешность измерения	0.05%
Временной дрейф коэффициента передачи	10 ppm/1000ч
Интегральная нелинейность	15 ppm от полной шкалы
Полоса пропускания при неравномерности 0,1 дБ, Гц	0 … 1000
Подавление синфазной составляющей входного сигнала	100 дб
Межканальное прохождение дифференциальной составляющей	-100дб
Входной ток	60 nA
Разность входных токов	30 nA

Функциональная схема модуля MR-212 приведена на рисунке 4.

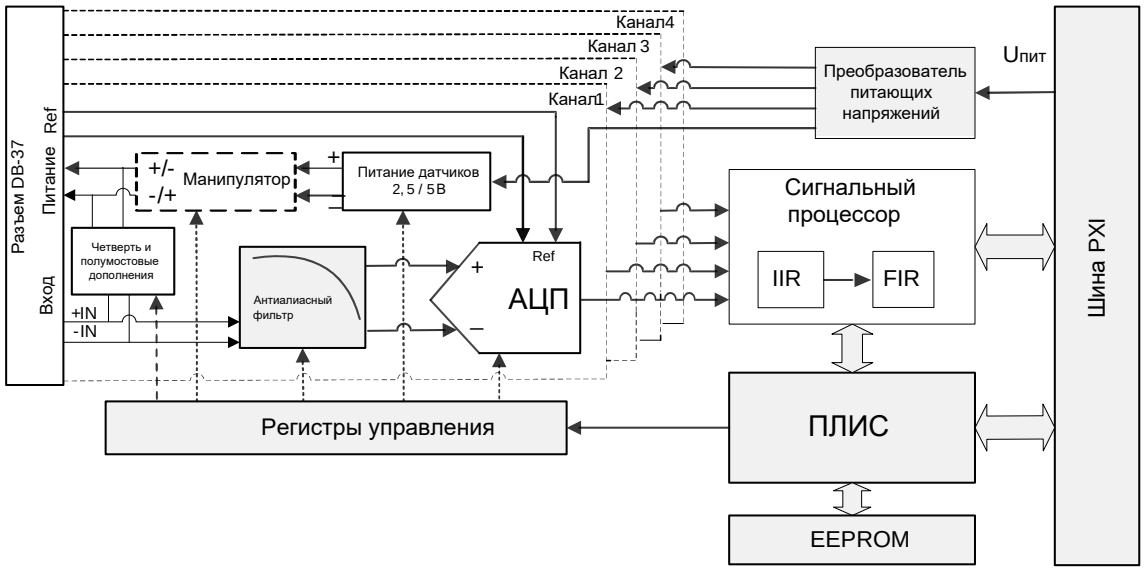

Рисунок 4 – Функциональная схема модуля MR212

Измерительные схемы включают в себя мостовые, полумостовые тензодатчики и одиночные тензорезисторы, включенные по схеме четвертьмоста, а также элементы дополнения до моста, содержащиеся в каждом канале. Шунт, подключаемый к одному из плеч измерительного моста предназначен для калибровки канала.

Модуль позволяет осуществлять измерения в режиме питания измерительной схемы постоянным током напряжением 2,5 или 5 В и в знакопеременном режиме, формируемом манипулятором. Знакопеременный режим применяется при измерениях стационарных и медленноменяющихся процессов.

С измерительной схемы измеряемое напряжение поступает на дифференциальный сигнальный вход АЦП (+In, -In), а опорное напряжение (+Ref, -Ref) поступает на дифференциальный вход опорного напряжения АЦП. На сигнальном входе АЦП стоят антиалиасные фильтры, улучшающие отношения сигнал/шум измерительных каналов.

С выхода АЦП сигналы поступают вход сигнального процессора (ЦСП), который одновременно запускает и синхронизирует работу всех преобразователей.

В ЦСП реализованы алгоритмы двух фильтров:

– Первый фильтр предназначен для выравнивания АЧХ модуля путем компенсации искажений, вносимых АЦП. Фильтр является IIR-фильтром второго порядка с загружаемыми коэффициентами;

– Второй фильтр предназначен для ограничения частотной полосы модуля с целью удаления из сигнала внеполосных шумов. Фильтр является FIR-фильтром 131 порядка с линейной ФЧХ. Полоса пропускания фильтра, определяемая его коэффициентами, (и количество коэффициентов) зависит от выбранного режима работы модуля, поэтому в комплекте с программой идут несколько заранее рассчитанных наборов коэффициентов для разных режимов работы модуля.

После цифровой фильтрации предусмотрена операция децимации данных, которая позволяет сократить объем передаваемых данных. Коэффициент децимации также зависит от режима работы модуля и может настраиваться управляющей программой.

ЦСП через PXI шину крейта связан со станцией сбора данных (управляющей ПЭВМ).

Управление работой модуля осуществляет ПЛИС, также связанная с через PXI шину с управляющей ПЭВМ. К ПЛИС подключена flash–память EEPROM, в которой сохраняются данные настройки и калибровки каналов модуля.

1.2 Анализ проблемы

Согласно существующей методики поверки, калибровка и поверка ИК с входным сигналом на модули MR-212 осуществляется по пяти точкам в прямом ходе и обратном. Суммарное количество замеров для одного канала равняется десяти, и занимает, в среднем, пять минут, без учёта занесения данных в протокол. Также на подключение ИК к измерительному оборудованию уходит до пяти минут.

Этапы методики поверки отражены в таблице 3 в качестве технологического процесса.

Таблица 3 – Технологический процесс поверки и калибровки

Номер и наименование операции	**Цель**	**Содержание**	**t, мин**
005 Подготовительная	Подготовить оборудование к поверке\калибровке	1. Включить измерительное оборудование 2. Собрать схему в соответствии с методикой поверки 3. Запустить ПО Recorder	20
010 Поверка\калибровка	Поверить\откалибровать ИК	1. Подключить ИК к собранной схеме 2. Установить значение сопротивления в соответствии с методикой поверки 3. Зафиксировать усреднённые данные, полученные по трём замерам, в протокол калибровки\поверки 4. Повторять пункты 2 и 3 для всех точек прямого хода и обратного	10
015 Заключительная	Сформировать протокол поверки\калибровки	1. Высчитать значение основной приведенной погрешности по формуле, приведённой в методике проверки 2. Вынести заключение о канале	10

Исходя из технологического процесса, время, необходимое для калибровки\поверки одного ИК, равняется 20 минутам, 20480 минут для 1024 ИК, соответственно. Исходя из этого, основной проблемной является расходование значительных временных ресурсов на калибровку\поверку измерительной системы стенда статических испытаний АРД.

Для решения этой проблемы необходимо автоматизировать фиксирование данных, расчёт погрешности и формирование протокола. Также, для ускорения процесса калибровки\поверки, необходимо

предусмотреть одновременную работу с четырьмя каналами (одним модулем MR-212).

1.3 Методы решения проблемы

Программное обеспечение Recorder, работающее с универсальным многоканальным измерительно-вычислительным комплексом MIC-236, поддерживает механизм подключения дополнительных библиотек, для расширения функциональности. Такие библиотеки называются plug-in`ами. Использование библиотек plug-in`ов позволяет специализировать ПО «Recorder» для решения любых задач, возлагаемых на измерительный комплекс. [7] Данные plug-in`ы пишутся на языках C++ или Delphi. Исходя из этого основным методом решения поставленных задач является расширение функционала ПО «Recorder» путём создания специального plug-in`a.

Согласно рассмотренной функциональной схеме измерительного модуля MR-212, ИК одного модуля можно подключать одновременно, т.к. питание датчиков является общим. Для этого необходимо разработать специальный адаптер.

Вывод по главе

В главе были определены задачи автоматизации фиксирования данных, расчёта погрешности, формирования протокола и одновременная работа со всеми четырьмя каналами модуля. Так же были выбраны методы решения поставленных задач, именно:

1. Расширение функционала ПО «Recorder» путём написания plug-in`a, который будет отвечать за фиксацию данных и генерирование отчёта о поверке\калибровке.
2. Разработка специального адаптера для подключения измерительных каналов одной платы MR-212 к поверочной схеме.

2 Конструкторско-технологический раздел

В данном разделе происходит реализация программных модулей поверки\калибровки и управления виртуальными тегами. Происходит тестирование и предлагается дальнейшая модернизация системы измерения стенда статических испытаний.

2.1 Реализация программного модуля поверки\ калибровки

Для реализации программного модуля (plug-in`a) поверки\калибровки был выбран язык C++, так как пакет средств для создания плагинов, предоставляемого НПП “МЕРА”, для этого языка является более полным, по сравнению с пакетом для Delphi, а сам язык C++ - более современным [8].

Программный модуль взаимодействует с тегами напрямую, но в тоже время зависит от состояний самого ПО “Recorder” [9], это нужно учитывать при разработке модуля.

Таким образом для практического использования программного модуля необходимо:

– реализовать графический интерфейс для взаимодействия с plug-in’ом;

– разработать методы фиксации данных и балансировки нуля тегов;

– создать генератор отчёта о поверке\калибровке в файл с расширением “xls”.

2.1.1 Реализация графического интерфейса

Интерфейс plug-in’а обеспечивает взаимодействие пользователя с программным модулем и включает в себя такие группы элементов управления, как:

– обзор состояния ПО “Recorder” и управление им;

– выбор группы каналов и управлением ими;

– управление фиксацией данных;

– управление зафиксированными данными.

Интерфейс программного модуля поверки\калибровки представлен на рисунке 5.

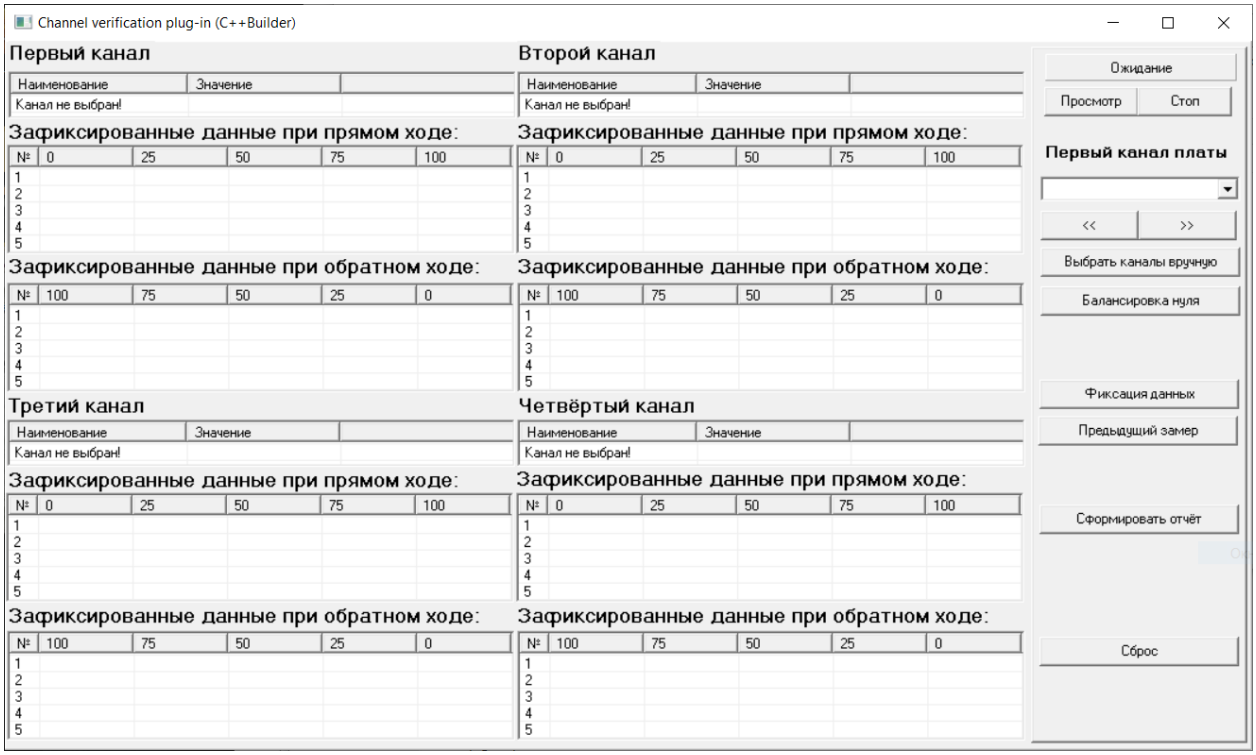

Рисунок 5 – Интерфейс программного модуля поверки\калибровки

Строка состояния отображает статус, возвращаемый ПО “Recorder”. Кнопка “Просмотр” переводит ПО “Recorder” в режим просмотра, если на данный момент это возможно, кнопка “Стоп” останавливает режим просмотра.

При выборе первого канала программа автоматически назначает следующие по индексу теги (если они имеются) для каналов 2,3 и 4. При нажатии на кнопки “<<” и “>>” происходит переход к предыдущей или к следующей группе тегов. Также каналы можно назначить вручную, если имеется такая потребность. Выбор каналов вручную продемонстрирован на рисунке 6.

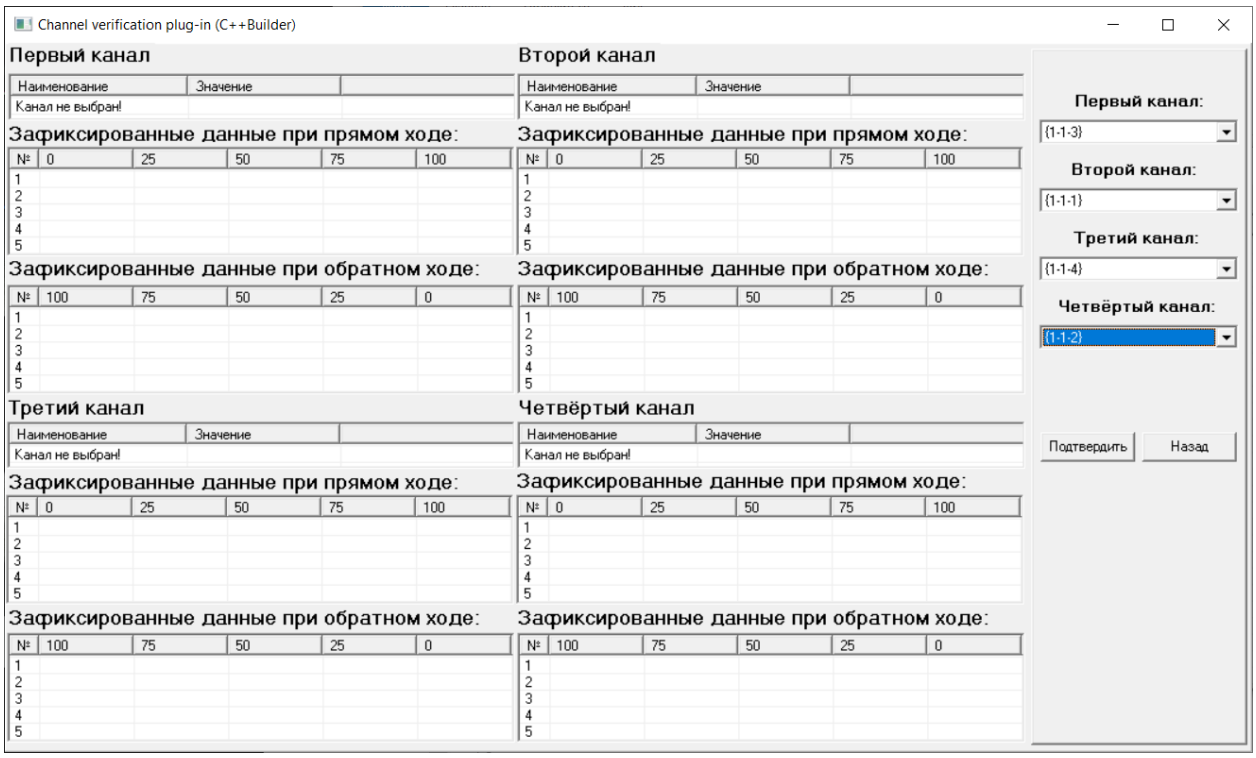

Рисунок 6 – Ручной выбор тегов

Назначение кнопок графического интерфейса:

“Балансировка нуля” запускает процесс приравнивания текущих значений выбранной группы тегов к нулю. Если тег связан с физическим каналом, а ПО “Recorder” находится в состоянии, не блокирующем выполнение балансировки нуля, данная команда применяется для данного тега.

“Фиксация данных” запускает таймер, по срабатыванию которого происходит фиксация текущего значения соответствующего тега в таблицу, закреплённую за данным тегом. Фиксация не будет происходить, если ПО “Recorder” не находится в состоянии просмотра или записи.

“Предыдущий замер” декрементирует значение счётчика фиксации «fVer.iteriteration[1]» и удаляет последние зафиксированные данные из таблиц.

“Сформировать отчёт” инициирует генерацию отчёта на основе заданного шаблона. Если фиксация данных не завершена, генерация не будет запущенна.

“Сброс” возвращает plug-in в исходное состояние, сохраняя выбранные теги.

2.1.2 Реализация программной части

2.1.2.1 Класс взаимодействия с тегами

Для взаимодействия с тегами, связанными с физическими каналами, был создан класс. В объект созданного класса передаётся ссылка на тег, с которым в данный момент взаимодействует программа.

Список публичных методов, описанных в классе «vTag»:

– void setTag(ITag* pTag) - передача ссылки на тег;

– string getName() - получение имени тега;

– double getValue() - получение текущего значения тега;

– bool checkTag() - проверка существования тега;

– bool setZero() - балансировка нуля.

– Список приватных методов и переменных, описанных в классе «vTag»:

– string name - имя тега;

– vector<TComInterface<ITag> > FTags - вектор существующих тегов;

– TComInterface<IRecorder> TRecorder - ссылка на Recorder;

– TComInterface<ITag> TTag - ссылка на тег, с которым взаимодействует программа;

– bool crtTag - индикатор связи объекта класса с существующим тегом;

– void updateTag() - обновление списка существующих тегов.

При вызове метода «setTag» в качестве аргумента передаётся ссылка на тег, которая фиксируется в приватную переменную «TTag», к которой во время работы программы будут обращается другие методы данного класса.

Метод «getName» возвращает значение типа string, хранящее имя тега, с которым связан объект класса. Если объект класса не связан с тегом, метод возвращает "Канал не выбран!".

Блок-схема метода getValue представлена на рисунке 7.

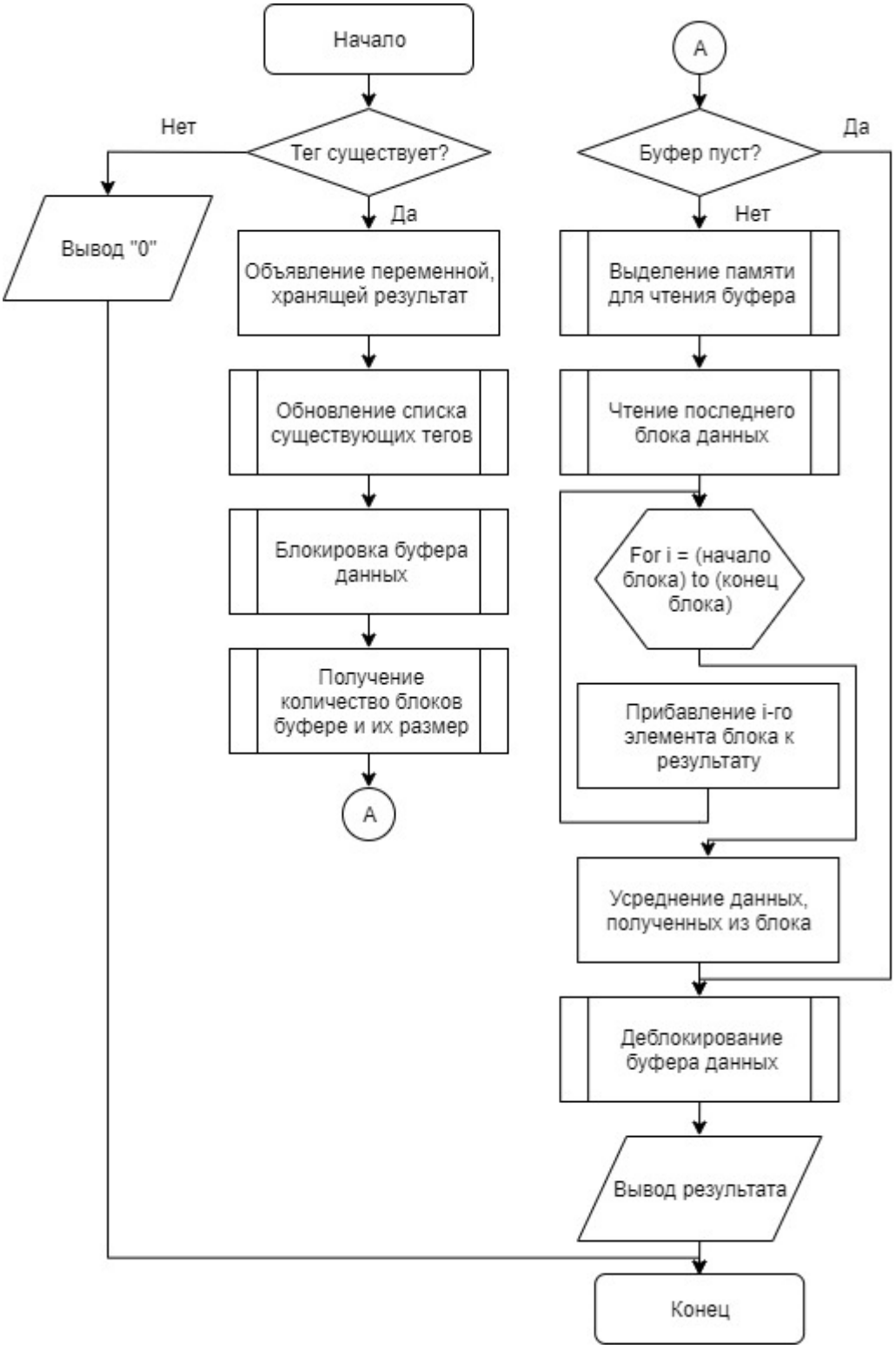

Рисунок 7 – Блок-схема метода getValue

Первым делом в методе проверяется, связал ли объект класса с существующим тегом, если да, то происходит обновление списка тегов, если нет – метод возвращает значение «0». Далее по ссылке на тег получаем ссылку на интерфейс блочного доступа, с помощью которого блокируется буфер с измеренными данными. Полученные данные усредняются и возвращаются методом в качестве результата своей работы. Перед завершением метода вектор необходимо разблокировать.

Метод «setZero» передаёт тегу команду на балансировку нуля. Данная команда применима только для тега, связанного с физическим каналом, поэтому метод проверяет не является ли тег виртуальным.

2.1.2.2 Фиксация данных

Для фиксации данных была создана структура, объект которой хранит в себе текущие итерации замеров, а также объекты класса взаимодействия с тегами (vTag).

```
struct sVer{
	short iteriteration[2];
	vTag mTag[4];
};
```

Как сказано в пункте 2.1 фиксация данных не может происходить, если ПО «Recorder» не находится в режиме просмотра или записи, это повлечет за собой ошибки, чтобы этого избежать ведётся отслеживание состояний ПО «Recorder», если оно переходит в необходимы режим, то происходит запуск таймера, который производит обновление текущих данных выбранных тегов в соответствующих таблицах. Пример отображения данных продемонстрирован на рисунке 8.

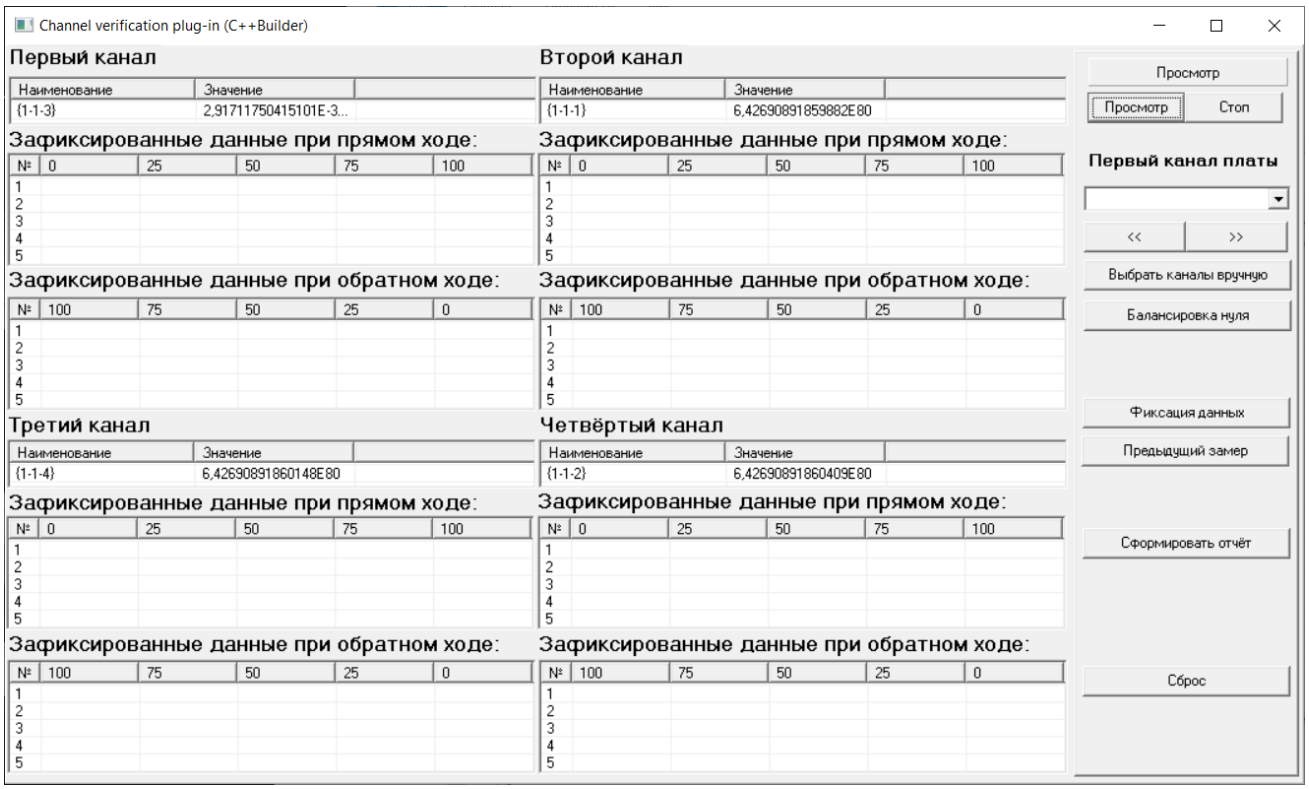

Рисунок 8 – Обновление данных

Когда каналы выбраны, а ПО «Recorder» находится в состоянии записи или просмотра, по нажатию на кнопку «Фиксация данных» запустится таймер с интервалом срабатывания в 600 миллисекунд. Блок-схема, метода вызываемого при срабатывании таймера представлен на рисунке 9.

В зависимости от текущего значения счётчиков определяется таблица и ячейка, в которую в данный момент необходимо занести значение соответствующего тега.

Таймеры отображения данных и фиксации отключаются, как только ПО «Recorder» переходит в режим, блокирующий фиксирование данных. Для отслеживания изменения состояния, ПО «Recorder» уведомляет об этом все работающие plug-in’ы.

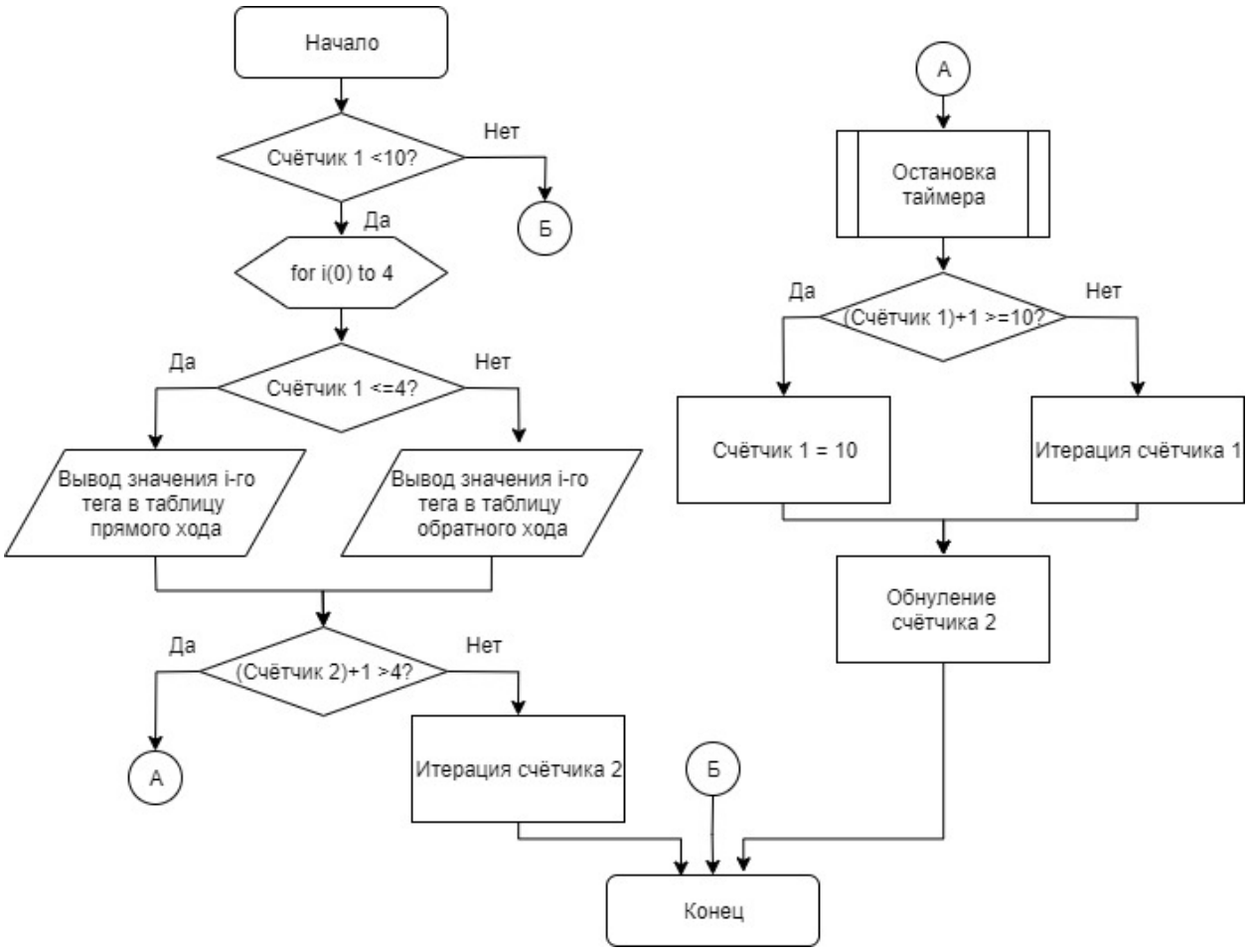

Рисунок 9 – Блок-схема метода вызываемого при срабатывании таймера

2.1.2.3 Генерация отчёта

Генерация отчёта инициируется нажатием кнопки «Сформировать отчёт» только тогда, когда зафиксированы все данные, то есть таблицы всех каналов заполнены, в ином случае выводится сообщение с предупреждением.

Для взаимодействия с ПО «Microsoft Excel» используется функция CreateOleObject, которая позволяет считывать и записывать данные в электронные таблицы.

Для сокращения объёма выводимых данных и для облегчения вносимых в отчёт изменений используется готовый шаблон отчёта, в котором происходит вычисление основной приведённой погрешности (γ) измерительных каналов по формуле (1.2), приведённой в существующей методике поверки [6].

$$\gamma = \frac{U_е - U_э}{U_в - U_н} * 100, \quad (1)$$

где: $U_е$ – измеренное значение относительного напряжения, мВ/В;

$U_э$ – номинальное значение;

$U_в, U_н$ – верхний и нижний пределы диапазона измерения, мВ/В.

Шаблон отчёта о поверке\калибровке располагается в папке «plugins» корневого каталога ПО «Recorder» и вызывается функцией «ExtractFilePath».

```
WideString fName = ExtractFilePath(Application->ExeName)+"plugins\\plgVer.xls";
```

После успешного открытия шаблона происходит фиксация имени тега и зафиксированных данных в заданные ячейки. Перед фиксацией, данные проверяются на допустимые символы:

```
String _tmpS;
for(int t(1);t<tmpS.Length();t++){
	String tmps=tmpS[t];
	if(tmps=="-" || tmps="," || tmps=="1" || tmps=="2" || tmps=="3" ||
	tmps=="4" || tmps=="5" || tmps=="6" || tmps=="7" || tmps=="8" || tmps=="9"
	|| tmps=="0"){
		_tmpS+=tmps;
	}
}
```

Проверка идёт по всем символам строки, считанной из таблицы. Если встречается символ, не входящий в список допустимых, он отбрасывается, иначе символ добавляется во временную переменную типа «string», которая по окончанию цикла зафиксируется в соответствующую ячейку отчёта.

2.1.3 Коммутация ИК к измерительному оборудованию

Для корректной работы программного модуля для поверки\калибровки ИК с входным сигналом на модули MR-212 необходимо разработать адаптер, рассчитанный на группу, состоящую из четырёх ИК и принадлежащих оному модулю MR-212, назначение выходов которого приведены в таблице 4.

Таблица 4 – Назначение контактов входных разъемов платы MR-212

№ контакта	Название	Назначение	№ контакта	Название	Назначение
1	AINR1	Калибровка канала 1	20	AINR3	Калибровка канала 3
2	AINR2	Калибровка канала 2	21	AINR4	Калибровка канала 4
3		Не используется	22	R4	Калибровка канала 4
4	- AIN4	-Вход канала 4	23	+AIN4	+Вход канала 4
5	- REFIN4	-Обратная связь канала 4	24	+REFIN4	+ Обратная связь канала 4
6	- EXC	-Питание датчиков	25	+EXC	+Питание датчиков
7		Не используется	26	R3	Калибровка канала 3
8	- AIN3	-Вход канала 3	27	+AIN3	+Вход канала 3
9	- REFIN3	- Обратная связь канала 3	28	+REFIN3	+ Обратная связь канала 3
10	- EXC	-Питание датчиков	29	+EXC	+Питание датчиков
11		Не используется	30	R2	Калибровка канала 2
12	- AIN2	-Вход канала 2	31	+AIN2	+Вход канала 2
13	- REFIN2	- Обратная связь канала 2	32	+REFIN2	+ Обратная связь канала 2
14	- EXC	-Питание датчиков	33	+EXC	+Питание датчиков
15		Не используется	34	R1	Калибровка канала 1
16	- AIN1	-Вход канала 1	35	+AIN1	+Вход канала 1
17	- REFIN1	- Обратная связь канала 1	36	+REFIN1	+ Обратная связь канала 1
18	- EXC	-Питание датчиков	37	+EXC	+Питание датчиков
19	AGND	Измерительная земля			

Существующая схема подсистемы измерения напряжений и деформаций подразумевает, что в групповой кабель сформированы по восемь каналов, то есть выходы с двух модулей MR-212. Групповой кабель оканчивается выходным разъёмом 2РМТ27Б24Г, схема подключения отображена на рисунке 10.

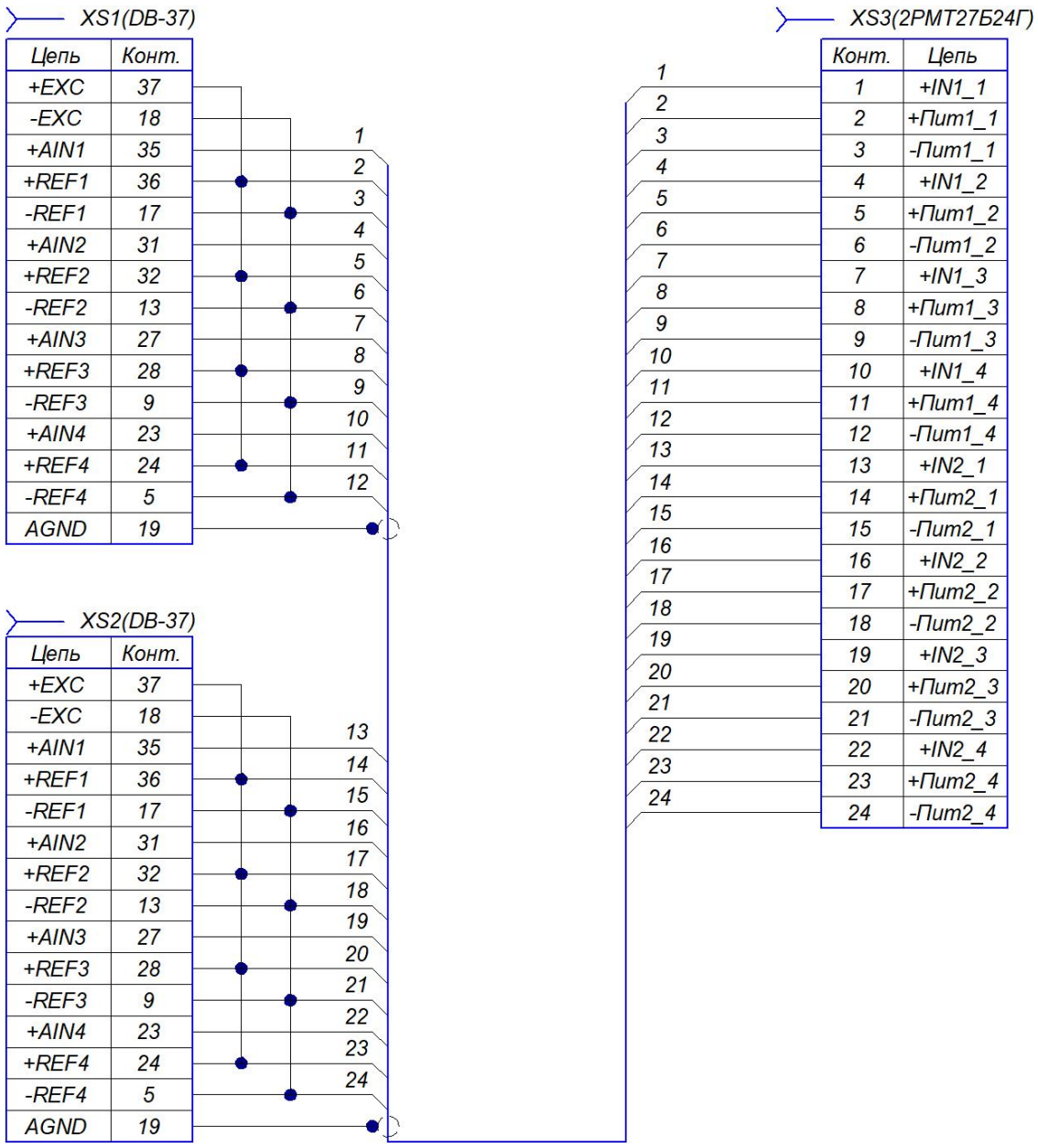

Рисунок 10 – Схема подключения группового кабеля к модулям MR-212

С учётом того, что питание датчиков (±EXC) является общим для всех четырёх каналов платы MR-212, подключение этих каналов к одной поверочной\калибровочной схеме допустимо, т.к. не искажает получаемый сигнал.

В результате описанного выше необходимо распаять кабель в соответствии со схемой подключения, представленной на рисунке 11.

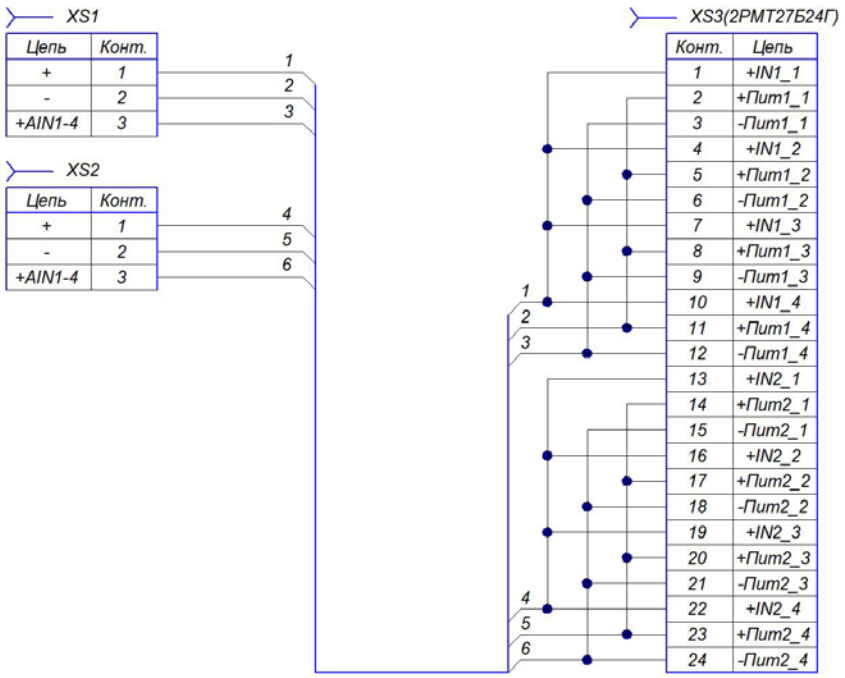

Рисунок 11 – Схема подключения группового кабеля к поверочной\калибровочной схеме

2.2 Реализация программного модуля управления виртуальными тегами

Для отладки и проверки работоспособности программного модуля поверки\калибровки необходимо разработать plug-in, который в свою очередь должен иметь следующий функционал:

– создание и удаление групп тегов;

– создание, изменение и удаление виртуальных тегов;

– имитация различных видов сигналов.

Таким образом для практического использования программного модуля необходимо:

реализовать графический интерфейс для взаимодействия с plug-in’ом;

– изучить и реализовать метод создания\удаления виртуальных тегов;

– изучить и реализовать метод передачи значения виртуальному тегу;

– реализовать создание\удаление групп тегов;

– реализовать генерацию различных видов сигналов;

– реализовать возможность изменения типа тега.

2.2.1 Реализация графического интерфейса

Графический интерфейс программного модуля управления виртуальными тегами можно разделить на блоки:

– отображения данных и информации о виртуальных тегах;

– управление состоянием plug-in’а;

– создание\удаление группы тегов;

– создание различных типов тегов;

– удаление выбранного тега;

– изменение типа тега.

Для разделения данных блоков на форме используется компонент «TPageControl», который позволяет управлять вкладками, вследствие этого отпадает необходимость в создании дополнительных форм под каждую реализуемую функцию интерфейса, таки образом используется два компонента, один для отображения данных, другой для управления plug-in’ом. Общее количество страниц, используемых для управления plug-in’ом равняется восьми, для отображения данных – варьируется, в зависимости от количества созданных групп тегов, но не меньше одной - общей.

На компоненте «TPageControl», предназначенном для отображения данных, первоначально создана одна вкладка, на которой расположена таблица для фиксирования в неё информации и значений всех созданных виртуальных тегов, независимо от их группы. При создании новой группы тегов автоматически создаётся новая вкладка, носящая имя группы. Код создания группы и вкладки будет описан в следующем пункте.

Интерфейс программного модуля управления виртуальными тегами представлен на рисунке 12.

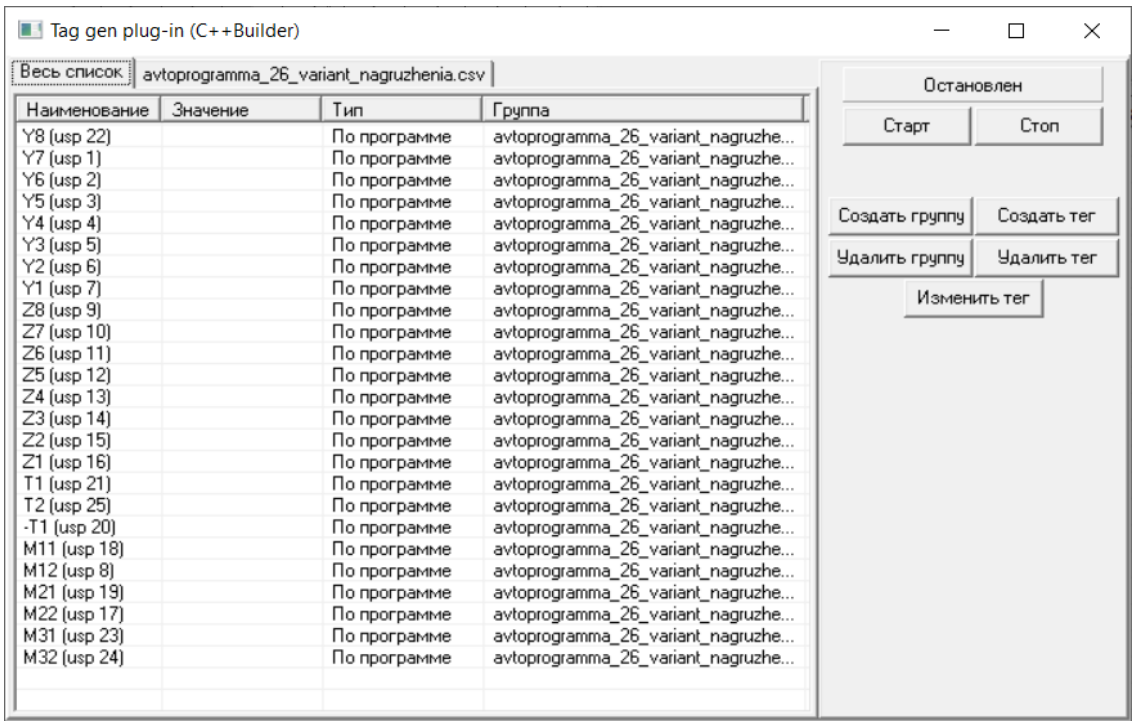

Рисунок 12 – Интерфейс программного модуля управления виртуальными тегами

По умолчанию у компонента «TPageControl», предназначенного для управления plug-in’ом, активна вкладка «root_pg», на которой расположена строка состояния plug-in’а, а также кнопки «Старт» и «Стоп» для управления этим состоянием. Остальные кнопки, в зависимости от их назначения, отвечают за переключение активной вкладки компонента «TPageControl. Наименование и назначение вкладок перечислены в таблице 5.

Таблица 5– Вкладки компонента «TPageControl»

Наименование	**Назначение**
root_pg	Вкладка, с основными элементами управления plug-in’ом.
ctrG_pg	Вкладка создания группы тегов.
crtT_pg	Основная вкладка создания виртуальных тегов, где производится выбор их типа и группы.
crt_prog	Создание тега, значения которого изменяется по заранее созданной программе.
crt_const	Создание тега с постоянным значением.
crt_var	Создание тега с изменяющимся значением.
delG_pg	Удаление группы тегов.
delT_pg	Удаление выбранного тега.

2.2.2 Реализация программной части

Исходя из того, что необходимо реализовать имитацию различных видов сигнала, было принято решение о создании четырёх типов виртуальных тегов:

– «постоянный» – значение тега остаётся неизменным во времени;

– «переменный» – значение тега меняется с заданным шагом в установленных границах (минимум и максимум);

– «плавный» – значение тега меняется с заданным шагом от начального к конечному;

– «по программе» – значение тега изменяется в соответствии с заранее созданным файлом конфигурации.

2.2.2.1 Класс управления виртуальными тегами

Класс управления виртуальными тегами был разработан для взаимодействия plug-in’a с интерфейсами ITag, IBlockAccess и ISignal. Интерфейс ITag предназначен для получения информации о теге. Интерфейсы IBlockAccess и ISignal предоставляют доступ к буферу данных тега.

Список публичных методов, описанных в классе «vTag»:

– vTag(IRecorder* pRecorder) - переопределенный конструктор, предназначенный для того, чтобы запомнить ссылку на IRecorder;

– bool CreateTag(string) - создание виртуального тега с заданным именем;

– void setConstTag(double) - настройка параметров для тега с фиксированным значением;

– void setVariableTag(double,double,double) - настройка параметров для тега с переменным значением;

– void setSmoothlyTag(double,double,double) - настройка параметров для тега с плавно изменяемым значением;

– void setProgramTag(vector<program_step>) - настройка параметров для тега с значением изменяемым по программе;

– void NextValue() - следующее значение (для изменяемых тегов);

– double getValue() - получение текущего значение тега;

– string getName() - получение имени тега;

– bool deleteTag() - удаление виртуального тега;

– string getType() - получение типа тега;

– void PrepareTagsView() - подготовка просмотра;

– bool getVarFlg()– проверка, достиг ли тег заданного значения;

– bool checkTag() - проверка существования тега.

Список приватных методов и переменных, описанных в классе «vTag»:

– string name - имя тега;

– double value, initV, finV, step - заданное значение тега, начальное и конечное значения, шаг (в сек.);

– int type, i_vector, timeV - тип тега (0-потоянный,1-переменный, 2-плавный, 3-по программе), итератор перехода по программе тега, время на шаге;

– bool end_flg, var_flg - флаг завершения программы (для тега с заданной программой), флаг переменного и программного тегов;

– vector<program_step> pStep - вектор шагов (для тега с заданной программой);

– typedef vector<TComInterface<ITag> > tagslist - переменная вектора тегов;

– tagslist FTags – перечень существующих тегов;

– TComInterface<IRecorder> TRecorder -переменная для сохранения ссылки на Recorder;

– TComInterface<ITag> TTag - переменная для сохранения ссылки на тег;

– bool crtTag - индикатор создания виртуального тега;

– void cleanParam() – сброс параметров тега;

– bool checkState() - проверка состояния ПО «Recorder»;

– void updateTag() - обновление списка существующих тегов.

В методе «CreateTag», блок-схема которого представлена на рисунке 13, происходит создание виртуального тега.

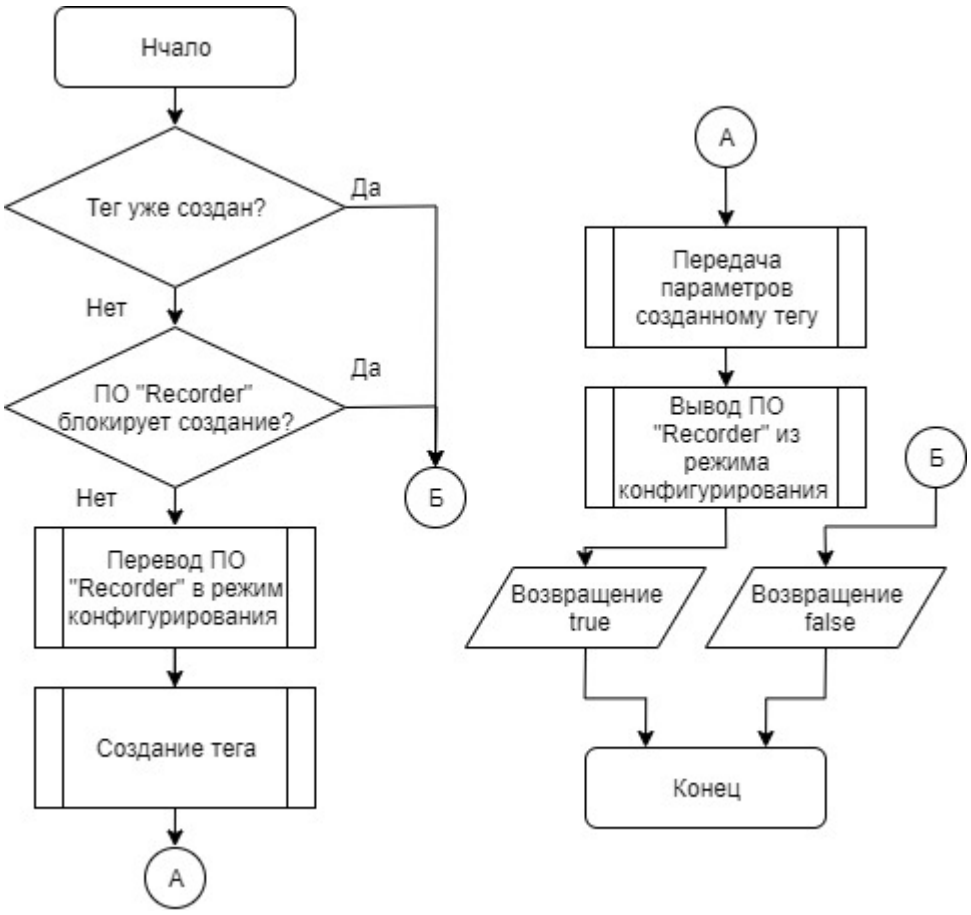

Рисунок 13 – Блок-схема метода «CreateTag»

В данный метод в качестве аргумента передается переменная типа «string», содержащая имя создаваемого тега. Перед тем, как перейти дальше, проверяется уникальность имени, а также состояние ПО «Recorder».

Исходя из руководства по созданию plag-in’ов [10] для создания и удаления виртуального тега ПО «Recorder» должно находится в состоянии конфигурирования, поэтому в методе предусмотрен перевод в данное состояние. После чего происходит создание нового тега и передача ему свойств через интерфейс TTag.

По окончанию создания тега ПО «Recorder» выводится из режима конфигурации.

Методы «setConstTag», «setVariableTag», «setSmoothlyTag» и «setProgramTag» устанавливают соответствующие значения переменных объекта класса, передаваемых им в качестве аргументов, а также устанавливают тип созданного виртуального тега.

Для изменения значение тега реализован метод «NextValue», в котором описаны алгоритмы изменения значения тега в зависимости от его типа:
Для типа тега «постоянный» устанавливается то значение, которое хранится в переменной типа double «value». Это значение устанавливается в момент инициализации тега методом «setConstTag»;

1. Алгоритм перехода на следующее значение для тега с переменным значением проверяет, не достигло ли значение тега одной из границ, если границ достигнута, то алгоритм меняет знак шага. Значения будут меняться, пока генерация не будет остановлена;
2. Изменение значения у тега типа «плавный» происходит путём изменения текущего значения на заданный шаг. Алгоритм прекращает изменение при достижении конечного значения вплоть до остановки генерации;
3. Первым делом для тега, изменяемого по программе, вычисляется значение шага, путём деления разности значения тега и значения текущего шага программы. Далее идёт изменение текущего значения тега на рассчитанный шаг, пока не будет достигнуто заданное значение, после чего оно будет удерживаться установленное количество времени. Если текущий шаг программы не является последним, алгоритм повторяется для следующего.

2.2.2.2 Создание группы тегов

Для удобства управления тегами и отображения их информации каждый создаваемый тег должен быть определён к одной из созданных групп. Для создания группы необходимо нажать на соответствующую кнопку, которая

переключит активную вкладку компонента «TPageControl» на «ctrG_pg», интерфейс которой представлен на рисунке 14.

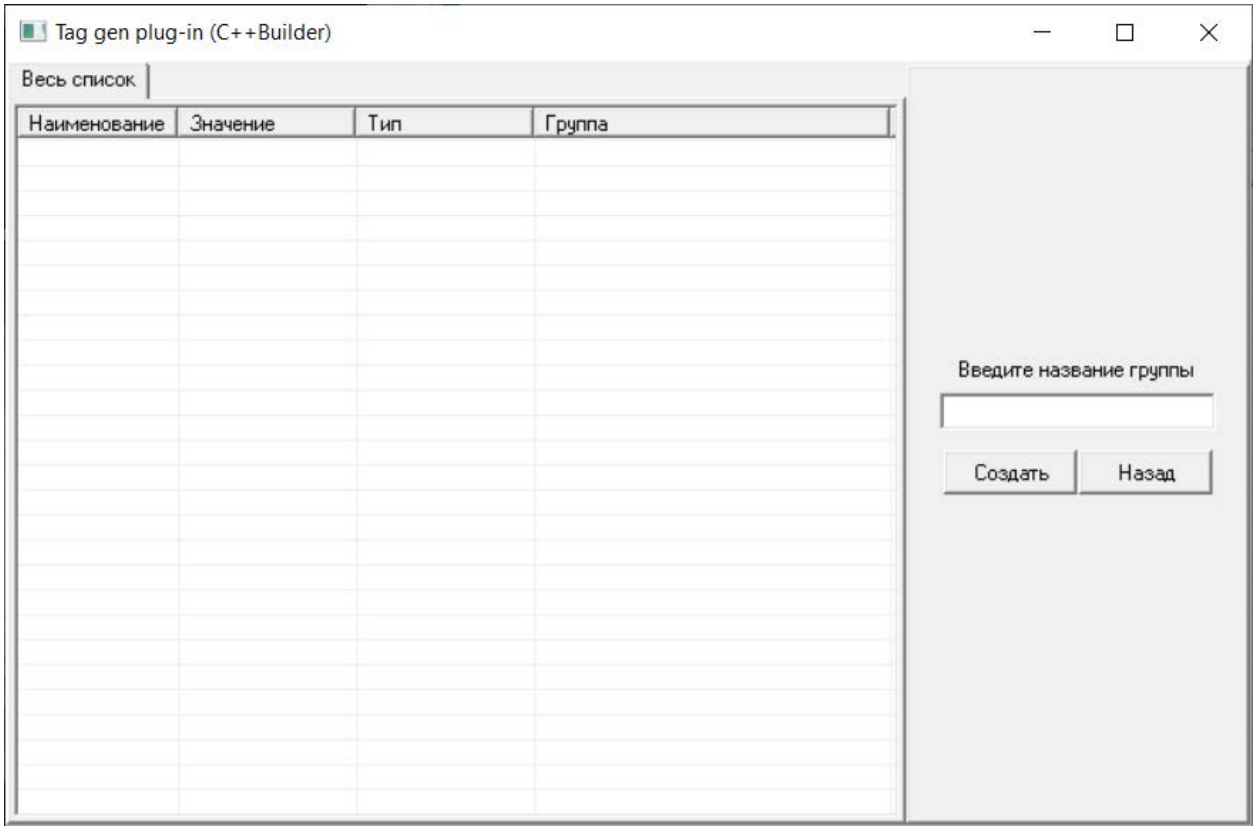

Рисунок 14 – Вкладка создания группы тегов

Группа тегов представляет собой структуру, в которой содержится такие элементы, как:

– name (String) – имя группы;

– TTS (TTabSheet) – закреплённая за группой вкладка;

– TLV (TListView) – закреплённая за группой таблица;

– sinch (bool) – флаг синхронности тегов (для программных);

– var_flg (bool) – достигли ли все теги заданного значения (для програмных);

– FvTag (vector<vTag>) – вектор объектов класса «vTag», принадлежащих данной группе.

Имя каждой группы тегов уникально, поэтому при создании новой группы тегов происходит проверка имен уже существующих групп. Также

каждой вновь создаваемой группе присваивается собственная вкладка отображения данных, которая создаётся как дубликат основной. Блок-схема метода создания группы тегов представлена на рисунке 15.

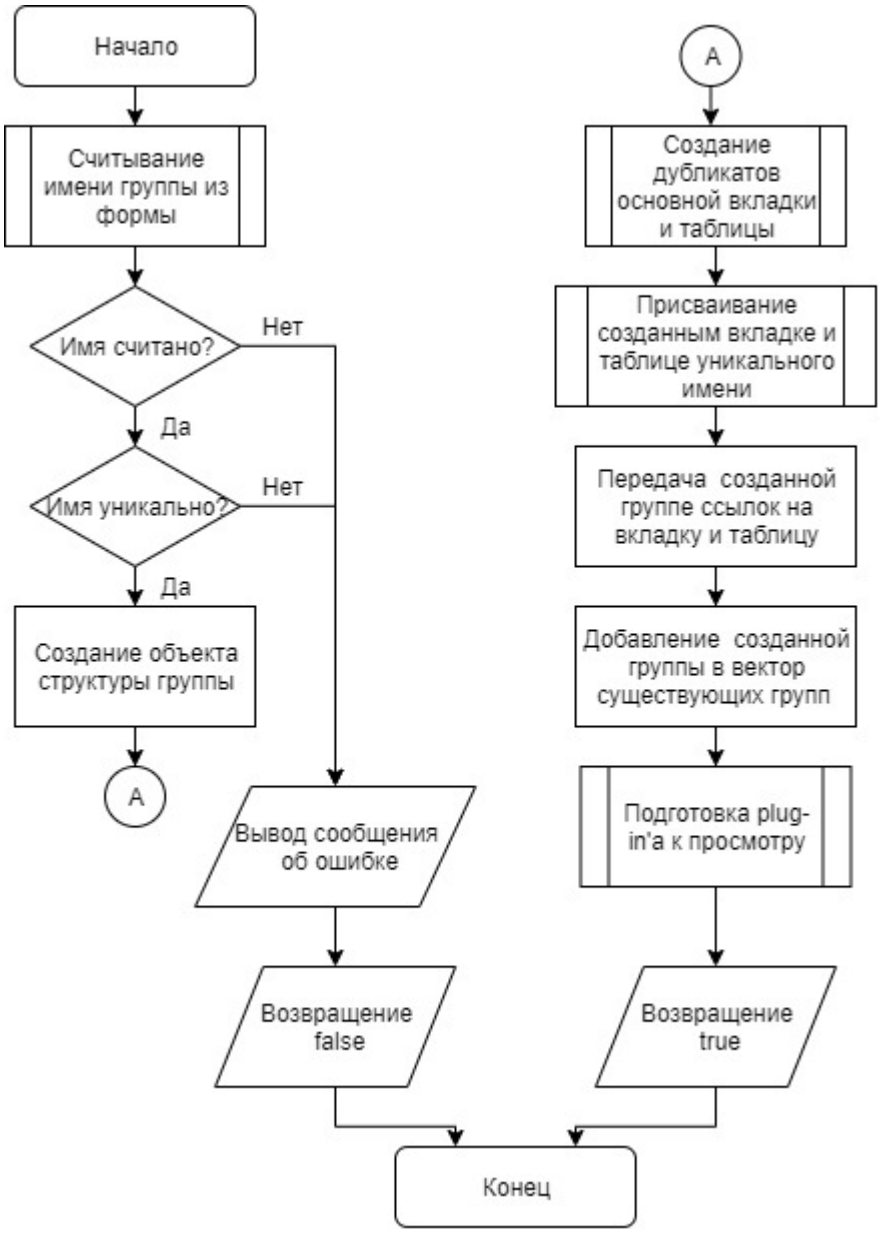

Рисунок 15 – Блок-схема метода создания группы тегов

2.2.2.3 Создание тега, изменяемого по программе

Создание тега, изменяемого по программе, осуществляется путём считывания необходимых параметров из файла с расширением «xls», пример конфигурации представлен на рисунке 16.

	A	B	C	D	E	F
1		Наименование	{1-1-1}	{1-1-2}	{1-1-3}	{1-1-4}
2	Шаг					
3						
4	1	усилие	-3037,38	-3037,38	-3037,38	-3037,38
5		время	6	6	6	6
6		скорость	1	1	1	1
7	2	усилие	-1542,06	-1542,06	-1542,06	-1542,06
8		время	5	5	5	5
9		скорость	1	1	1	1
10	3	усилие	0	0	0	0
11		время	5	5	5	5
12		скорость	1	1	1	1

Рисунок 16 – Пример конфигурационного файла

В файле конфигурации задаётся:

– количество создаваемых тегов и их имена;

– количество шагов программы;

– значение тега на каждом шаге;

– время в секундах, которое программа должна удерживать заданное значение, прежде чем перейти к следующему шагу;

– количество времени в секундах, за которое программа должна установить значение, заданное на шаге.

Считывание файла происходит путём открытия диалогового окна по средствам компонента «TOpenDialog», в котором указывается путь к файлу конфигурации. После этого происходит считывание данных по средствам метода «CreateOleObject».

2.3 Тестирование программного модуля поверки\калибровки

Для тестирования разработанного plug-in’а был сконфигурирован файл, содержащий в себе информацию о четырёх тегах и девяти шагах программы. Шаги программы приставлены в таблице 6.

Таблица 6 – Параметры файла конфигурации

Шаг	Значение	Время	Скорость
1	-3037,38	6	1
2	-1542,06	5	1
3	0	5	1
4	1542,056	5	1
5	3037,38	7	1
6	1542,056	5	1
7	0	5	1
8	-1542,06	5	1
9	-3037,38	5	1

Результаты работы программных представлены на рисунках 17 и 18.

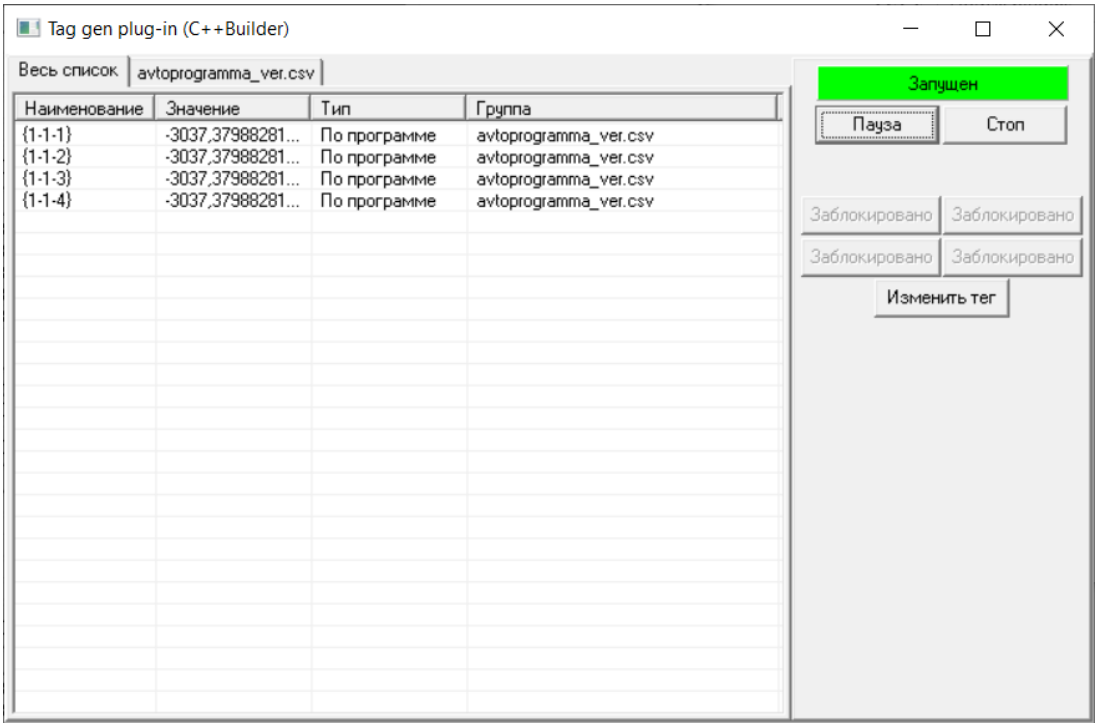

Рисунок 17 – Имитация сигнала измерительных каналов

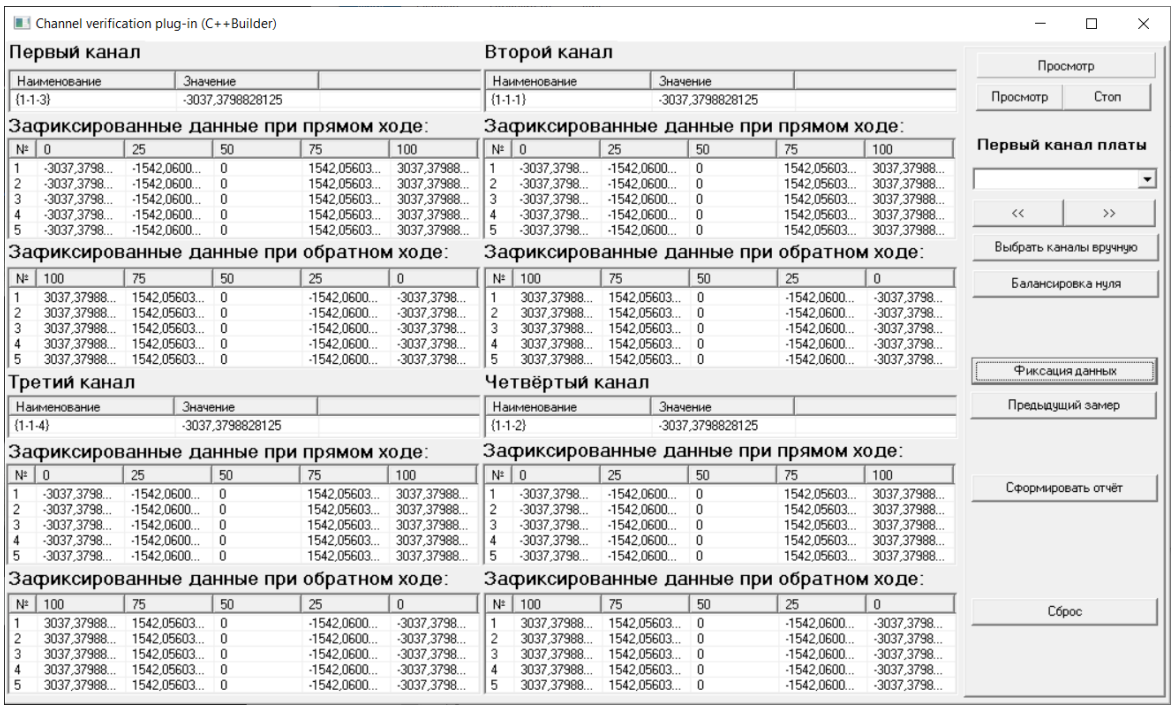

Рисунок 18 – Фиксация данных

Сгенерированные отчёты представлены в приложении Г.

2.3.1 Возможные пути модернизации

Вследствие то, что в схеме поверки\калибровки используется мера электрического сопротивления постоянного тока многозначная типа Р3026-1 (рисунок 19), установка значений осуществляется вручную.

Рисунок 19 – Мера электрического сопротивления постоянного тока многозначная типа Р3026-1

Для автоматизации этого процесса следует заменить Р3026-1 на современный калибратор, которые поддерживают управление с помощью COM-порта [11], например Н4-14, или Н4-56. Эта замена позволит управлять выходом калибратора из plug-in’а с помощью набора команд, позволив значительно сократить время поверки\калибровки.

Вывод по главе

В данной главе были реализованы графическая и программная части plug-in’а поверки\калибровки измерительных каналов системы измерения стенда статических испытаний, имеющего следующий функционал:

- автоматический выбор тегов по первому каналу платы;
- ручной выбор каналов;
- балансировка нуля выбранных тегов;
- переход к следующей группе тегов, возврат к предыдущей;
- автоматическая фиксация данных;
- генерация отчёта о поверка\калибровке.

Помимо этого, была предложена схема подключения, которая обеспечит одновременную поверку\калибровку четырёх измерительных каналов одной платы MR-212.

Для тестирования и отладки программного модуля поверки\калибровки были реализованы графическая и программная части plug-in’а управления виртуальными тегами, имеющего следующий функционал:

- отображения данных и информации о виртуальных тегах;
- управление состоянием plug-in’а;
- создание\удаление группы тегов;
- создание различных типов тегов;
- удаление выбранного тега;
- изменение типа тега.

С помощью разработанного plug-in’а был отлажен и протестирован программный модуль поверки\калибровки.

Также была рассмотрена возможность дальнейшая модернизация программного модуля и процесса поверки\калибровки в целом, путём замены устаревшего оборудования на современный калибратор. Вывод по главе

3 Экономический раздел

Экономическое обоснование проектов необходимо при создании любой новой сущности в той или иной мере. Если разрабатываемый продукт не является более выгодным и конкурентоспособным, то нет причин для его разработки. Так и в данном проекте нужно знать стоимость разработанного программного модуля и его эффективность относительно предыдущего способа выполнения работы. Так как работы производились ручной установкой устройства съемки в необходимое положение, то предположительная производительность проекта с манипулятором высока. Так же необходимо рассчитывать стоимость разработанной программ, как её разработчику.

3.1 Методика расчета стоимости программного продукта

Рассмотренная ниже методика в полной мере описана в методическом пособии [12]. В данном пункте будут приведены лишь те части методики, которые использованы в проекте.

Затраты на программный продукт меняются в соответствии с этапами жизненного цикла программного продукта. Затраты в соответствии с этапами:

– $С_р$ – совокупные затраты на разработку программ

– $С_э$ – затраты на эксплуатацию программ и аппаратные средства ЭВМ, реализующей данный программный продукт

– $С_с$ – затраты на сопровождение программного продукта за время $t_с$, включающие затраты на хранение и контроль его состояния, проведение модернизаций и исправление ошибок, тиражирование и т.д.

Для работы потребуются лишь затраты на разработку программы. По методике затраты равны произведению трудоемкости проекта в человеко-месяцах и цены одного человеко-месяца.

$$С_р = ЧМ * Ц_{чм}, \quad (2)$$

где: ЧМ – трудоемкость проекта в человеко-месяцах;

$Ц_{чм}$ – цена одного человеко-месяца.

Цена одного человеко-месяца включает в себя большое количество параметров, такие как: затраты на содержание и эксплуатацию основных фондов, затраты на содержание управленческого персонала, среднюю зарплату работников, отчисления на социальные нужды. Но в данном проекте нет такого количества известных данных, поэтому берется средняя зарплата начинающего программиста 20 т.р.. Трудоемкость проекта в человеко-месяцах (3) так же определяется по методике.

$$ЧМ = 3 * КЧИК^{1,12} * \prod_{ji} C_{ij}, \quad (3)$$

где: КЧИК – число исходных команда в тысячах;

C_{ij}–коэффициенты изменения трудоемкости.

Коэффициенты C_{ij} отражают изменение трудоемкости непосредственной разработки строки текста программы за весь цикл создания программного продукта при воздействии ij-фактора.

Атрибуты создаваемого программного продукта:

– ТНПП – требуемая надежность программного продукта

– СПП – сложность программного продукта

– РБД – размер базы данных

– МК – мобильность (переносимость) использования компонентов программного продукта для других разработок

Атрибуты ЭВМ:

– ОБД – ограничение по быстродействию

– ОП – ограничение по оперативной памяти

– Атрибуты исполнителей:

– КА – квалификация аналитика

– КП – квалификация программиста

– КЗ – квалификация заказчика

Атрибуты проекта:

– ПСМ – применение современных методов разработки программного продукта

– ИИС – использование инструментальных средств

Каждому из указанных стоимостных атрибутов соответствует коэффициент C_{ij}, характеризующий влияние атрибута на программную разработку. В таблице 7 представлены численных характеристики коэффициентов.

Таблица 7– Численные характеристики коэффициентов

Фактор трудоемкости	Коэффициент для рейтингов				
	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий
ТНПП	0.75	0.88	1	1.15	1.4
СПП	0.43	0.87	1	1.69	2-3
РБД	-	0.94	1	1.08	1.16
МК	-	-	1	1.15	1.53
ОБД	-	-	1	1.67	3.33
ОП	-	-	1	1.67	3.33
КА	1.46	1.19	1	0.86	0.71
КП	1.42	1.17	1	0.86	0.7
КЗ	1.23	1.11	1	0.91	0.85
ПСМ	1.24	1.1	1	0.91	0.82
ИИС	1.24	1.1	1	0.91	0.83

Последний параметр КЧИК вычислен по формуле (4).

$$КЧИК = ЧИК/10^3, \quad (4)$$

где: КЧИК – число исходных команд в тысячах;

ЧИК – число исходных команд.

3.2 Вычисление стоимости программного модуля

Дальнейшие параметры выбраны с помощью таблицы 7. Исходя из спроектированной программы и требований технического задания были выбраны коэффициенты для рейтингов:

– ТНПП = 0.88
– ССП = 0.87
– КА = 1.46
– КП = 1
– КЗ = 1.23
– ПСМ = 1.1
– ИИС = 1.1

Параметр КЧИК взят за 2 тысяч строк кода, с учетом разработки программного модуля для тестирования, тогда по формуле (2.4) ЧМ равен 10,85. Конечное значение затрат равно 216,95 тысячам рублей по первой формуле раздела (2).

3.3 Эффективность внедрения программного модуля

При общении с заказчиком была получена оценка выполнения поверки\калибровки без использования дополнительного ПО. При ручном фиксировании данных и заполнении отчёта поверка\калибровка одного канала занимает 1200 секунд, 4800 секунд для четырёх каналов соответственно. Так же примерная оценка заказчиком времени, затрачиваемого на поверку\калибровку четырёх каналов с использованием программного модуля поверки\калибровки, составляет 630 секунд. Исходя из этого равен разности этого времени (5).

$$Э = t_р - t_м = 4800 - 630 = 4170 \text{ сек.}, \quad (5)$$

где: $t_р$ – время поверки\калибровки без использования дополнительного ПО;

t_m – время поверки\калибровки с использованием программного модуля.

Вывод по разделу

После применения методики определения затрат на проектирование программного продукта были получены затраты на разработку, которые составили 216,95 тысяч рублей. Использование программного модуля позволило увеличить производительность в 7,6 раз, что делает разработку и использование программного модуля целесообразным.

4 Защита информации

Под информационной безопасностью Российской Федерации (информационной системы) подразумевается техника защиты информации от преднамеренного или случайного несанкционированного доступа и нанесения тем самым вреда нормальному процессу документооборота и обмена данными в системе, а также хищения, модификации и уничтожения информации.

Вопросы защиты информации в информационных системах решаются для того, чтобы изолировать нормально-функционирующую информационную систему от несанкционированных управляющих воздействий и доступа посторонних лиц или программ к данным с целью хищения.

Под фразой «угрозы безопасности информационных систем» понимаются реальные или потенциально возможные действия или события, которые способны исказить хранящиеся в информационной системе данные, уничтожить их или использовать в каких-либо целях, не предусмотренных регламентом заранее.

4.1 Определение класса защиты

Согласно руководящему документу «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации» [13] устанавливается семь классов защищенности СВТ от НСД к информации. Самый низкий класс – седьмой, самый высокий – первый.

Классы подразделяются на четыре группы, отличающиеся качественным уровнем защиты:

– первая группа содержит только один седьмой класс;

– вторая группа характеризуется дискреционной защитой и содержит шестой и пятый классы;

– третья группа характеризуется мандатной защитой и содержит четвертый, третий и второй классы;

– четвертая группа характеризуется верифицированной защитой и содержит только первый класс.

Перечень показателей по классам защищенности СВТ приведен в таблице 8.

Обозначения:

«-» – нет требований к данному классу;

«+» – новые или дополнительные требования,

«=» – требования совпадают с требованиями к СВТ предыдущего класса.

Таблица 8 – Показатели по классам защищенности

Наименование показателя	Класс защищенности					
	6	5	4	3	2	1
Дискреционный принцип контроля доступа	+	+	+	=	+	=
Мандатный принцип контроля доступа	-	-	+	=	=	=
Очистка памяти	-	+	+	+	=	=
Изоляция модулей	-	-	+	=	+	=
Маркировка документов	-	-	+	=	=	=
Защита ввода и вывода на отчуждаемый физический носитель информации	-	-	+	=	=	=
Сопоставление пользователя с устройством	-	-	+	=	=	=
Идентификация и аутентификация	+	=	+	=	=	=
Гарантии проектирования	-	+	+	+	+	+
Регистрация	-	+	+	+	=	=
Взаимодействие пользователя с КСЗ	-	-	-	+	=	=
Надежное восстановление	-	-	-	+	=	=
Целостность КСЗ	-	+	+	+	=	=
Контроль модификации	-	-	-	-	+	=
Контроль дистрибуции	-	-	-	-	+	=
Гарантии архитектуры	-	-	-	-	-	+
Тестирование	+	+	+	+	+	=
Руководство для пользователя	+	=	=	=	=	=
Руководство по КСЗ	+	+	=	+	+	=
Тестовая документация	+	+	+	+	+	=
Конструкторская (проектная) документация	+	+	+	+	+	+

Седьмой класс присваивают СВТ, к которым предъявлялись требования по защите от НСД к информации, но при оценке защищенность СВТ оказалась ниже уровня требований шестого класса.

Разрабатываемый программный модуль можно отнести к классу защиты 4, т.к. предполагается разграничение доступа к программному модулю на производстве и защита ввода и вывода на отчуждаемый физический носитель информации.

Персонал, использующий программный модуль несёт ответственность за соблюдение условий и сроков хранения, сохранность и конфиденциальность результатов измерения (протоколов).

Вывод по разделу

В данном разделе были рассмотрены классы защиты средств вычислительной техники, было произведено определение класса защиты программного модуля поверки\калибровки. Программный модуль соответствует 4 классу защиты.

Ответственность за соблюдение условий и сроков хранения, сохранность и конфиденциальность результатов измерения (протоколов) возлагается на персонал, использующий программный модуль.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе были выделены основные требования к программному модулю, проанализирована методика поверки измерительных каналов, спроектированны программный модули поверки\калибровки и модуль для его тестирования.

Результатом выпускной квалификационной работы являются программный модуль поверки\калибровки измерительных каналов системы измерения стенда статических испытаний, при создании которого были соблюдены требования технического задания, а также программный модуль управления виртуальными тегами для тестирования и отладки.

Программный модуль поверки\калибровки решает все поставленные в выпускной квалификационной работе задачи, а именно:

- подготовка измерительных каналов к поверке\калибровке;
- автоматизация фиксирования данных;
- генерация отчётов по результатам измерений.

Программный модуль управления виртуальными тегами помимо задачь тестирования и отладки, может быть полезен при дальнейших разработках ПО для системы измерения стенда статических испытаний, либо использоваться для проведения лабораторных работ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Горбунов Г.М., Солохин Э.Л. «Испытания авиационных воздушно-реактивных двигателей». М.: Машиностроение, 1967

2. Леонтьев М.К. «Тензометрирование в авиационных газотурбинных двигателях». Учебное пособие. - М.: Изд-во МАИ, 2000

3. НПО «Мера», ОАО «СТЕНД СТАТИЧЕСКИХ ИСПЫТАНИЙ ДВИГАТЕЛЯ ПД-14 ОАО «Авиадвигатель»”. Руководство по эксплуатации - БЛИЖ.401201.300.441 РЭ, 2013

4. MIC-236 Универсальный многоканальный магистрально-модульный измерительный комплекс [Электронный ресурс] / URL: http://www.nppmera.ru/mic-236 (дата обращения: 29.10.19)

5. MR-212 Модуль для работы с тензометрическими датчиками [Электронный ресурс] / URL: http://www.nppmera.ru/mr-212 (дата обращения: 29.10.19)

6. НПО «Мера», “КОМПЛЕКС ИЗМЕРИТЕЛЬНО-ВЫЧИСЛИТЕЛЬНЫЙ MIC”. Методика поверки - БЛИЖ.40 1250.001МП, 2014

7. Recorder открытое ПО для универсального применения [Электронный ресурс] / URL: http://www.nppmera.ru/recorder (дата обращения: 29.10.19)

8. Шилдт Герберт "C++. Полное руководство. Классическое издание". Вильямс, 2019 г.

9. НПО «Мера», «Программное обеспечение сбора измерительных данных Recorder. Разработка plug-in`ов.». Руководство программиста - г. Королёвта, 2004

10. НПО «Мера», «Инструментальное программное обеспечение измерительного комплекса ПО “Recorder”». Руководство программиста, 2004

11. Michael R. Sweet «Serial Programming Guide for POSIX Operating Systems», 5th Edition Copyright 1994-1999, All Rights Reserved

12. Старков Ю.В., Тимофеева Г.А. Определение затрат на проектирование программного продукта: методическое пособие. Пермский государственный технический университет. 2006 г. 16 с.

13. Руководящий документ Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. Классификация автоматизированных систем и требования по защите информации. -М.: Гостехкомиссия России, 1992.

ПРИЛОЖЕНИЕ А.

Исходный код программного модуля и интерфейса поверки\калибровки

VerFormUnit.h

```
struct sVer{
	short iteriteration[2];
	vTag mTag[4];
};

//---------------------------------------------------------------------------
class TVerForm : public TForm
{
__published: // IDE-managed Components
	TTimer *TimeTimer;
	TTimer *DataUpdateTimer;
	TPanel *DataPanel;
	TListView *TagsListView1;
	TStaticText *StaticText1;
	TStaticText *StaticText2;
	TListView *TagListFixS1;
	TStaticText *StaticText3;
	TListView *TagListFixR1;
	TStaticText *StaticText4;
	TListView *TagsListView2;
	TStaticText *StaticText5;
	TListView *TagListFixR2;
	TListView *TagListFixS2;
	TStaticText *StaticText6;
	TStaticText *StaticText7;
	TStaticText *StaticText8;
	TListView *TagsListView3;
	TListView *TagsListView4;
	TStaticText *StaticText9;
	TStaticText *StaticText10;
	TListView *TagListFixS3;
	TListView *TagListFixS4;
	TStaticText *StaticText11;
	TStaticText *StaticText12;
	TListView *TagListFixR3;
	TListView *TagListFixR4;
	TPageControl *pgc;
	TTabSheet *ts_root;
	TTabSheet *ts_tag;
	TPanel *status_pnl;
	TButton *btn_start;
	TButton *btn_stop;
	TStaticText *StaticText13;
	TComboBox *SBoxTags;
	TButton *SBtnPrevP;
	TButton *SBtnNextP;
	TButton *SBtnFix;
	TButton *SBtnPrev;
	TButton *SBtnReport;
	TButton *SBtnReset;
	TButton *SBtnZero;
	TButton *SBtnManual;
	TStaticText *txt_ch1;
	TComboBox *cbb_ch1;
	TStaticText *txt_ch2;
	TComboBox *cbb_ch2;
	TStaticText *txt_ch3;
	TComboBox *cbb_ch3;
	TComboBox *cbb_ch4;
	TStaticText *txe_cp4;
	TButton *btn_ok;
	TButton *btn_back;
	void __fastcall DataUpdateTimerTimer(TObject *Sender);
```

```
        void __fastcall SBoxTagsDropDown(TObject *Sender);
        void __fastcall TimeTimerTimer(TObject *Sender);
        void __fastcall SBoxTagsChange(TObject *Sender);
        void __fastcall SBtnPrevPClick(TObject *Sender);
        void __fastcall SBtnNextPClick(TObject *Sender);
        void __fastcall SBtnManualClick(TObject *Sender);
        void __fastcall btn_backClick(TObject *Sender);
        void __fastcall btn_okClick(TObject *Sender);
        void __fastcall SBtnFixClick(TObject *Sender);
        void __fastcall SBtnResetClick(TObject *Sender);
        void __fastcall SBtnPrevClick(TObject *Sender);
        void __fastcall SBtnZeroClick(TObject *Sender);
        void __fastcall btn_startClick(TObject *Sender);
        void __fastcall btn_stopClick(TObject *Sender);
        void __fastcall SBtnReportClick(TObject *Sender);
private:
        //переменная для сохранения ссылки на Recorder
        TComInterface<IRecorder> FRecorder;

        //Тип для описания вектора тегов
        typedef vector<TComInterface<ITag> > tagslist;
        //переменная вектор тегов; перечень тегов,...
        tagslist FTags; //для которых данные отображаются

        //Подготока окна (таблиц) для отображения данных.
        //Устанавливается необходимое кол-во строк, отображаются
        //имена тегов...
        void __fastcall PrepareTagsView(void);

        //Обновление данных. Получение данных у списка тегов
        //и отображение данных в табличном виде...
        void __fastcall DataUpdate(void);
        void __fastcall updateTag(void);
        vector<TListView*> TLV;
        vector<TListView*> TLV_S;
        vector<TListView*> TLV_R;
    sVer fVer;
public:
        //Переопределенный конструктор, предназначен для того,
        //чтобы запомнить в форме ссылку на Recorder.
        __fastcall TVerForm(IRecorder* pRecorder);
        //Метод для обработки начала изменения настройки.
        bool __fastcall BeginConfigure(void);
        //Метод для обработки завершения изменения настройки.
        bool __fastcall EndConfigure(void);
};
```

VerFormUnit.cpp

```
__fastcall TVerForm::TVerForm(IRecorder* pRecorder)
            : TForm( (Classes::TComponent*)0)
{
        FRecorder = pRecorder;
        vTag tT(pRecorder);
        for(int i(1);i<=4;i++){
                for(int i1(0); i1 < ComponentCount; i1++) {
                        if(Components[i1]->Name==("TagsListView"+IntToStr(i)))
TLV.push_back(dynamic_cast<TListView*>(Components[i1]));else
                        if(Components[i1]->Name==("TagListFixS"+IntToStr(i)))
TLV_S.push_back(dynamic_cast<TListView*>(Components[i1]));else
                        if(Components[i1]->Name==("TagListFixR"+IntToStr(i)))
TLV_R.push_back(dynamic_cast<TListView*>(Components[i1]));
                }
        }
        for(int i(0);i<4; i++) fVer.mTag[i]=tT;
}
//---------------------------------------------------------------------------
//Метод для обработки начала изменения настройки.
bool __fastcall TVerForm::BeginConfigure(void)
{
        TimeTimer->Enabled = false;
```

```
      DataUpdateTimer->Enabled=false;
      status_pnl->Caption="Конфигурирование";
    return true;
}
//---------------------------------------------------------------------------
//Метод для обработки завершения изменения настройки.
bool __fastcall TVerForm::EndConfigure(void)
{
      updateTag();
    //Подготовка таблиц для отображениятегов
      PrepareTagsView();

    return true;
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::DataUpdateTimerTimer(TObject *Sender)
{
      DataUpdate();
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::DataUpdate(void){
      int count1=fVer.iteriteration[0];
      int count2=fVer.iteriteration[1];
      if(count1<10){
            for(int i(0);i<4;i++){
                  if(count1<=4){
                        TLV_S[i]->Items->Item[count2]->SubItems-
>Strings[count1]=TLV[i]->Items->Item[0]->SubItems[0].Text;
                  }else{
                        TLV_R[i]->Items->Item[count2]->SubItems-
>Strings[count1-5]=TLV[i]->Items->Item[0]->SubItems[0].Text;
                  }
            }
            if((count2+1)>4){
                  DataUpdateTimer->Enabled=false;
                  status_pnl->Caption="Просмотр";
                  if((count1+1)>=10)      fVer.iteriteration[0]=10;      else
fVer.iteriteration[0]++;
                  fVer.iteriteration[1]=0;
            }else fVer.iteriteration[1]++;
      }
}
//---------------------------------------------------------------------------
//Метод подготовки формы после изменения списка тегов.
//Подготока окна (таблицы) для отображения данных.
//Устанавливается необходимое кол-во строк, отображаются
//имена тегов...
void __fastcall TVerForm::PrepareTagsView(void)
{
    DataUpdateTimer->Enabled=false;
      for(int i(0);i<4;i++){
            TLV[i]->Clear();
            TLV_S[i]->Clear();
            TLV_R[i]->Clear();
      }

      for(int i1(0);i1<4;i1++){
            TLV[i1]->Items->Add();
            TLV[i1]->Items->Item[0]->Caption=fVer.mTag[i1].getName().c_str();
            TLV[i1]->Items->Item[0]->SubItems->Add("");
            for(int i2(0);i2<5;i2++){
                  TLV_S[i1]->Items->Add();
                  TLV_S[i1]->Items->Item[i2]->Caption=IntToStr(i2+1);
                  for(int   s(0);s<5;s++)TLV_S[i1]->Items->Item[i2]->SubItems-
>Add("");
                  TLV_R[i1]->Items->Add();
                  TLV_R[i1]->Items->Item[i2]->Caption=IntToStr(i2+1);
                  for(int   s(0);s<5;s++)TLV_R[i1]->Items->Item[i2]->SubItems-
>Add("");
            }
```

```
	}
	fVer.iteriteration[0]=0;
	fVer.iteriteration[1]=0;
	status_pnl->Caption="Ожидание";
}
//---------------------------------------------------------------------------
//Заполнение списка тегов
void __fastcall TVerForm::SBoxTagsDropDown(TObject *Sender)
{
	updateTag();
	TComboBox* cbb_temp = dynamic_cast<TComboBox*>(Sender);
	int index=-1;
	if(cbb_temp->Text!="") index=cbb_temp->ItemIndex;
	cbb_temp->Items->Clear();
	tagslist::iterator i = FTags.begin();
	for (; i != FTags.end(); i++){
		cbb_temp->Items->Add((*i)->GetName());
	}
	if(index>=0&index<=(FTags.size()-1)) cbb_temp->ItemIndex=index;

}
//---------------------------------------------------------------------------
void __fastcall TVerForm::TimeTimerTimer(TObject *Sender)
{
	for(int i(0); i<4; i++){
		TLV[i]->Items->Item[0]->SubItems->Strings[0]                    =
fVer.mTag[i].getValue();
	}
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBoxTagsChange(TObject *Sender)
{
	int index = SBoxTags->ItemIndex;
	for(int i(0);i<4;i++){
		if((index+i)<=(FTags.size()-1))fVer.mTag[i].setTag(FTags[SBoxTags-
>ItemIndex+i]);
	}
	PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnPrevPClick(TObject *Sender)
{
	int index = SBoxTags->ItemIndex;
	if(index-4>=0){  SBoxTags->ItemIndex=(index-4);  index-=4;}else{  SBoxTags-
>ItemIndex=0; index=0;}
	for(int i(0);i<4;i++){
		if((index+i)<=(FTags.size()-1))fVer.mTag[i].setTag(FTags[SBoxTags-
>ItemIndex+i]);
	}
	PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnNextPClick(TObject *Sender)
{
	int index = SBoxTags->ItemIndex;
	if(index+4<=(FTags.size()-4)){              SBoxTags->ItemIndex=(index+4);
index+=4;}else{ SBoxTags->ItemIndex=(FTags.size()-4); index=(FTags.size()-4);}
	for(int i(0);i<4;i++){
		if((index+i)<=(FTags.size()-1))fVer.mTag[i].setTag(FTags[SBoxTags-
>ItemIndex+i]);
	}
	PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnManualClick(TObject *Sender)
{
	pgc->ActivePageIndex=1;
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::btn_backClick(TObject *Sender)
```

```
{
	pgc->ActivePageIndex=0;
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::btn_okClick(TObject *Sender)
{
	if((cbb_ch1->Text!="")&(cbb_ch2->Text!="")&(cbb_ch3->Text!="")&(cbb_ch4->Text!="")){
		fVer.mTag[0].setTag(FTags[cbb_ch1->ItemIndex]);
		fVer.mTag[1].setTag(FTags[cbb_ch2->ItemIndex]);
		fVer.mTag[2].setTag(FTags[cbb_ch3->ItemIndex]);
		fVer.mTag[3].setTag(FTags[cbb_ch4->ItemIndex]);
		SBoxTags->ItemIndex=cbb_ch1->ItemIndex;
		pgc->ActivePageIndex=0;
		PrepareTagsView();
	}else ShowMessage("Выберете все 4 канала!");
}
//---------------------------------------------------------------------------

void __fastcall TVerForm::SBtnFixClick(TObject *Sender)
{
	if(fVer.mTag[0].checkTag()||fVer.mTag[1].checkTag()||fVer.mTag[2].checkTag()||fVer.mTag[3].checkTag()){
		if(FRecorder->CheckState(RS_VIEW)||FRecorder->CheckState(RS_REC)){
			DataUpdateTimer->Enabled=true;
			status_pnl->Caption="Фиксация данных";
		}else ShowMessage("Recorder должен находится в режиме просмотра!");
	}else ShowMessage("Выбраны не все каналы!");
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnResetClick(TObject *Sender)
{
	if(MessageDlg(L"Сбросить зафиксированные данные?", mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0) == mrYes)
	PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnPrevClick(TObject *Sender)
{
    DataUpdateTimer->Enabled=false;
	if(fVer.iteriteration[0]-1<=0) PrepareTagsView(); else{
		fVer.iteriteration[0]--;
		fVer.iteriteration[1]=0;
		int count1=fVer.iteriteration[0];
		for(int i1(0);i1<4;i1++){
			for(int i2(0);i2<5;i2++){
				if(fVer.iteriteration[0]<=4){
					TLV_S[i1]->Items->Item[i2]->SubItems->Strings[count1]="";
				}else{
					TLV_R[i1]->Items->Item[i2]->SubItems->Strings[count1-5]="";
				}
			}
		}
	}
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::SBtnZeroClick(TObject *Sender)
{

	if(FRecorder->CheckState(RS_STOP)){
		for(int i(0);i<4;i++) fVer.mTag[i].setZero();
	}else ShowMessage("Recorder должен быть остановлен!");
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::btn_startClick(TObject *Sender)
{
	if(FRecorder->CheckState(RS_STOP)&!(FRecorder->CheckState(RS_CONFIGMODE))){
```

```
            FRecorder->Notify(RCN_VIEW,0);
            TimeTimer->Enabled = true;
            status_pnl->Caption="Просмотр";
      }else
      if(FRecorder->CheckState(RS_VIEW)||FRecorder->CheckState(RS_REC)){
       TimeTimer->Enabled = true;
      }else ShowMessage("Recorder занят");
}
//---------------------------------------------------------------------------
void __fastcall TVerForm::btn_stopClick(TObject *Sender)
{
      TimeTimer->Enabled = false;
      DataUpdateTimer->Enabled=false;
      if(FRecorder->CheckState(RS_VIEW)||FRecorder->CheckState(RS_REC)){
            FRecorder->Notify(RCN_STOP,0);
            status_pnl->Caption="Ожидание";
      }
}
//---------------------------------------------------------------------------

void __fastcall TVerForm::SBtnReportClick(TObject *Sender)
{
      WideString              fName       =       ExtractFilePath(Application-
>ExeName)+"plugins\\plgVer.xls";
      Variant MyExcel = CreateOleObject("EXCEL.Application");
      Variant Cells;
      if(fVer.iteriteration[0]==10){
            status_pnl->Caption="Генерация отчёта";
            for(int i(0);i<4;i++){
                  Variant MyExcel = CreateOleObject("EXCEL.Application");
                  Variant                                 Book                        =
MyExcel.OlePropertyGet("WorkBooks").OlePropertyGet("Open", fName);
                  MyExcel.OlePropertySet("Visible",false);
                  Variant Sheet = Book.OlePropertyGet("Worksheets", 1);
                  Cells = Sheet.OlePropertyGet("Cells",7,3);
                  Cells.OlePropertySet("Value",WideString(TLV[i]->Items-
>Item[0]->Caption).c_bstr());
                  Cells = Sheet.OlePropertyGet("Cells",7,7);
                  Cells.OlePropertySet("Value",WideString(TLV[i]->Items-
>Item[0]->Caption).c_bstr());
                  for(int Row(21);Row<32;Row++){
                        if(Row!=26){
                              for(int Column(3);Column<8;Column++){
                                    if(Row<26){
                                          String       tmpS=TLV_S[i]->Items-
>Item[Row-21]->SubItems->Strings[Column-3];
                                          String _tmpS;
                                          for(int t(1);t<tmpS.Length();t++){
                                                String tmps=tmpS[t];
                                                if(tmps=="-
"||tmps==","||tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||
tmps=="7"||tmps=="8"||tmps=="9"||tmps=="0")
                                                      _tmpS+=tmps;
                                          }

      Sheet.OlePropertyGet("Cells",Row,Column).OlePropertySet("Value",_tmpS.c_s
tr());
                                    }else{
                                          String       tmpS=TLV_R[i]->Items-
>Item[Row-27]->SubItems->Strings[Column-3];
                                          String _tmpS;
                                          for(int t(1);t<tmpS.Length();t++){
                                                String tmps=tmpS[t];
                                                if(tmps=="-
"||tmps==","||tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||
tmps=="7"||tmps=="8"||tmps=="9"||tmps=="0")
                                                      _tmpS+=tmps;
                                          }
```

```
            Sheet.OlePropertyGet("Cells",Row,(10-
      Column)).OlePropertySet("Value",_tmpS.c_str());
                                          }

                                    }
                              }
                        }
                        MyExcel.OlePropertySet("Visible",true);
                  }
                  status_pnl->Caption="Ожидание";
            }else ShowMessage("Зафиксированы еще не все данные!");
      }
      //---------------------------------------------------------------------------
      void __fastcall TVerForm::updateTag(void){
      //Очистка внутреннего списка тегов
          FTags.clear();

          //Получение (обновленного) списка тегов
          int count = FRecorder->GetTagsCount();
          for (int i = 0; i < count; i++)
          {
              //временная переменная...
              TComInterface<ITag> pp;
              //...для получения ссылки на тег
                    pp.Bind(FRecorder->GetTagByIndex(i),1);
              //...и добавления ссылки в список
                    FTags.push_back ( pp);
            }
      }
//---------------------------------------------------------------------------
```

VerTag.h

```
      class vTag{
      public:
            vTag(){ crtTag=0;};
            vTag(IRecorder*        pRecorder){TRecorder=pRecorder;        crtTag=0;};
      //Переопределенный конструктор, предназначен для того,

                        //чтобы запомнить в форме ссылку на Recorder.
            void setTag(ITag* pTag); //Привязка тега
            double getValue(); //Получить текущее значение тега
            string getName(); //Получить имя тега
            bool checkTag(); //Проверка существования тега
            bool setZero(); //Балансировка нуля
            ~vTag(){};
      private:
            string name; //Имя тега
            typedef vector<TComInterface<ITag> > tagslist; //Переменная вектора тегов
            tagslist FTags; //Перечень тегов
            TComInterface<IRecorder> TRecorder; //Переменная для сохранения ссылки на
      Recorder
            TComInterface<ITag> TTag; //Переменная для сохранения ссылки на тег
            bool crtTag; //Индикатор привязвн ли тег
            void updateTag();  //Обновление списка тегов
};
```

VerTag.cpp

```
      void vTag::updateTag(){
            FTags.clear();  //Очистка внутреннего списка тегов
            int count = TRecorder->GetTagsCount(); //Получение (обновленного) списка
      тегов
            for (int i = 0; i < count; i++){
                  //временная переменная...
                  TComInterface<ITag> pp;
                  //...для получения ссылки на тег
                  pp.Bind(TRecorder->GetTagByIndex(i),1);
                  //...и добавления ссылки в список
                  FTags.push_back (pp);
            }
      }
```

```
//---------------------------------------------------------------------------
void vTag::setTag(ITag* pTag){
	TTag=pTag;
	name=TTag->GetName();
	crtTag=1;
}
//---------------------------------------------------------------------------

string vTag::getName(){
	if(crtTag) return name; else return "Канал не выбран!";
}
//---------------------------------------------------------------------------
double vTag::getValue(){
	if(crtTag){
		double result;
		updateTag();
		//получить по ссылке на тег, ссылку на
		//интерфейс блочного доступа
		TComInterface<IBlockAccess> pba;
		TTag->QueryInterface(                          IID_IBlockAccess,
reinterpret_cast<void**>(&pba));
		//блокировать буфер с измеренными данными, это обязательное
		//действие; так как буфер постоянно пополняется измеренными
		//данными - необходимо остановить изменение вектора на
		//время отображения данных.
		pba->LockVector();
		try{
			//получить размер блока
			int bl_size = pba->GetBlocksSize();
			//получить кол-во блоков
			int bl_count = pba->GetBlocksCount();
			//проверка
			if ((bl_count != 0) && (bl_size != 0)){
				//выделяем память для чтения блока....
				double * pdata = reinterpret_cast<double *>(alloca(
sizeof(double) * bl_size));
				//чтение последнего блока измеренных данных
				pba->GetVectorR8( pdata, bl_count - 1, bl_size, true);
				//вычисление среднего арифметического
				double aod= 0;
				for (int i= 0; i < bl_size; i++) aod = aod + pdata[i];
				aod= aod / bl_size;
				result = aod;
			}
		}
		__finally{
			//разблокировать вектор
			pba->UnlockVector();
		}
		return result;
	}else return 0;
}
//---------------------------------------------------------------------------
bool vTag::checkTag(){
	if(crtTag) return true;
	return false;
}
//---------------------------------------------------------------------------
bool vTag::setZero(){
	if(crtTag){
		TTag->Notify(TN_ZEROBALANCE,0);
		return true;
	}return false;
}
 //---------------------------------------------------------------------------
```

ПРИЛОЖЕНИЕ Б.

Исходный код программного модуля и интерфейса управления виртуальными тегами

vTag.h

```
struct program_step{ //Структура шагов (для тега с заданной программой)
	double step_value; //Значение шага
	int step_speed, step_time; //Скорость изменения, время удержания значения
};
//---------------------------------------------------------------------------
class vTag{
public:
	vTag(IRecorder*		pRecorder){TRecorder=pRecorder;		crtTag=0;};
//Переопределенный конструктор, предназначен для того,

			//чтобы запомнить в форме ссылку на Recorder.
	bool CreateTag(string); //Создание виртуального тега с заданным именем
	void setConstTag(double); //Настройка параметров для тега с фикс. значением
	void setVariableTag(double,double,double); //Настройка параметров для тега
с перемен. значением
	void setSmoothlyTag(double,double,double); //Настройка параметров для тега
с плавн. измен. значением
	void setProgramTag(vector<program_step>);  //Настройка параметров для тега
с измен. по программе
	void NextValue();  //Следующее значение (для изменяемых тегов)
	double getValue(); //Получить текущее значение тега
	string getName();  //Получить имя тега
	bool deleteTag();  //Удаление виртуального тега
	string getType();  //Получение типа тега
	void PrepareTagsView(); //Подготовка просмотра
	bool getVarFlg(){return var_flg;}; //Достиг ли тег заданной полки
	bool checkTag(); //Проверка существования тега
	~vTag(){};
private:
	string name; //Имя тега
	double value, initV, finV, step; //Заданное значение тега, начальное и
конечное значения, шаг(в сек.)
	int type, i_vector, timeV; //Тип тега (0-потоянный,1-переменный, 2-плавный,
3-по программе), итератор перехода по программе тега, время на шаге
	bool end_flg, var_flg; //Флаг завершения программы (для тега с заданной
программой), флаг переменного и программного тегов
	vector<program_step> pStep; //Вектор шагов (для тега с заданной программой)
	typedef vector<TComInterface<ITag> > tagslist;//Переменная вектора тегов
	tagslist FTags; //Перечень тегов
	TComInterface<IRecorder> TRecorder; //Переменная для сохранения ссылки на
Recorder
	TComInterface<ITag> TTag; //Переменная для сохранения ссылки на тег
	bool crtTag; //Индикатор создан ли виртуальный тег
	void cleanParam(); //Сброс параметров тега
	bool checkState(); //Проверка состояния рекордера
	void updateTag();  //Обновление списка тегов
};
```

vTag.cpp

```
void vTag::updateTag(){
	FTags.clear();  //Очистка внутреннего списка тегов
	int count = TRecorder->GetTagsCount(); //Получение (обновленного) списка
тегов
	for (int i = 0; i < count; i++){
		//временная переменная...
		TComInterface<ITag> pp;
		//...для получения ссылки на тег
		pp.Bind(TRecorder->GetTagByIndex(i),1);
		//...и добавления ссылки в список
		FTags.push_back (pp);
	}
}
```

```
//---------------------------------------------------------------------------
bool vTag::CreateTag(string name_tag){
	name=name_tag;
	if(checkState()&&!crtTag&&!checkTag()){
		const char *tempName = new char[name_tag.length()];
		tempName = name_tag.c_str();
		TRecorder->EnterConfigMode(0,0);  //Перевести Recorder в режим
конфигурирования
		TTag = TRecorder->CreateTag(tempName,LS_VIRTUAL,0); //Создание тега
с заданным именем
		TTag->SetProperty(TAGPROP_TYPE,  Variant(TTAG_SCALAR));  //Задание
тегу тип (скалярный)
		TTag->SetProperty(TAGPROP_DATATYPE, Variant((double)0)); //Задания
тегу тип данных (double)
		TRecorder->LeaveConfigMode(0,0);  //Вывести Recorder из режима
конфигурирования
		type=0;
		crtTag=true;
		return true;
		//fale-Ошибка! Тег с таким именем уже существует!
		//true-Успешное создание тега
	}
	return false;
}
//---------------------------------------------------------------------------
string vTag::getType(){
	switch (type) {
	case 0:{
		return "Постоянный";
		break;
	}
	case 1:{
		return "Переменный";
		break;
	}
	case 2:{
		return "Плавный";
		break;
	}
	case 3:{
		return "По программе";
		break;
	}
	default:{
		return "Ошибка!";
		break;
	}
	}
}
//---------------------------------------------------------------------------
string vTag::getName(){
	if(crtTag) return name; else return "NULL";
}
//---------------------------------------------------------------------------
double vTag::getValue(){
	if(crtTag){
		double result;
		updateTag();
		//получить по ссылке на тег, ссылку на
		//интерфейс блочного доступа
		TComInterface<IBlockAccess> pba;
		TTag->QueryInterface(                          IID_IBlockAccess,
reinterpret_cast<void**>(&pba));
		//блокировать буфер с измеренными данными, это обязательное
		//действие; так как буфер постоянно пополняется измеренными
		//данными - необходимо остановить изменение вектора на
		//время отображения данных.
		pba->LockVector();
		try{
			//получить размер блока
```

```
                  int bl_size = pba->GetBlocksSize();
                  //получить кол-во блоков
                  int bl_count = pba->GetBlocksCount();
                  //проверка
                  if ((bl_count != 0) && (bl_size != 0)){
                        //выделяем память для чтения блока....
                        double * pdata = reinterpret_cast<double *>(alloca(
sizeof(double) * bl_size));
                        //чтение последнего блока измеренных данных
                        pba->GetVectorR8( pdata, bl_count - 1, bl_size, true);
                        //вычисление среднего арифметического
                        double aod= 0;
                        for (int i= 0; i < bl_size; i++) aod = aod + pdata[i];
                        aod= aod / bl_size;
                        result = aod;
                  }
            }
            __finally{
                  //разблокировать вектор
                  pba->UnlockVector();
            }
            return result;
      }else return 0;
}
//---------------------------------------------------------------------------
void vTag::cleanParam(){
      pStep.clear();
      type=-1;
      value=0;
      initV=0;
      finV=0;
      step=0;
      end_flg=false;
      var_flg=true;
      i_vector=0;
      step=0;
      timeV=0;
}
//---------------------------------------------------------------------------
void vTag::setConstTag(double _value){
      cleanParam();
      value=_value;
      type=0;
}
//---------------------------------------------------------------------------
void vTag::setVariableTag(double _init,double _fin,double _step){
      cleanParam();
      if(_step==0) _step=0.1;
      initV=_init;
      finV=_fin;
      step=_step;
      value=(_init+_fin)/2;
      type=1;
}
//---------------------------------------------------------------------------
void vTag::setSmoothlyTag(double _init,double _fin,double _step){
      cleanParam();
      if(_step==0) _step=0.1;
      if(_init>_fin&_step>0) _step*=(-1); else
      if(_init<_fin&_step<0) _step*=(-1);
      initV=_init;
      finV=_fin;
      step=_step;
      value=_init;
      type=2;
}
//---------------------------------------------------------------------------
void vTag::setProgramTag(vector<program_step> _pStep){
      cleanParam();
      if(!_pStep.empty()){
```

```
		for(int i(0);i<_pStep.size();i++){
			if(_pStep[i].step_speed==0) _pStep[i].step_speed=1; else
			if(_pStep[i].step_speed<0)_pStep[i].step_speed*=(-1);
		}
		pStep=_pStep;
		if(pStep[0].step_time==0)timeV=-1; //Чтобы не перескакивал значение
		value=pStep[0].step_value;
		type=3;
	}
}
//---------------------------------------------------------------------------
void vTag::NextValue(){
	if(crtTag&&checkTag()){
		switch (type) {
			case 0:{
				TTag->PushValue(value,0);
				break;
			}
			case 1:{
				TTag->PushValue(value,0);
				if(initV==finV&&step) end_flg=true; else{
					if(var_flg){
						if(initV>finV&&step>0) step*=(-1); else
						if(initV<finV&&step<0) step*=(-1);
						if(step>0){
							if(value+step>=finV){value=finV;
var_flg=false;} else value+=step;
						}else{
							if(value+step<=finV){value=finV;
var_flg=false;} else value+=step;}
					}else{
						if(finV>initV&&step>0) step*=(-1); else
						if(finV<initV&&step<0) step*=(-1);
						if(step>0){
							if(value+step>=initV){value=initV;
var_flg=true;} else value+=step;
						}else{
							if(value+step<=initV){value=initV;
var_flg=true;} else value+=step;}
						}
				}
				break;
			}
			case 2:{
				TTag->PushValue(value,0);
				if(initV==finV) end_flg=true; else{
					if(step>0){
						if(value+step>=finV){value=finV;
end_flg=true;} else value+=step;
					}else{
						if(value+step<=finV){value=finV;
end_flg=true;} else value+=step;
					}
				}
				break;
			}
			case 3:{
				if(!end_flg){
					if(var_flg){

	if(timeV<pStep[i_vector].step_time)timeV++;else
							if(i_vector+1<pStep.size()){
								timeV=0;
								i_vector++;
								initV=value;

	finV=pStep[i_vector].step_value;
							double diff;

	if(initV>=0&&finV>=0){diff=abs(initV-finV);}else
```

```
	if(initV<0&&finV>=0){diff=abs(initV-finV);}else

	if(initV>=0&&finV<0){diff=abs(finV-initV);}else{diff=abs(initV-finV);}

	step=diff/pStep[i_vector].step_speed;
                                                    if(initV>finV) step*=(-1);
                                                    if(step>0){

	if(value+step>=finV){value=finV; var_flg=true;} else value+=step;
                                                    }else{

	if(value+step<=finV){value=finV; var_flg=true;} else value+=step;
                                                    }
                                                    var_flg=false;
                                             }else end_flg=true;
                                  }else{
                                        if(step>0){
                                        if(value+step>=finV){value=finV;
var_flg=true;} else value+=step;
                                        }else{
                                        if(value+step<=finV){value=finV;
var_flg=true;} else value+=step;
                                        }
                                  }
                            }
                            TTag->PushValue(value,0);
                            break;
                      }
                      default:{
                            break;
                      }
                }

	}
}
//---------------------------------------------------------------------------
bool vTag::checkTag(){
	updateTag();
	tagslist::iterator i = FTags.begin();
	for (; i != FTags.end(); i ++)
		if((*i)->GetName()==name) return true;
	return false;
}
//---------------------------------------------------------------------------
void vTag::PrepareTagsView(){
	if(crtTag&&checkTag()){
		switch (type) {
		case 0:{
			setConstTag(value);
			break;
		}
		case 1:{
			setVariableTag(initV,finV,step);
			break;
		}
		case 2:{
			setSmoothlyTag(initV,finV,step);
			break;
		}
		case 3:{
			setProgramTag(pStep);
			break;
		}
		default:{
			break;
		}
		}
	}
}
```

```
//---------------------------------------------------------------------------
 bool vTag::deleteTag(){
      if(checkState()&&crtTag&&checkTag()){
            TRecorder->EnterConfigMode(0,0);  //Перевести Recorder в режим
конфигурирования
            TRecorder->CloseTag(TTag); //Удаление тега
            TRecorder->LeaveConfigMode(0,0);  //Вывести Recorder из режима
конфигурирования
            crtTag=false;
            return true;
      }
      return false;
 }
//---------------------------------------------------------------------------
 bool vTag::checkState(){
      int state = TRecorder->GetState(RS_FULLMASK);
      if(state==2049||state==2433||state==6145||state==1||state==4097||state==6
529||state==6401) return true;
      return false;
 }
//---------------------------------------------------------------------------
```

TGenUnit.h

```
struct vTagGroup{
      String name;
      TTabSheet *TTS; //Закреплённая за группой вкладка
      TListView *TLV; //Закреплённая за группой таблица
      bool sinch;  //Флаг синхронности тегов (для програмных)
      bool var_flg; //Достигли ли все теги заданного значения (для программных)
      vector<vTag> FvTag;
};
using std::vector;
using std::string;

//Эта форма реализует два различных способа получения данных из тегов.
//Првый способ - это получение оценки. Второй - получение вектора и
//вычисление среднего. Для каждого из методов описан свой собственный
//метод получения данных. Полученные данные всегда преобразуются в строку
//для отображения. Какой из методов будет испеользоваться завсисит от
//того определен ли идентификатор DATATHROUGHVECTOR. Если этот идентификатор
//определен, то используется метод получения вектора данных.
//Выбор используемого метода осуществляется директивами прекомпиляции
//#ifdef, #ifndef, #else, #endif...
#define DATATHROUGHVECTOR

//---------------------------------------------------------------------------
//Форма для тестового plufg-in`a, для отображения измеренных данных.
//Форма производит периодически получение данных у списка тегов и
//отображение данных в табличном виде...
class TGenForm : public TForm
{
__published: // IDE-managed Components
      TButton *btn_start;
      TButton *btn_stop;
      TTimer *tmr1;
      TOpenDialog *dlgOpen1;
      TPageControl *pgc1;
      TTabSheet *ts1;
      TListView *lv1;
      TPanel *RootPNL;
      TButton *btn_crtG;
      TPageControl *pgc_cfg;
      TTabSheet *root_pg;
      TTabSheet *crtG_pg;
      TEdit *edtGroup;
      TBitBtn *btnCrtG;
      TBitBtn *btnBackG;
      TButton *btn_crtT;
      TTabSheet *crtT_pg;
```

```
TComboBox *cbb_ct;
TComboBox *cbb_tT;
TBitBtn *btn_TNext;
TBitBtn *btn_bkT;
TTabSheet *ts_crt_prog;
TBitBtn *btn_open;
TBitBtn *btn_back;
TTabSheet *crt_const;
TBitBtn *btn_backC;
TBitBtn *btn_crtC;
TEdit *edt_NC;
TEdit *edt_VT;
TTabSheet *crt_var;
TCheckBox *chk_V;
TEdit *edt_NV;
TEdit *edt_minV;
TBitBtn *btn_crtVT;
TBitBtn *btn_backV;
TEdit *edt_maxV;
TEdit *edt_step;
TPanel *Cabinet_1;
TPanel *Cabinet_2;
TPanel *Cabinet_3;
TButton *btn_delG;
TTabSheet *delG_pg;
TComboBox *cbb_G;
TBitBtn *btn_bkDG;
TBitBtn *btn_nDG;
TButton *btn_delT;
TTabSheet *delT_pg;
TBitBtn *btn_NDT;
TBitBtn *btn_BDK;
TComboBox *cbb_GDT;
TComboBox *cbb_DT;
TCheckBox *chk_sinch;
TPanel *status_pnl;
TButton *btn_chgT;
TCheckBox *chk_cng;
TComboBox *cbb_tN;
TStaticText *txt_NT;
TStaticText *txt_VT;
TStaticText *txt_GN;
TStaticText *txt_CG;
TStaticText *txt_CGDT;
TStaticText *txt_CTDT;
TStaticText *txt_CGC;
TStaticText *txt_CTTC;
TStaticText *txt_CTCT;
TTabSheet *com_p;
TComboBox *cbb_com;
TStaticText *txt_com;
TBitBtn *btn_cb;
TBitBtn *btn_oc;
TTimer *tmr2;
void __fastcall btn_startClick(TObject *Sender);
void __fastcall btn_stopClick(TObject *Sender);
void __fastcall tmr1Timer(TObject *Sender);
void __fastcall btnCrtGClick(TObject *Sender);
void __fastcall btnBackGClick(TObject *Sender);
void __fastcall btn_crtGClick(TObject *Sender);
void __fastcall btn_crtTClick(TObject *Sender);
void __fastcall btn_bkTClick(TObject *Sender);
void __fastcall btn_backClick(TObject *Sender);
void __fastcall btn_openClick(TObject *Sender);
void __fastcall cbb_tTChange(TObject *Sender);
void __fastcall btn_TNextClick(TObject *Sender);
void __fastcall btn_crtCClick(TObject *Sender);
void __fastcall edt_VTChange(TObject *Sender);
void __fastcall btn_crtVTClick(TObject *Sender);
void __fastcall edt_minVChange(TObject *Sender);
```

```
        void __fastcall edt_stepChange(TObject *Sender);
        void __fastcall edt_maxVChange(TObject *Sender);
        void __fastcall btn_delGClick(TObject *Sender);
        void __fastcall btn_nDGClick(TObject *Sender);
        void __fastcall btn_delTClick(TObject *Sender);
        void __fastcall cbb_GDTChange(TObject *Sender);
        void __fastcall btn_NDTClick(TObject *Sender);
        void __fastcall chk_sinchClick(TObject *Sender);
        void __fastcall btn_chgTClick(TObject *Sender);
        void __fastcall cbb_ctChange(TObject *Sender);
        void __fastcall status_pnlDblClick(TObject *Sender);
        void __fastcall cbb_comDropDown(TObject *Sender);
        void __fastcall btn_cbClick(TObject *Sender);
        void __fastcall btn_ocClick(TObject *Sender);
        void __fastcall tmr2Timer(TObject *Sender);

private:
             bool __fastcall crtProgT(void);
             void __fastcall UpdateTagData(void);
        //переменная для сохранения ссылки на Recorder
             TComInterface<IRecorder> FRecorder;

        //Подготока окна (таблицы) для отображения данных.
        //Устанавливается необходимое кол-во строк, отображаются
        //имена тегов...
        void __fastcall PrepareTagsView(void);

        //Обновление данных. Получение данных у списка тегов
        //и отображение данных в табличном виде...
             void __fastcall DataUpdate(void);
             bool __fastcall CrtGroup(String);
             void __fastcall CrtGroup(vTagGroup);

public:
        //Переопределенный конструктор, предназначен для того,
        //чтобы запомнить в форме ссылку на Recorder.
             __fastcall TGenForm(IRecorder* pRecorder);
        //Метод для обработки начала изменения настройки.
        bool __fastcall BeginConfigure(void);
             //Метод для обработки завершения изменения настройки.
             bool __fastcall EndConfigure(void);
             bool __fastcall LockChenge(void); //Запрет создания/удаления тегов
             void __fastcall EndPLG(void);
             vector<vTagGroup> GvTag; //Вектор для групп векторов тегов
};
```

TGenUnit.cpp

```
__fastcall        TGenForm::TGenForm(IRecorder*        pRecorder):        TForm(
(Classes::TComponent*)0){
       FRecorder = pRecorder;
       pgc_cfg->ActivePageIndex=0;
}
//---------------------------------------------------------------------------
//Метод для обработки начала изменения настройки.
bool __fastcall TGenForm::BeginConfigure(void){
       tmr1->Enabled = false;
       tmr2->Enabled = false;
    return true;
}
//---------------------------------------------------------------------------
//Метод для обработки завершения изменения настройки.
bool __fastcall TGenForm::EndConfigure(void){
       //Подготовка таблицы для отображениятегов
       PrepareTagsView();
       pgc_cfg->ActivePageIndex=0;
       btn_crtT->Enabled=true;
       btn_crtT->Caption="Создать тег";
       btn_crtG->Enabled=true;
       btn_crtG->Caption="Создать группу";
       btn_delG->Enabled=true;
```

```
	btn_delG->Caption="Удалить группу";
	btn_delT->Enabled=true;
	btn_delT->Caption="Удалить тег";
	return true;
}
//---------------------------------------------------------------------------
//Метод обновления отображения измеренных данных.
//Обновление данных. Получение данных у списка тегов
//и отображение данных в табличном виде...
void __fastcall TGenForm::DataUpdate(void){
	//Проверка существования тега
	for(int i(0);i<GvTag.size();i++)
		for(int i1(0);i1<GvTag[i].FvTag.size();i1++)
			if(!GvTag[i].FvTag[i1].checkTag())
GvTag[i].FvTag.erase(GvTag[i].FvTag.begin()+i1);
	//Установка значения в таблицу
	int num(0);
	for(int i(0);i<GvTag.size();i++)
		for(int i1(0);i1<GvTag[i].FvTag.size();i1++){
		//проверка корректности таблицы
		if (lv1->Items->Count > i1){
			TListItem* pil = lv1->Items->Item[num];
			if (pil->SubItems->Count >= 1)
				pil->SubItems-
>Strings[0]=GvTag[i].FvTag[i1].getValue();
			if (pil->SubItems->Count >= 2)
				pil->SubItems-
>Strings[1]=GvTag[i].FvTag[i1].getType().c_str();
			num++;
			}
		if (GvTag[i].TLV->Items->Count > i1){
			TListItem* pil = GvTag[i].TLV->Items->Item[i1];
			if (pil->SubItems->Count >= 1)
				pil->SubItems-
>Strings[0]=GvTag[i].FvTag[i1].getValue();
			if (pil->SubItems->Count >= 2)
				pil->SubItems-
>Strings[1]=GvTag[i].FvTag[i1].getType().c_str();
			}
		}
}
//---------------------------------------------------------------------------
//Метод подготовки формы после изменения списка тегов.
//Подготока окна (таблицы) для отображения данных.
//Устанавливается необходимое кол-во строк, отображаются
//имена тегов...
void __fastcall TGenForm::PrepareTagsView(void)
{
	//Отчистка таблиц
	lv1->Items->Clear();
	for(int i(0);i<GvTag.size();i++)
		GvTag[i].TLV->Items->Clear();
	//Проверка существования тега
//	for(int i(0);i<GvTag.size();i++)
//		for(int i1(0);i1<GvTag[i].FvTag.size();i1++)
//			if(!GvTag[i].FvTag[i1].checkTag())
GvTag[i].FvTag.erase(GvTag[i].FvTag.begin()+i1);
	for(int i(0);i<GvTag.size();i++)
		for(int i1(0);i1<GvTag[i].FvTag.size();i1++){
				TListItem* pil = lv1->Items->Add();
				pil->Caption = GvTag[i].FvTag[i1].getName().c_str();
				pil->SubItems->Add(""); //Под значение тега
				pil->SubItems-
>Add(GvTag[i].FvTag[i1].getType().c_str()); //Под тип тега
				pil->SubItems->Add(GvTag[i].name); //Под группу тега

				pil=GvTag[i].TLV->Items->Add();
				pil->Caption = GvTag[i].FvTag[i1].getName().c_str();
				pil->SubItems->Add(""); //Под значение тега
```

```
                        pil->SubItems-
>Add(GvTag[i].FvTag[i1].getType().c_str()); //Под тип тега
                        pil->SubItems->Add(GvTag[i].name); //Под группу тега
                }
//      //Подготовка списка тегов к просмотру
        for(int i(0);i<GvTag.size();i++)
                for(int                         i1(0);i1<GvTag[i].FvTag.size();i1++)
GvTag[i].FvTag[i1].PrepareTagsView();
        btn_start->Caption="Старт";
        status_pnl->Color=clBtnFace;
        status_pnl->Caption="Остановлен";
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_startClick(TObject *Sender)
{
        if(btn_start->Caption=="Старт"){
                btn_start->Caption="Пауза";
                tmr1->Enabled = true;
                status_pnl->Color=clLime;
                status_pnl->Caption="Запущен";
                btn_crtT->Enabled=false;
                btn_crtT->Caption="Заблокировано";
                btn_crtG->Enabled=false;
                btn_crtG->Caption="Заблокировано";
                btn_delG->Enabled=false;
                btn_delG->Caption="Заблокировано";
                btn_delT->Enabled=false;
                btn_delT->Caption="Заблокировано";
        }else{
                btn_start->Caption="Старт";
                tmr1->Enabled = false;
                status_pnl->Color=clYellow;
                status_pnl->Caption="Пауза";
        }
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_stopClick(TObject *Sender)
{
        tmr1->Enabled = false;
        btn_crtT->Enabled=true;
        btn_crtT->Caption="Создать тег";
        btn_crtG->Enabled=true;
        btn_crtG->Caption="Создать группу";
        btn_delG->Enabled=true;
        btn_delG->Caption="Удалить группу";
        btn_delT->Enabled=true;
        btn_delT->Caption="Удалить тег";
        PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::tmr1Timer(TObject *Sender)
{
        UpdateTagData();
}
//---------------------------------------------------------------------------
bool __fastcall TGenForm::crtProgT(void){
        int Column=3;
        vTagGroup tmpTG; //Временная переменная группы
        Variant MyExcel = CreateOleObject("EXCEL.Application");
        WideString  fName = dlgOpen1->FileName;
        tmpTG.name  =  ExtractFileName(dlgOpen1->FileName);  //Имя  группы  по  имени
файла конфигурации
        Variant  Book  =  MyExcel.OlePropertyGet("WorkBooks").OlePropertyGet("Open",
fName);
        MyExcel.OlePropertySet("Visible",false);
        Variant Sheet = Book.OlePropertyGet("Worksheets", 1);
        String tmpName = Sheet.OlePropertyGet("Cells",1,Column);
        //Заполнение группы тегами
        while(tmpName!=""){
                program_step tmpPS; //Временная переменная шага
```

```
            vector<program_step> tmpVPS; //Временная переменная вектора шагов
            vTag tmpTag(FRecorder); //Временная переменная тега
            AnsiString _astr = tmpName;
            if(tmpTag.CreateTag(_astr.c_str())){
                  int Row=4;
                  String tmp_value=Sheet.OlePropertyGet("Cells",Row,Column);
                  String tmp_time=Sheet.OlePropertyGet("Cells",Row+1,Column);
                  String tmp_speed=Sheet.OlePropertyGet("Cells",Row+2,Column);
                  while(tmp_value!=""&&tmp_time!=""&&tmp_speed!=""){
                        Row+=3;
                        tmpPS.step_value = StrToFloat(tmp_value);
                        tmpPS.step_time = StrToInt(tmp_time);
                        tmpPS.step_speed = StrToInt(tmp_speed);
                        tmpVPS.push_back(tmpPS);
                        tmp_value=Sheet.OlePropertyGet("Cells",Row,Column);
                        tmp_time=Sheet.OlePropertyGet("Cells",Row+1,Column);
                        tmp_speed=Sheet.OlePropertyGet("Cells",Row+2,Column);
                  }
                  tmpTag.setProgramTag(tmpVPS);
                  tmpTG.FvTag.push_back(tmpTag);
         }
                  Column++;
                  tmpName = Sheet.OlePropertyGet("Cells",1,Column);
      }
      MyExcel.OleProcedure(WideString("Quit"));
      if(chk_sinch->Checked)tmpTG.sinch=true; else tmpTG.sinch=false;
      if(!tmpTG.FvTag.empty()) CrtGroup(tmpTG); else return false;
      PrepareTagsView();
      return true;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::UpdateTagData(void){
      for(int i(0);i<GvTag.size();i++)
            for(int i1(0);i1<GvTag[i].FvTag.size();i1++)
                  if(!GvTag[i].FvTag[i1].checkTag())
GvTag[i].FvTag.erase(GvTag[i].FvTag.begin()+i1);
      for(int i(0);i<GvTag.size();i++)
            for(int i1(0);i1<GvTag[i].FvTag.size();i1++){
                  if((GvTag[i].FvTag[i1].getType()=="По
программе")&&(GvTag[i].sinch==true)){
                        int endF(0);
                        //Дошли ли все теги до заданного значения
                        for(int i2(0);i2<GvTag[i].FvTag.size();i2++)
                              if(GvTag[i].FvTag[i2].getVarFlg())endF++;
                        if(endF==GvTag[i].FvTag.size())   GvTag[i].var_flg   =
true;
                        //Индикатор того, что все теги выдержали заданную паузу
                        if(endF==0) GvTag[i].var_flg = false;
                        if(GvTag[i].FvTag[i1].getVarFlg()){

      if(GvTag[i].var_flg)GvTag[i].FvTag[i1].NextValue();
                        }else GvTag[i].FvTag[i1].NextValue();
                  }else GvTag[i].FvTag[i1].NextValue();
            }
      //Обновление данных. Получение данных у списка тегов
      //и отображение данных в табличном виде...
      DataUpdate();
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::EndPLG(void){
      GvTag.clear();
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btnCrtGClick(TObject *Sender){
      if(edtGroup->Text=="") ShowMessage("Введите название группы!");
      else  if(CrtGroup(edtGroup->Text)){
        edtGroup->Text="";
            pgc_cfg->ActivePageIndex=0;
      }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TGenForm::btnBackGClick(TObject *Sender)
{
      edtGroup->Text="";
      pgc_cfg->ActivePageIndex=0;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_crtGClick(TObject *Sender)
{
      pgc_cfg->ActivePageIndex=1;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_crtTClick(TObject *Sender)
{
      if(cbb_tT->Items->Count<4) cbb_tT->Items->Add("По программе");
      cbb_ct->Visible=false;
    txt_CGC->Visible=false;
      cbb_tN->Visible=false;
      txt_CTCT->Visible=false;
      chk_cng->Checked=true;
      cbb_ct->Clear();
      cbb_tT->ItemIndex=-1;
      for(int i(0);i<GvTag.size();i++) cbb_ct->Items->Add(GvTag[i].name);
      pgc_cfg->ActivePageIndex=2;
}
//---------------------------------------------------------------------------
bool __fastcall TGenForm::CrtGroup(String NameGRP){
      if(NameGRP!=""){
            bool flg(true);
            for(int i(0);i<GvTag.size();i++) if(GvTag[i].name==NameGRP) flg =
false;
            if(flg){
                  vTagGroup tmpTG;
                  tmpTG.name=NameGRP;
                  std::auto_ptr<TMemoryStream> ms(new TMemoryStream);
                  lv1->Items->Clear();
                  //Копирование параметров
                  ms->WriteComponent(lv1);
                  ms->Position = 0;
                  TTabSheet *TabSheet = new TTabSheet(pgc1); //Создание новой
вкладки
                  TabSheet->Parent = pgc1; //Объект-родитель (на котором
создастся)
                  int num(2);
                  bool flg1(true);
                  do{
                        flg1=true;
                        String tN = "ts"+IntToStr(num);
                        for(int i(0);i<GvTag.size();i++)
                              if(GvTag[i].TTS->Name==tN){flg1=false; num++;}
                  }while(!flg1);
                  TabSheet->Name = "ts"+IntToStr(num); //Имя вкладки
                  TabSheet->Caption = tmpTG.name; //Отображаемое название
                  TabSheet->PageControl = pgc1;
                  TListView* pil = new TListView(TabSheet);
                  pil->Parent = TabSheet; //Объект-родитель (на котором
создастся)
                  ms->ReadComponent(pil); //Клонирование параметров
                  pil->Name="lv"+IntToStr(num);
                  tmpTG.TLV=pil;
                  tmpTG.TTS=TabSheet;
                  GvTag.push_back(tmpTG);
                  PrepareTagsView();
                  return true;
            }else ShowMessage("Группа с данным названием уже существует!");
      }
      return false;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::CrtGroup(vTagGroup TGRP){
```

```
	bool flg(true);
	int num(0);
	do{
		flg = true;
		if(!flg) TGRP.name=TGRP.name+"_("+IntToStr(num)+")";
		for(int i(0);i<GvTag.size();i++) if(GvTag[i].name==TGRP.name) flg =
false;
		num++;
	}while(!flg);
	std::auto_ptr<TMemoryStream> ms(new TMemoryStream);
	lv1->Items->Clear();
	//Копирование параметров
	ms->WriteComponent(lv1);
	ms->Position = 0;
	TTabSheet *TabSheet = new TTabSheet(pgc1); //Создание новой вкладки
	TabSheet->Parent = pgc1; //Объект-родитель (на котором создастся)
	flg=true;
	num=2;
	do{
		flg=true;
		String tN = "ts"+IntToStr(num);
		for(int i(0);i<GvTag.size();i++)
			if(GvTag[i].TTS->Name==tN){flg=false; num++;}
	}while(!flg);
	TabSheet->Name = "ts"+IntToStr(num); //Имя вкладки
	TabSheet->Caption = TGRP.name; //Отображаемое название
	TabSheet->PageControl = pgc1;
	TListView* pil = new TListView(TabSheet);
	pil->Parent = TabSheet; //Объект-родитель (на котором создастся)
	ms->ReadComponent(pil); //Клонирование параметров
	pil->Name="lv"+IntToStr(num);
	TGRP.TLV=pil;
	TGRP.TTS=TabSheet;
	GvTag.push_back(TGRP);
	PrepareTagsView();
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_bkTClick(TObject *Sender)
{
	cbb_ct->ItemIndex = -1;
	cbb_tT->ItemIndex = -1;
	pgc_cfg->ActivePageIndex=0;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_backClick(TObject *Sender)
{
	pgc_cfg->ActivePageIndex=2;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_openClick(TObject *Sender)
{
	RootPNL->Enabled=false;
	dlgOpen1->Execute();
	if(dlgOpen1->Files!="") if(crtProgT()) pgc_cfg->ActivePageIndex=0;
    RootPNL->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::cbb_tTChange(TObject *Sender)
{
	if(cbb_tT->Text=="По программе"&&chk_cng->Checked==true){
		cbb_ct->Visible=false;
		txt_CGC->Visible=false;
	}else{ cbb_ct->Visible=true; txt_CGC->Visible=true;}
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_TNextClick(TObject *Sender)
{
	switch(cbb_tT->ItemIndex){
		case 0:{
```

```
                        if(cbb_tN->Text==""&&chk_cng->Checked==false)
ShowMessage("Выберите тег!"); else
                        if(cbb_ct->Text=="") ShowMessage("Выберите группу!"); else{
                              if(chk_cng->Checked==false){
                                    edt_NC->Visible=false;
                                    txt_NT->Visible=false;
                                    btn_crtC->Caption="Изменить";
                              }
                              else{
                                    edt_NC->Visible=true;
                                    txt_NT->Visible=true;
                                    btn_crtC->Caption="Создать";
                              }
                              pgc_cfg->ActivePageIndex=4;
                        }
                        break;
                  }
                  case 1:{
                        if(cbb_tN->Text==""&&chk_cng->Checked==false)
ShowMessage("Выберите тег!"); else
                        if(cbb_ct->Text=="") ShowMessage("Выберите группу!"); else{
                              if(chk_cng->Checked==false){
                                    edt_NV->Visible=false;
                                    btn_crtVT->Caption="Изменить";
                                    Cabinet_1->Visible=false;
                              }
                              else{
                                    Cabinet_1->Visible=true;;
                                    edt_NV->Visible=true;
                                    btn_crtVT->Caption="Создать";
                              }
                              chk_V->Checked=false;
                              pgc_cfg->ActivePageIndex=5;
                        }
                        break;
                  }
                  case 2:{
                        if(cbb_tN->Text==""&&chk_cng->Checked==false)
ShowMessage("Выберите тег!"); else
                        if(cbb_ct->Text=="") ShowMessage("Выберите группу!"); else{
                              if(chk_cng->Checked==false){
                                    edt_NV->Visible=false;
                                    btn_crtVT->Caption="Изменить";
                                    Cabinet_1->Visible=false;
                              }
                              else{
                                    Cabinet_1->Visible=true;;
                                    edt_NV->Visible=true;
                                    btn_crtVT->Caption="Создать";
                              }
                              chk_V->Checked=true;
                              pgc_cfg->ActivePageIndex=5;
                        }
                        break;
                  }
                  case 3:{
                        if(chk_cng->Checked==false)   ShowMessage("Данный   тип   не
доступен!"); else
                        pgc_cfg->ActivePageIndex=3;
                        break;
                  }
                  default:{
                        ShowMessage("Выберите тип тега!");
                        break;
                  }
            }
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_crtCClick(TObject *Sender)
{
```

```
	if(chk_cng->Checked==false)
		if(edt_VT->Text!=""){
			double tmpD = edt_VT->Text.ToDouble();
			GvTag[cbb_ct->ItemIndex].FvTag[cbb_tN-
>ItemIndex].setConstTag(tmpD);
			pgc_cfg->ActivePageIndex=0;
			edt_NC->Text="";
			edt_VT->Text="";
		}else ShowMessage("Введите значение тега!");
	else
	if(edt_NC->Text!="") if(edt_VT->Text!=""){
		vTag s(FRecorder);
		string tmps;
		AnsiString tmpa = AnsiString(edt_NC->Text.c_str());
		tmps= tmpa.c_str();
		if(s.CreateTag(tmps)){
			double tmpD = edt_VT->Text.ToDouble();
			s.setConstTag(tmpD);
			int i = cbb_ct->ItemIndex;
			GvTag[i].FvTag.push_back(s);
			pgc_cfg->ActivePageIndex=0;
			PrepareTagsView();
			edt_NC->Text="";
			edt_VT->Text="";
		}else ShowMessage("Тег с данным названием уже существует!");
	}
	else ShowMessage("Введите значение тега!");
	else ShowMessage("Введите название тега!");
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::edt_VTChange(TObject *Sender)
{
	String tmp_;
	bool flg(false);
	bool flg_(false);
	for(int i(1);i<=edt_VT->Text.Length();i++){
		String tmps = edt_VT->Text[i];
		if(!flg&&i!=1&&tmps==","){
			tmp_+=tmps;
			flg = true;
		}
		if(!flg_&&i==1&&tmps=="-"){
			tmp_+=tmps;
			flg_ = true;
		}

	if(tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||tmps
=="7"||tmps=="8"||tmps=="9"||tmps=="0")
			tmp_+=tmps;
	}
	edt_VT->Text=tmp_;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_crtVTClick(TObject *Sender)
{
	if(chk_cng->Checked==false)
		if(edt_minV->Text!=""&&edt_maxV->Text!=""&&edt_step->Text!=""){
	       double tmpMIN = edt_minV->Text.ToDouble();
			double tmpMAX = edt_maxV->Text.ToDouble();
			double tmpStep = edt_step->Text.ToDouble();
			if(chk_V->Checked)    GvTag[cbb_ct->ItemIndex].FvTag[cbb_tN-
>ItemIndex].setSmoothlyTag(tmpMIN,tmpMAX,tmpStep);
			else                  GvTag[cbb_ct->ItemIndex].FvTag[cbb_tN-
>ItemIndex].setVariableTag(tmpMIN,tmpMAX,tmpStep);
			edt_NV->Text="";
			edt_minV->Text="";
			edt_maxV->Text="";
			edt_step->Text="";
			pgc_cfg->ActivePageIndex=0;
		}else ShowMessage("Введите значения тега!");
```

```
	else
	if(edt_NV->Text!="")  if(edt_minV->Text!=""&&edt_maxV->Text!=""&&edt_step-
>Text!=""){
		vTag s(FRecorder);
		string tmps;
		AnsiString tmpa = AnsiString(edt_NV->Text.c_str());
		tmps= tmpa.c_str();
		if(s.CreateTag(tmps)){
			double tmpMIN = edt_minV->Text.ToDouble();
			double tmpMAX = edt_maxV->Text.ToDouble();
			double tmpStep = edt_step->Text.ToDouble();
			if(chk_V->Checked) s.setSmoothlyTag(tmpMIN,tmpMAX,tmpStep);
			else s.setVariableTag(tmpMIN,tmpMAX,tmpStep);
			int i = cbb_ct->ItemIndex;
			GvTag[i].FvTag.push_back(s);
			edt_NV->Text="";
			edt_minV->Text="";
			edt_maxV->Text="";
			edt_step->Text="";
			pgc_cfg->ActivePageIndex=0;
			PrepareTagsView();
		}else ShowMessage("Тег с данным названием уже существует!");
	}
	else ShowMessage("Введите значения тега!");
	else ShowMessage("Введите название тега!");
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::edt_minVChange(TObject *Sender)
{
	String tmp_;
	bool flg(false);
	bool flg_(false);
	for(int i(1);i<=edt_minV->Text.Length();i++){
		String tmps = edt_minV->Text[i];
		if(!flg&&i!=1&&tmps==","){
			tmp_+=tmps;
			flg = true;
		}
		if(!flg_&&i==1&&tmps=="-"){
			tmp_+=tmps;
			flg_ = true;
		}

	if(tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||tmps
=="7"||tmps=="8"||tmps=="9"||tmps=="0")
			tmp_+=tmps;
	}
	edt_minV->Text=tmp_;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::edt_stepChange(TObject *Sender)
{
	String tmp_;
	bool flg(false);
	bool flg_(false);
	for(int i(1);i<=edt_step->Text.Length();i++){
		String tmps = edt_step->Text[i];
		if(!flg&&i!=1&&tmps==","){
			tmp_+=tmps;
			flg = true;
		}
		if(!flg_&&i==1&&tmps=="-"){
			tmp_+=tmps;
			flg_ = true;
		}

	if(tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||tmps
=="7"||tmps=="8"||tmps=="9"||tmps=="0")
			tmp_+=tmps;
	}
```

```
	edt_step->Text=tmp_;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::edt_maxVChange(TObject *Sender)
{
	String tmp_;
	bool flg(false);
	bool flg_(false);
	for(int i(1);i<=edt_maxV->Text.Length();i++){
		String tmps = edt_maxV->Text[i];
		if(!flg&&i!=1&&tmps==","){
			tmp_+=tmps;
			flg = true;
		}
		if(!flg_&&i==1&&tmps=="-"){
			tmp_+=tmps;
			flg_ = true;
		}

	if(tmps=="1"||tmps=="2"||tmps=="3"||tmps=="4"||tmps=="5"||tmps=="6"||tmps
=="7"||tmps=="8"||tmps=="9"||tmps=="0")
			tmp_+=tmps;
	}
	edt_maxV->Text=tmp_;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_delGClick(TObject *Sender)
{
	cbb_G->Clear();
	for(int i(0);i<GvTag.size();i++) cbb_G->Items->Add(GvTag[i].name);
	pgc_cfg->ActivePageIndex=6;
}
//---------------------------------------------------------------------------

void __fastcall TGenForm::btn_nDGClick(TObject *Sender)
{
	if(cbb_G->Text!=""){
		for(int                                          i(0);i<GvTag[cbb_G-
>ItemIndex].FvTag.size();i++)GvTag[cbb_G->ItemIndex].FvTag[i].deleteTag();
		GvTag[cbb_G->ItemIndex].FvTag.clear();
		delete GvTag[cbb_G->ItemIndex].TLV;
		delete GvTag[cbb_G->ItemIndex].TTS;
		GvTag.erase(GvTag.begin()+cbb_G->ItemIndex);
		PrepareTagsView();
		pgc_cfg->ActivePageIndex=0;
	}else ShowMessage("Выберите группу");
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_delTClick(TObject *Sender)
{
	cbb_GDT->Clear();
	cbb_DT->Visible=false;
	txt_CTDT->Visible=false;
	for(int i(0);i<GvTag.size();i++) cbb_GDT->Items->Add(GvTag[i].name);
	pgc_cfg->ActivePageIndex=7;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::cbb_GDTChange(TObject *Sender)
{
	cbb_DT->Clear();
	for(int i(0);i<GvTag[cbb_GDT->ItemIndex].FvTag.size();i++){
		string tmp = GvTag[cbb_GDT->ItemIndex].FvTag[i].getName();
		String tmp2 = tmp.c_str();
		cbb_DT->Items->Add(tmp2);
	}
	cbb_DT->Visible=true;
	txt_CTDT->Visible=true;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_NDTClick(TObject *Sender)
```

```
{
	if(cbb_GDT->Text!=""){
		if(cbb_DT->Text!=""){
			if(GvTag[cbb_GDT->ItemIndex].FvTag[cbb_DT-
>ItemIndex].deleteTag())
			GvTag[cbb_GDT->ItemIndex].FvTag.erase(GvTag[cbb_GDT-
>ItemIndex].FvTag.begin()+cbb_DT->ItemIndex);
			pgc_cfg->ActivePageIndex=0;
		    PrepareTagsView();
		     }else ShowMessage("Выберите тег");
	}else ShowMessage("Выберите группу");
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::chk_sinchClick(TObject *Sender)
{
	if(chk_sinch->Checked) chk_sinch->Caption="Синхронное изменение";
	else chk_sinch->Caption="Асинхронное изменение";
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_chgTClick(TObject *Sender)
{
	cbb_ct->Clear();
	cbb_tN->Clear();
	for(int i(0);i<GvTag.size();i++)
		cbb_ct->Items->Add(GvTag[i].name);
	cbb_ct->Visible=true;
	txt_CTDT->Visible=true;
	cbb_tN->Visible=false;
    txt_CTCT->Visible=false;
	chk_cng->Checked=false;
	cbb_tT->ItemIndex = -1;
	if(cbb_tT->Items->Count==4) cbb_tT->Items->Delete(3);
	pgc_cfg->ActivePageIndex=2;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::cbb_ctChange(TObject *Sender)
{
	if(chk_cng->Checked){cbb_tN->Visible=false; txt_CTDT->Visible=false;}
	else{
		cbb_tN->Clear();
		for(int i(0);i<GvTag[cbb_ct->ItemIndex].FvTag.size();i++){
			string tmp = GvTag[cbb_ct->ItemIndex].FvTag[i].getName();
			String tmp2 = tmp.c_str();
			if(GvTag[cbb_ct->ItemIndex].FvTag[i].getType()!="По
программе")
				cbb_tN->Items->Add(tmp2);
		}
		cbb_tN->Visible=true;
		txt_CTCT->Visible=true;
	}
}
//---------------------------------------------------------------------------

bool __fastcall TGenForm:: LockChenge(void){
	pgc_cfg->ActivePageIndex=0;
	btn_crtT->Enabled=false;
	btn_crtT->Caption="Заблокировано";
	btn_crtG->Enabled=false;
	btn_crtG->Caption="Заблокировано";
	btn_delG->Enabled=false;
	btn_delG->Caption="Заблокировано";
	btn_delT->Enabled=false;
	btn_delT->Caption="Заблокировано";
	return true;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::status_pnlDblClick(TObject *Sender)
{
	if(GvTag.size()==1)
		if(GvTag[0].name=="ver_root"&GvTag[0].FvTag.size()==4)
```

```
                    if(MessageDlg(L"Войти в режим иммитации COM-порта?",
mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0) == mrYes){
                          pgc_cfg->ActivePageIndex=8;
                    }
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::cbb_comDropDown(TObject *Sender)
{
       cbb_com->Clear();
       AnsiString KeyName = "\\Hardware\\DeviceMap\\SerialComm";
       TStringList *SerialCommValues = new TStringList();
       TRegistry *Registry = new TRegistry;
       try{
        Registry->RootKey = HKEY_LOCAL_MACHINE;
        Registry->OpenKeyReadOnly( KeyName );
        Registry->GetValueNames( SerialCommValues );
             for(int i=0; i<SerialCommValues->Count; i++){
                    cbb_com->Items->Add(Registry->ReadString(SerialCommValues-
>Strings[i]));
        }
    }
    __finally{
        delete Registry;
             delete SerialCommValues;
       }

}
//---------------------------------------------------------------------------

void __fastcall TGenForm::btn_cbClick(TObject *Sender)
{
       cbb_com->Clear();
       tmr2->Enabled=false;
       pgc_cfg->ActivePageIndex=0;
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::btn_ocClick(TObject *Sender)
{
       if(cbb_com->Text!="") tmr2->Enabled=true;
       else ShowMessage("Выберите COM-порт!");
}
//---------------------------------------------------------------------------
void __fastcall TGenForm::tmr2Timer(TObject *Sender)
{
       OVERLAPPED OverlapWRT;
    OverlapWRT.OffsetHigh = 0;
    OverlapWRT.Offset = 0;
    OverlapWRT.hEvent = CreateEvent(NULL, false, false, NULL);

       OVERLAPPED OverlapRD;
    OverlapRD.OffsetHigh = 0;
       OverlapRD.Offset = 0;
       OverlapRD.hEvent = NULL;

       tmr2->Enabled=false;
       bool flg=false;
       HANDLE hSerial;
       LPCTSTR sPortName = cbb_com->Text.w_str();
       hSerial = CreateFile(sPortName,
             GENERIC_READ | GENERIC_WRITE,
             FILE_SHARE_READ,
             NULL,
             OPEN_EXISTING,
             FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
                    NULL);
       //hSerial         =         ::CreateFile(sPortName,GENERIC_READ         |
GENERIC_WRITE,0,0,OPEN_EXISTING ,FILE_ATTRIBUTE_NORMAL,0);
       if(hSerial==INVALID_HANDLE_VALUE){
              if(GetLastError()==ERROR_FILE_NOT_FOUND){
                    flg=true;
```

```
                        ShowMessage("serial port does not exist.");
                }
                flg=true;
                ShowMessage("some other error occurred.");
          }

        DCB dcbSerialParams = {0};
          dcbSerialParams.DCBlength=sizeof(dcbSerialParams);
          if (!GetCommState(hSerial, &dcbSerialParams)){
                flg=true;
                ShowMessage("getting state error");
          }
          dcbSerialParams.BaudRate=CBR_9600;
          dcbSerialParams.ByteSize=8;
          dcbSerialParams.StopBits=ONESTOPBIT;
          dcbSerialParams.Parity=NOPARITY;
          if(!SetCommState(hSerial, &dcbSerialParams)){
                flg=true;
                ShowMessage("error setting serial port state");
          }
          DWORD iSize;
          char sReceivedChar;
          string tmp;
          for(int i(0);i<100;i++){
                ReadFile(hSerial, &sReceivedChar, 1, &iSize, &OverlapRD);
                if (iSize > 0 ){
                      tmp+=sReceivedChar;
                }
          }
          if(tmp.find("*IDN?")!=-1){
                char data[] = "KBIS,N4-14,,1.1";
                DWORD dwSize = sizeof(data);
                DWORD dwBytesWritten;
                LPOVERLAPPED ov;
                for(int                 i(0);i<2000;i++)                WriteFile
      (hSerial,data,dwSize,&dwBytesWritten ,NULL);
          }else if(tmp.find("CONF:RES")!=-1){
                tmp.erase(0,8);
                String tmp1 = tmp.c_str();
                double tmp_d = StrToFloat(tmp1);
                for(INT i(0);i<4;i++){
                      GvTag[0].FvTag[i].setConstTag(tmp_d);
                }
          }
        UpdateTagData();
          CloseHandle(hSerial);
          if(!flg)tmr2->Enabled=true;
}
```

ПРИЛОЖЕНИЕ В.

Техническое задание

СОГЛАСОВАНО	УТВЕРЖДАЮ
Доцент	Заведующий кафедрой Информационных
_____ А.Л. Погудин	технологий и автоматизированных систем
«___»_____20__	_____ Р.А. Файзрахманов
	«___»_____20__

Разработка программного модуля калибровки измерительных каналов с входным сигналом на модули MR-212 в составе стенда статических испытаний

Техническое задание

На 13 листах
Действует с __.__.__

Разработал студент гр. ЭВТ-16-1бзу
_____ Габерман С.А.
«___»_____2020

– Содержание:

1. Общие сведения

1.1. Полное наименование системы и ее условное обозначение

Настоящее Техническое задание определяет требования и порядок реализации программного модуля калибровки измерительных каналов с входным сигналом на модули MR-212 в составе стенда статических испытаний.

1.2. Шифр темы

Настоящее Техническое задание разработано в рамках выполнения выпускной квалификационной работы бакалавра. Кафедра ИТАС, специальность 09.03.01 – Информатика и вычислительная техника (ИВТ).

1.3. Наименование предприятия разработчика и заказчика

Заказчик: АО «ОДК–Авиадвигатель»

Адрес фактический: г. Пермь, Комсомольский проспект 93.

Телефон / Факс: +7 (342) 240-92-67 / +7 (342) 281-54-77

Исполнитель: студент группы ЭВТ-16-1бзу Габерман Сергей Александрович (далее – исполнитель).

1.4. Основание для разработки

Настоящая работа выполняется на основании учебного плана специальности «09.03.01 – Информатика и вычислительная техника (ИВТ)» и темы выпускной квалификационной работы, согласованной с заведующим кафедрой ИТАС ПНИПУ, Файзрахмановым Р.А.

1.5. Источники и порядок финансирования

Источники финансирования отсутствуют.

1.6. Плановые сроки начала и окончания работ

Срок проектирования модуля регламентирован действующим учебным планом и предполагает сдачу проекта подсистемы до 28.01.2020 г.

1.7. Порядок оформления и предъявления результатов работ

Работы по созданию программного модуля производятся и принимаются поэтапно. По окончании каждого из этапов работ, установленных Учебным планом, студенты представляют заведующему кафедрой соответствующие результаты. Заведующий кафедрой оценивает работу.

2. Назначение и цели проектирования программного модуля

2.1. Назначение программного модуля

Проектируемый программный модуль входит в состав измерительной системы стенда статических испытаний АРД, Программный модуль предназначен для проведения калибровки измерительных каналов с входным сигналом на модули MR-212 в составе стенда статических испытаний.

2.2. Цель и задачи проектирования подсистемы

Целью данной работы является создание программного модуля решения задачи калибровки измерительных каналов с входным сигналом на модули MR-212 в составе стенда статических испытаний.

Для достижения поставленной цели решались следующие задачи:

1. изучить предметную область;
2. спроектировать программный модуль;
3. реализовать программный модуль.

3. Характеристика объекта автоматизации

3.1. Краткие сведения об объекте автоматизации

Объектом автоматизации является процесс калибровки измерительных каналов с входным сигналом на модули MR-212 в составе стенда статических испытаний.

3.2. Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды

3.2.1. Условия эксплуатации комплекса технических средств

Все процессы, протекающие в модуле, осуществляются посредством имеющегося программного обеспечения.

3.2.2. Характеристики окружающей среды

Во время эксплуатации программного модуля характеристики окружающей среды должны соответствовать требованиям существующей методики поверки:

- температура окружающего воздуха, °C 20 ± 5
- относительная влажность воздуха, % 30...80
- атмосферное давление, кПа 84...106.7 (мм. рт. ст. 630...800)

4. Требования к программному модулю

4.1. Требования к программному модулю в целом

Общие требования к программному модулю:

- Одновременная работа с четырьмя каналами (со всем модулем MR-212);
- Задание количества и значений точек калибровки;
- Возможность возвращения к предыдущей точке калибровки;
- Фиксирование усреднённого значения канала по трём замерам в прямом ходе и обратном;

– Расчёт основной приведенной погрешности;

– Автоматическое генерирование протокола калибровки по её завершению;

– Программный модуль должен быть реализован на языке C++ или Delphi для обеспечения программной совместимости с ПО «Recorder».

4.1.1. Требования к интерфейсу программного модуля

Интерфейс программного модуля должен состоять из двух блоков: отображения данных и управления. В блоке отображения данных находится таблица, в которой располагаются зафиксированные данные ИК на каждой точке калибровки\поверки. В блоке управления располагаются элементы управления модулем.

Требования к интерфейсу могут быть дополнены или изменены в процессе испытания.

4.1.2. Требования к численности и квалификации персонала

Для работы с модулем, сотрудникам необходим доступ к программному модулю. Все сотрудники, работающие с модулем, должны пройти обучение работе с программным модулем, а также иметь соответствующую метрологическую квалификацию.

4.1.3. Требования к надежности программного модуля

Программный модуль должен безошибочно выполнять все требуемые функции, и быть пригодным для эксплуатации.

4.1.4. Требования к безопасности

Не предъявляются.

4.1.5. Требования по эргономике

Для установки системы необходимо оборудовать рабочие места компьютером. Необходим компьютерный стол, компьютерное кресло, хорошее освещение. Должно быть установлено следующее ПО: операционной системой "Windows 7 и выше, с установленным ПО «Recorder».

4.2. Требования к функциям

Программа должна обеспечивать безошибочное выполнение перечисленных ниже функций:

- Фиксирование данных в памяти модуля;
- Расчёт основной приведенной погрешности;
- Генерирование протокола калибровки.

4.3. Требования к видам обеспечения

4.3.1. Требования к техническому обеспечению

Для работы программного модуля рабочее место должны иметь конфигурацию не ниже, чем: Intel Core i5-2500 по 3,3ГГц, ОЗУ не менее 8Гб, HDD не менее 1Тб, 1Гб GeForce GTS 450, DVD±RW.

4.3.2. Требования к ПО

Для работы программного модуля необходимо установленное ПО “Recorder”.

5. Состав и содержание работ по созданию подсистемы

5.1. Этапы создания продукта

Этапы создания продукта приведены в таблице 1.

Таблица В.1 – этапы создания продукта

Наименование этапа	**Срок исполнения**
Анализ исходных данных	15.09.2019
Решение задач покарки\калибровки	20.12.2019
Реализация программного модуля	01.01.2020
Практическая реализация	15.01.2020
Написание пояснительной записки	20.01.2020

6. Порядок контроля и приемки модуля

6.1. Общие требования к приемке работ

Сдача-приемка этапов выполненных работ осуществляется по предъявлении Исполнителем отчета по выпускной квалификационной работе и других сопроводительных комплектов документов.

Сдача-приемка этапов выполненных работ завершается оформлением акта сдачи-приемки научно-технической продукции, подписанного Исполнителем, и утвержденного Заказчиком.

Погрешности в программном обеспечении и эксплуатационной документации, обнаруженные при сдаче подсистемы в опытную эксплуатацию или в процессе опытной эксплуатации, должны быть устранены Исполнителем до предъявления подсистемы на приемочные испытания.

Порядок и сроки проведения приемочных испытаний согласуются на этапе «Опытная эксплуатация».

Этап «Проведение приемочных испытаний» заканчивается оформлением акта о приемке подсистемы в промышленную эксплуатацию, подписанного специально для этого созданной комиссией.

6.2. Требования к испытаниям программного модуля

6.2.1. Виды испытаний

Для подсистемы устанавливаются следующие виды испытаний:

– Опытная эксплуатация;

– Приемочные испытания.

6.2.2. Состав испытаний

Опытная эксплуатация проводится в реальном режиме работы всех подразделений организации Заказчика, охватываемых модулем. Все функции персонала подразделения, автоматизация которых предусмотрена в модуле, должны выполняться с использованием средств автоматизации.

Во время опытной эксплуатации на всех объектах автоматизации должны вестись рабочие журналы, в которые должны заноситься сведения о продолжительности функционирования модуля, отказах, сбоях, аварийных ситуациях, изменениях параметров объекта автоматизации, проводимых корректировках документации и программных средств, наладке технических средств, а также замечания персонала по удобству эксплуатации модуля.

Приемочные испытания проводятся в соответствии с Программой и методикой приемочных испытаний путем выполнения комплексных тестов, подготовленных Разработчиком и согласованных с Заказчиком.

6.2.3. Место проведения испытаний

АО «ОДК–Авиадвигатель»

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу подсистемы в действие

7.1. Создание условий функционирования объекта автоматизации

Для создания условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемым подсистемой требованиям,

содержащимся в настоящем Техническом задании, и возможность эффективного использования модуля, в организации Заказчика на этапе работ «Подготовка объекта автоматизации к вводу системы в действие» должен быть проведен комплекс мероприятий.

7.2. Изменения, которые необходимо осуществить в объекте автоматизации

7.2.1. Организационные мероприятия

Для поддержки функционирования подсистемы Заказчиком до начала этапа «Ввод в действие» должна быть создана Служба эксплуатации.

После завершения этапа работ «Разработка рабочей документации» Исполнитель должен предоставить Заказчику Программу обучения персонала работе с модулем.

8. Требования к документированию

8.1. Документация технического проекта

8.1.1. Пояснительная записка к техническому проекту

Пояснительная записка к техническому проекту должна содержать:

– Общие положения;
– Описание процесса деятельности;
– Описание методики технического проектирования;
– Описание основных технических решений.

8.1.2. Описание информационного обеспечения

Описание информационного обеспечения должно содержать:

– Состав информационного обеспечения;
– Описание организации информационного обеспечения;
– Описание организации сбора, обработки и передачи информации;
– Перечень входных/выходных документов и данных.

8.1.3. Описание комплекса технических средств

Описание комплекса технических средств должно содержать описание решений по оснащению рабочих мест.

8.1.4. Описание программного обеспечения

Описание программного обеспечения должно содержать:

– Описание компонентов ПО, их назначения и функций;

– Перечень используемых операционных систем и всех их расширений.

8.2. Эксплуатационная документация

8.2.1. Общее описание модуля

Общее описание модуля должно содержать:

– Назначение модуля;

– Вид деятельности, для автоматизации которой предназначен модуль;

– Перечень функций, реализуемых модулем.

8.2.2. Руководство системного администратора

Руководство системного администратора должно включать порядок интеграции программного модуля.

8.2.3. Руководства пользователей

Для осуществления функций модуля должно быть разработано руководство пользователя.

8.3. Требования к документированию комплектующих элементов

Общее программное обеспечение и технические средства снабжаются документацией, входящей в поставляемый производителем комплект.

8.4. Общие требования к документированию

Документация в электронном виде может быть получена отдельно от дистрибутива.

Помимо документации в электронном виде, пользователям и разработчикам может быть представлена документация на бумажных носителях.

9. Источники разработки

Исходными документами для разработки настоящего технического задания являются:

– Нормативно-техническая документация Заказчика;

– ГОСТ 34.602-89;

– Образцы рабочих документов, полученных в процессе обследования;

– Информационные материалы и проектная документация на аналогичные автоматизированные системы.

ПРИЛОЖЕНИЕ Г.

Протокол измерений при поверке измерительного канала

Тип поверяемого измерительного канала: канал измерения деформации

Индекс ИК {1-1-1} Наименование ИК {1-1-1}

Диапазон измерений -3000…+3000 Еденицы измерений МЛН^(−1)

Средства измерений применяемые при поверке_____

_____

Нормированное значение погрешности измерительного канала ±0,2%

Измеряемая величина деформация Дата поверки_____					
% шкалы изм.	0	25	50	75	100
Значение	-3037,38	-1542,06	0	1542,056	3037,38
№ измерений	Результаты измерений. Прямой ход				
1	-3037,37	-1542,06	0,00	1542,06	3037,38
2	-3037,37	-1542,06	0,00	1542,06	3037,38
3	-3037,37	-1542,06	0,00	1542,06	3037,38
4	-3037,37	-1542,06	0,00	1542,06	3037,38
5	-3037,37	-1542,06	0,00	1542,06	3037,38
	Результаты измерений. Обратный ход				
1	-3037,37	-1542,06	0,00	1542,06	3037,38
2	-3037,37	-1542,06	0,00	1542,06	3037,38
3	-3037,37	-1542,06	0,00	1542,06	3037,38
4	-3037,37	-1542,06	0,00	1542,06	3037,38
5	-3037,37	-1542,06	0,00	1542,06	3037,38
Среднее значение	-3037,37	-1542,06	0,00	1542,06	3037,38
	Погрешность измерений				
Основная приведенная погрешность	0,00	0,00	0,00	0,00	0,00
Результаты поверки изм. канала	Годен (негоден)				

Поверитель_____