ООО "Научно-производственное предприятие "Мера"

Инструментальное программное обеспечение измерительного комплекса ПО «Recorder». Руководство программиста

г. Королёв 2004 г.

Содержание

1 Инструментальное программное обеспечение измерительного комплекса

Одной из наиболее важных составных частей измерительного комплекса является программное обеспечение. Инструментальное программное обеспечение, которое производит управление аппаратными ресурсами комплекса, выполняет получение измеренных данных и (или) подготовку данных для генерации сигналов, обработку, регистрацию данных и отображение.

Предметом данного документа является ПО «Recorder», структура и программные интерфейсы, которые позволяют специализировать и наращивать его функциональность.

ПО «Recorder» решает, перечисленные выше задачи, для измерительных вычислительных комплексов MIC.

Основные функции ПО «Recorder»:

1. Предоставление возможности настройки и тестирования аппаратных устройств. ПО «Recorder» содержит группу диалоговых окон, которые позволяют необходимым образом настроить устройства, и их каналы для процесса измерения и (или) генерации сигнала.
2. Сохранение настройки в файлы и чтение из файлов. Эта функция позволяет хранить в виде файлов несколько различных настроек и загружать любую из них при необходимости.
3. Получение измеренных данных от устройств и сохранение измеренных данных в файлы. Регистрация измеренных сигналов в файловом виде позволяет производить анализ измеренных данных в любой момент времени после завершения процесса измерения.
4. 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.
5. Отображение измеренных данных в виде осциллограмм и в виде таблицы. В табличном виде отображаются математические оценки измеренных данных: математическое ожидание, среднеквадратическое значение, среднеквадратическое отклонение, амплитуда, размах.

ПО «Recorder» является комплексным программным продуктом, состоит из группы независимых программных модулей, обладает развитой внутренней структурой.

2 Структура инструментального программного обеспечения

ПО «Recorder» состоит из четырех основных функциональных блоков. Упрощенная структурная схема показана на рисунке 1. Это блоки: блок драйверов, блок управления, блок тегов, блок модулей plug-in`ов. Блок драйверов позволяет производить управление и обмен данными с аппаратными устройствами. Блок управления служит для координации работы всех блоков, для чтения и записи настройки, для инициализации и тестирования аппаратных устройств и т.п. Блок тегов представляет собой группу именованных буферов для хранения измеренных и обработанных данных и для данных, подготовленных к генерации сигналов. Блок модулей plug-in`ов предназначен для управления библиотеками дополнительных функций.

Кроме уже упомянутых функций, программные блоки ПО «Recorder» выполняют функции предварительной обработки и сохранения измеренных данных, отображения измеренных и обработанных данных.

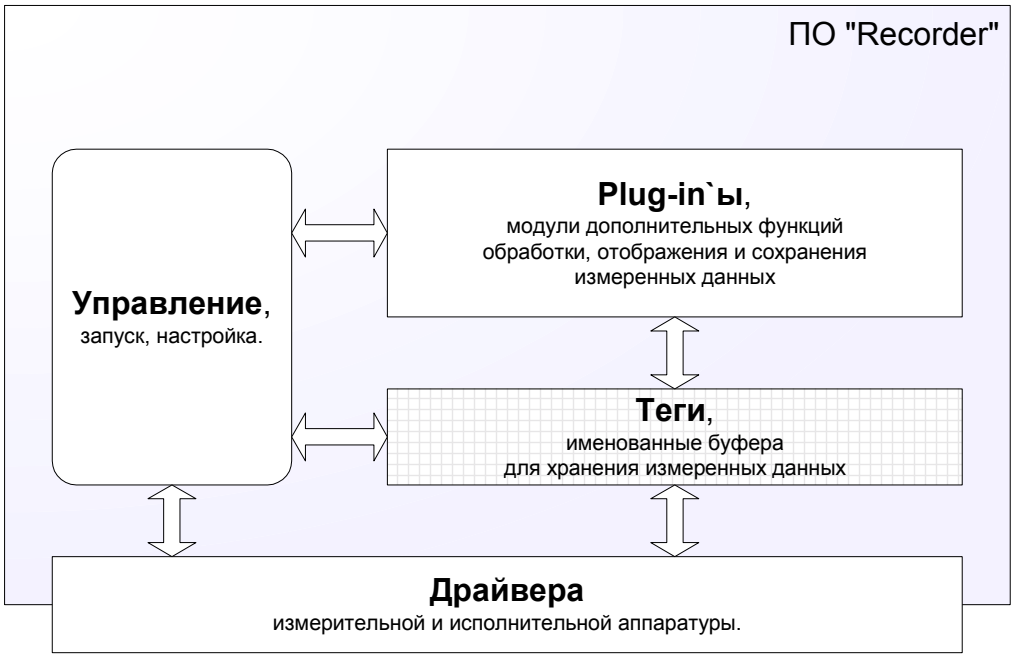


Рисунок 1 Упрощенная структура ПО «Recorder»

Блочная структура программы позволяет разрабатывать и отлаживать ее по частям, отдельными блоками – программными модулями, наращивать функциональность отдельных блоков, модернизировать, дорабатывать их, не затрагивая другие.

Блок управления, блок драйверов, блок тегов, функции управления модулями plug-in`ов, функции регистрации данных вместе составляют ядро ПО «Recorder». Ядро предназначено для решения общих задач инструментального ПО, таких как инициализация аппаратных средств, их тестирование, получение измеренных данных, регистрация данных и т.п. Специализация ПО «Recorder» выполняется при помощи отдельных модулей plug-in`ов.

Для взаимодействия ядра ПО «Recorder» и модулей plug-in`ов разработан специальный программный интерфейс PluginAPI. Интерфейс PluginAPI позволяет:

1. читать данные из тегов и записывать данные в теги,
2. получать оповещения о поступлении новых данных от устройств, в процессе измерения,
3. получать параметры настройки ПО «Recorder», его внутренних объектов,
4. производить изменение настройки ПО «Recorder», его внутренних объектов,
5. отображать специализированные формуляры, с таблицами, с гистограммами, осциллограммами и т.п.,
6. получать оповещения об изменении внутреннего состояния ядра ПО «Recorder», к примеру, когда программа переходит в режим измерения или завершает режим измерения или переходит в режим настройки и т.п.

Интерфейс PluginAPI позволяет получать, в модулях plug-in`ов, полный объем real-time измеренных данных, поступивших от аппаратных устройств. ПО «Recorder» выполняет буферизированный ввод данных. Потоки измеренных данных из драйверов устройств поступают в буферы – теги, для каждого измерительного канала существует отдельный тег. О получении новых порций данных ядро ПО «Recorder» оповещает все модули plug-

in`ов и внутренние функциональные блоки. Данные, хранимые тегами, используются для обработки, отображения и регистрации в файлы. Схема протекания потоков данных показана на рисунке 2.

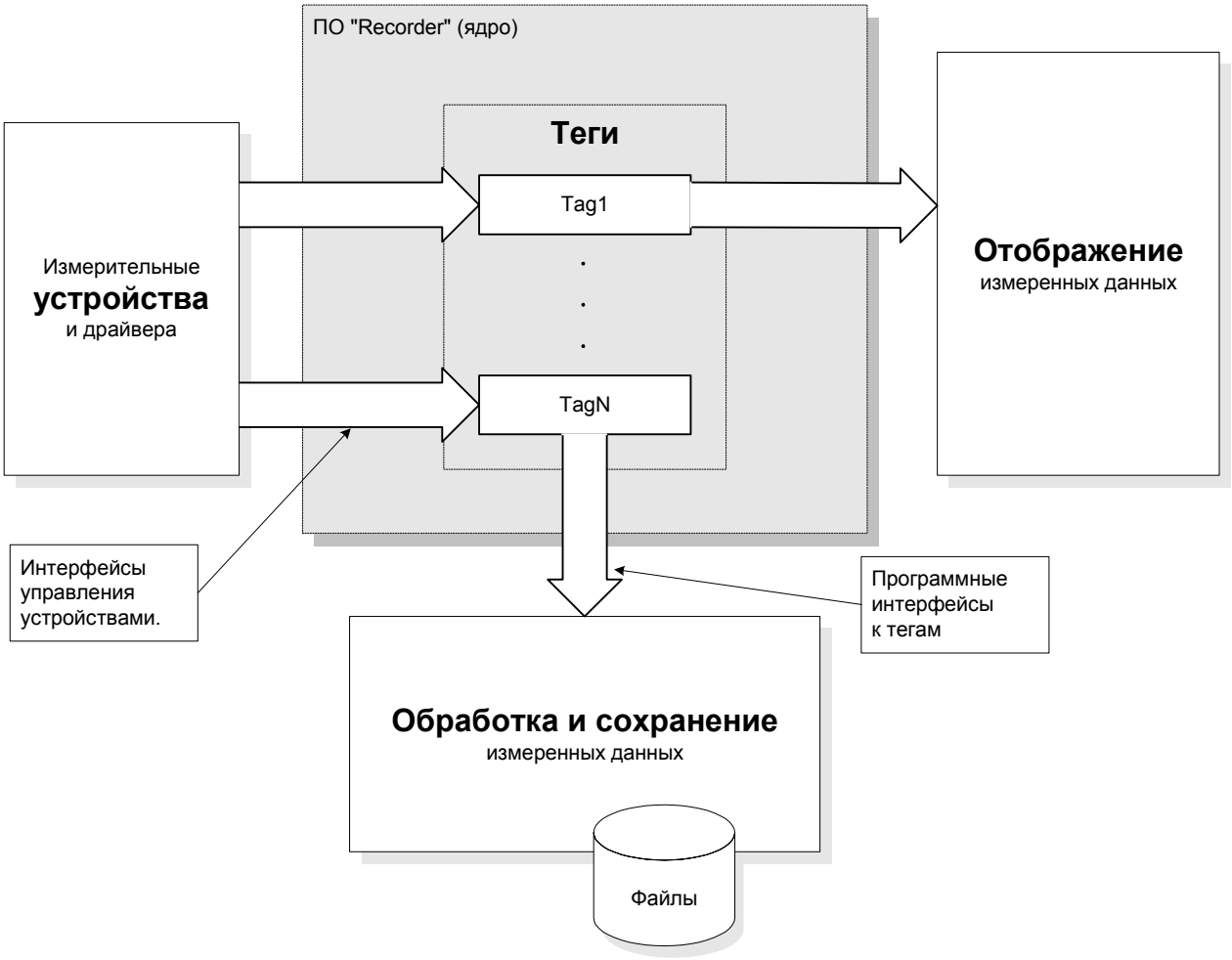


Рисунок 2. Схема протекания потоков измеренных данных в ПО «Recorder»

2.1 Функциональные блоки ПО «Recorder»

2.1.1 Драйверы аппаратного обеспечения.

Драйверы устройств – специальные программные модули, которые работают с ресурсами аппаратных измерительных и исполнительных устройств, такими как регистры, области памяти DMA, прерывания и пр. Каждое устройство может обладать собственными регистрами, и собственной настройкой, особенными режимами работы и тестирования. Для каждого устройства существует драйвер. Основное назначение драйверов – это реализация унифицированного, полнофункционального программного интерфейса, который позволит максимально одинаково взаимодействовать со всем перечнем разнотипных устройств.

Драйверы устройств работают в рамках ядра операционной системы для обеспечения необходимого быстродействия обработки запросов и для непосредственного использования ресурсов аппаратных устройств.

Программный интерфейс драйверов DevAPI, позволяет ПО «Recorder» выполнять следующие функции:

1. поиск аппаратных устройств, непосредственно присутствующих в системе (физически подключенных к ИВК);
2. инициализация аппаратных устройств, тестирование работоспособности;
3. настройка аппаратных устройств и их отдельных каналов для выполнения измерения и (или) генерации сигналов;
4. получение измеренных данных и вывод данных для генерации сигналов.

Программный интерфейс DevAPI и основные принципы работы драйверов описан в [1].

2.1.2 Список тегов.

Одним из основных блоков, структуры ПО «Recorder», в процессе обработки данных является список тегов. Тег – это программный объект, у которого есть ряд свойств, таких как имя, срока описания, тег обладает буфером для хранения данных. Буфер тега является кольцевым, данные в нем хранятся блоками, в блоке может быть от одного до нескольких значений.

Теги делятся на две категории: теги, предназначенные для обмена данными с аппаратными каналами, и теги, предназначенные для хранения данных в памяти. При помощи тегов, связанных с аппаратным каналом, осуществляется управление каналом. Теги позволяют устанавливать свойства каналов, такие как частота дискретизации, диапазон измерения, параметры функций калибровки чувствительности, градуировки и т.п.

Буфер данных тега, связанного с аппаратным каналом, хранит, либо измеренные данные, полученные от драйвера устройства, либо данные, подготовленные к передаче в исполнительное устройство.

Буфер тега, предназначенного для хранения данных в памяти, используется модулями plug-in`ов. В этот буфер заносятся обработанные данные. Модули plug-in`ов имеют возможность создавать теги и записывать в них данные.

2.1.3 Список plug-in`ов.

Plug-in – это программный модуль, который позволяет дополнять и расширять функциональность ПО «Recorder».

Функции plug-in`ов:

1. Любая обработка измеренных данных – это вычисление каких-либо оценок, совместный анализ данных нескольких тегов, анализ данных на превышение уставок и т.п.
2. Генерация сигналов, к примеру, формирование управления для исполнительных устройств в зависимости от измеренных данных.
3. Сохранение каких-либо промежуточных данных в файловом виде.
4. Отображение измеренных и обработанных данных в любом виде (графики, таблицы, гистограммы, и пр.)
5. Формирование отчетов, журналов с измеренными и обработанными данными, для печати.

6. Передача данных в любом виде в другие программы, в том числе и через информационную сеть.
7. Изменение режима работы ПО «Recorder», изменение настройки, создание тегов, создание групп, и т.п.

2.1.4 Блок управления

Блок управления – это группа программных объектов, выполняющих общее управление ПО «Recorder».

2.2 Расширение функциональности ПО «Recorder»

Ниже приведен ряд типовых решений с использованием ПО «Recorder», описываются модули plug-in`ов, которые дополняют функции ПО «Recorder».

ПО «Recorder» отображает измеренные данные либо в табличном виде, либо в виде осциллограмм. Если необходимы специальные формуляры для просмотра измеренных данных, если необходима обработка измеренных данных, следует создать новый модуль plug-in`a. Взаимодействие plug-in`a и ПО «Recorder» иллюстрировано на рисунке 3.

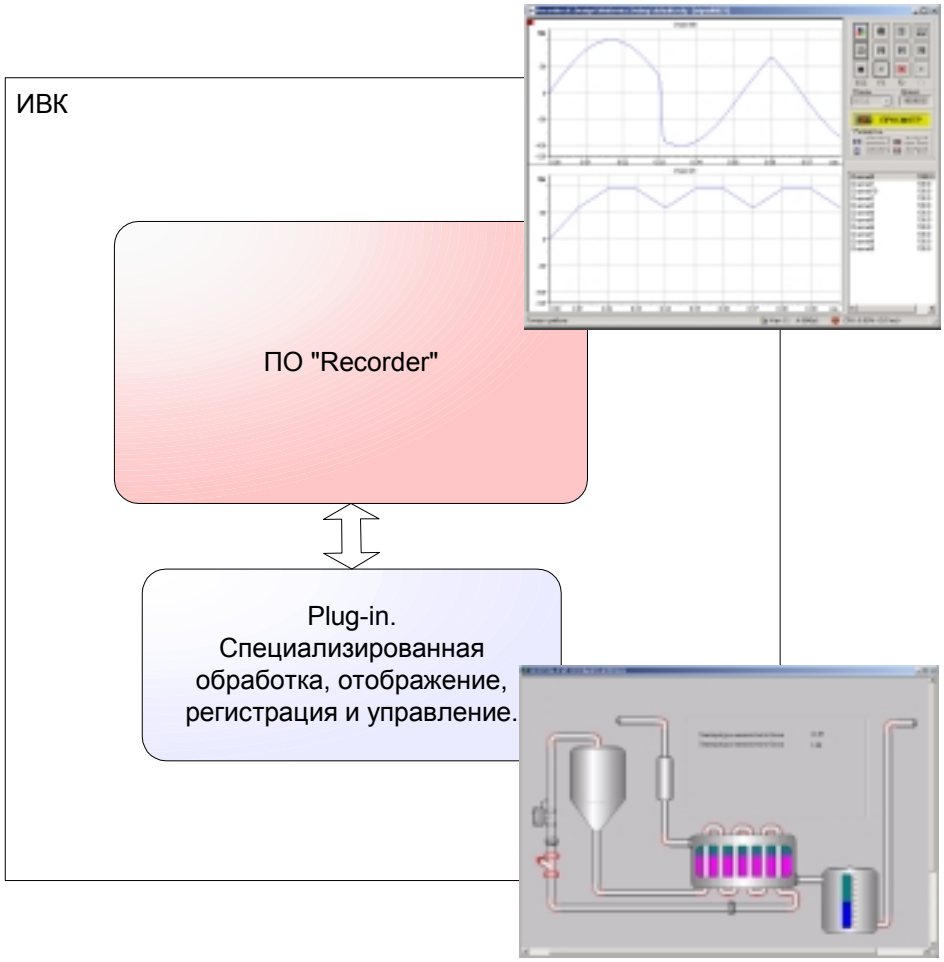


Рисунок 3. Использование plug-in`a специализированного отображения

Для ПО «Recorder» реализован модуль plug-in`a, с именем «COMCtrlPlugin», который предоставляет группу COM объектов и интерфейсов для управления изменения на-

стройки ПО «Recorder». COM интерфейсы управления настройкой позволяют просматривать конфигурацию ПО «Recorder», менять её удаленно (из другого компьютера).

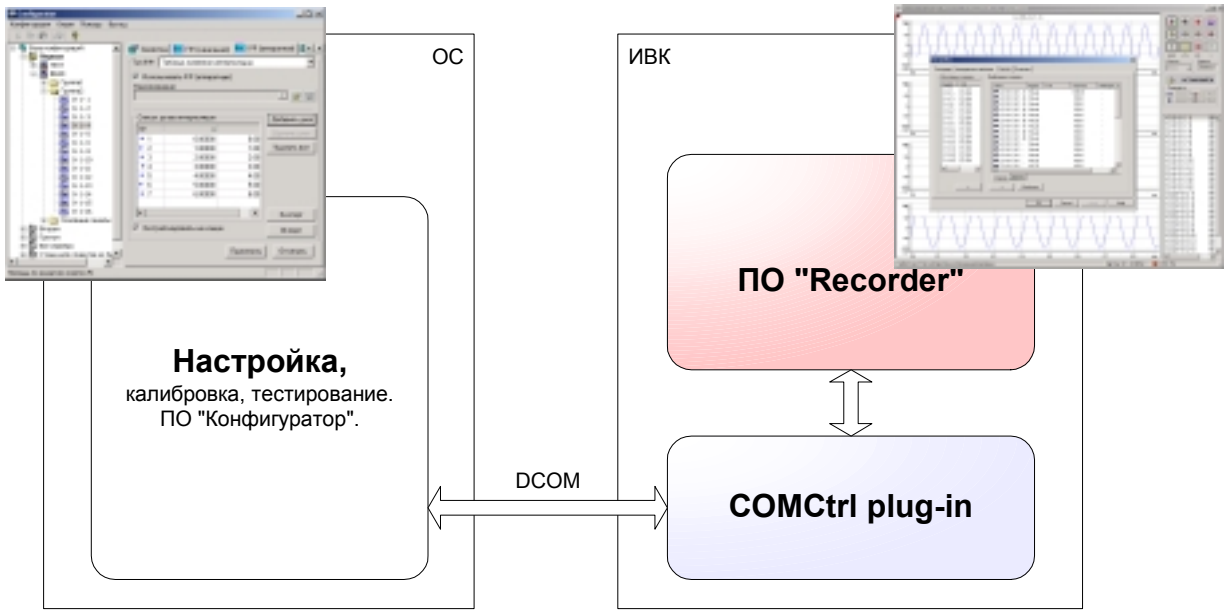


Рисунок 4. Удаленная настройка ПО «Recorder»

На рисунке 4 схематически показано взаимодействие ПО «Recorder» и ПО «Конфигуратор». ПО «Конфигуратор» реализует функции централизованного управления и настройки распределенной измерительной информационной системы, построенной на ПО «Recorder».

Для ПО «Recorder» реализован модуль plug-in`a, с именем «OPCPlugin», который реализу-

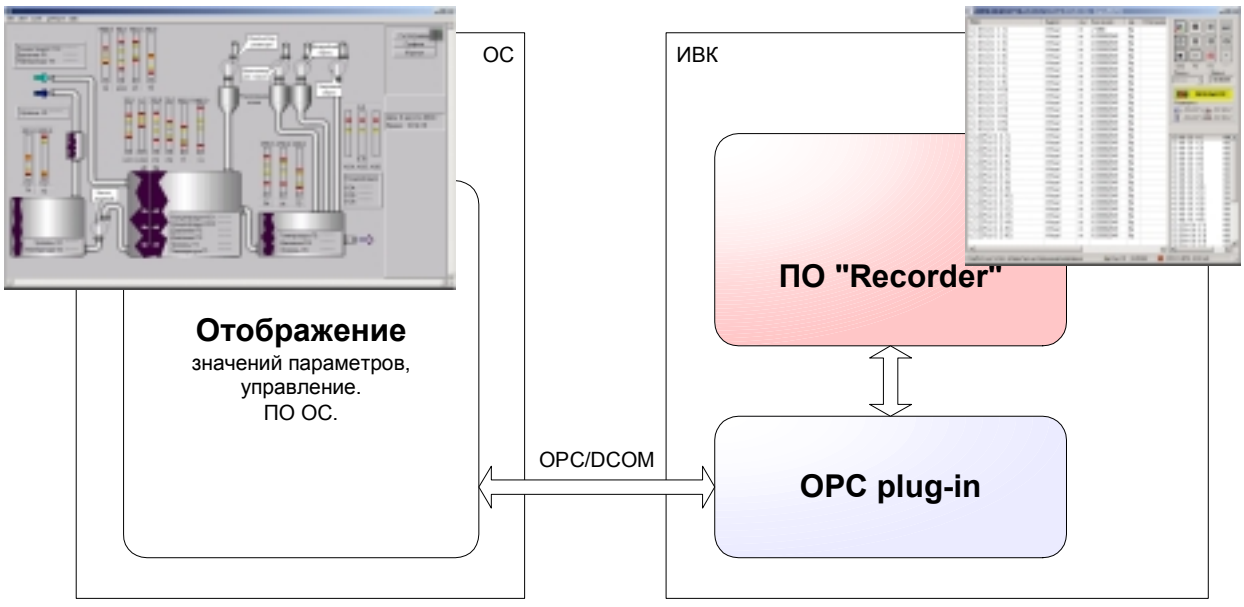


Рисунок 5. Получение и отображение измеренных данных в распределенной системе

ет протокол OPC – универсальный протокол для доступа к данным. Протокол OPC является стандартом в области программных систем АСУТП.

Рисунок 5 иллюстрирует взаимодействие ПО «Recorder» и OPC клиента. В качестве OPC клиента может выступать любое специализированное ПО либо SCADA система.

3 Программный интерфейс PluginAPI

Программный интерфейс PluginAPI предназначен для предоставления возможности разработки модулей plug-in`ов, расширяющих функциональность ПО «Recorder». Программный интерфейс PluginAPI описывает ряд функций, COM интерфейсы и COM объекты, которые позволяют ядру ПО «Recorder» взаимодействовать с plug-in`ами, а модулям plug-in`ов предоставляют доступ к внутренним объектам ядра ПО «Recorder».

3.1 Ядро ПО «Recorder»

3.1.1 Структура ядра ПО «Recorder»

Обобщенная структура ПО «Recorder» изображена на рисунке 6.

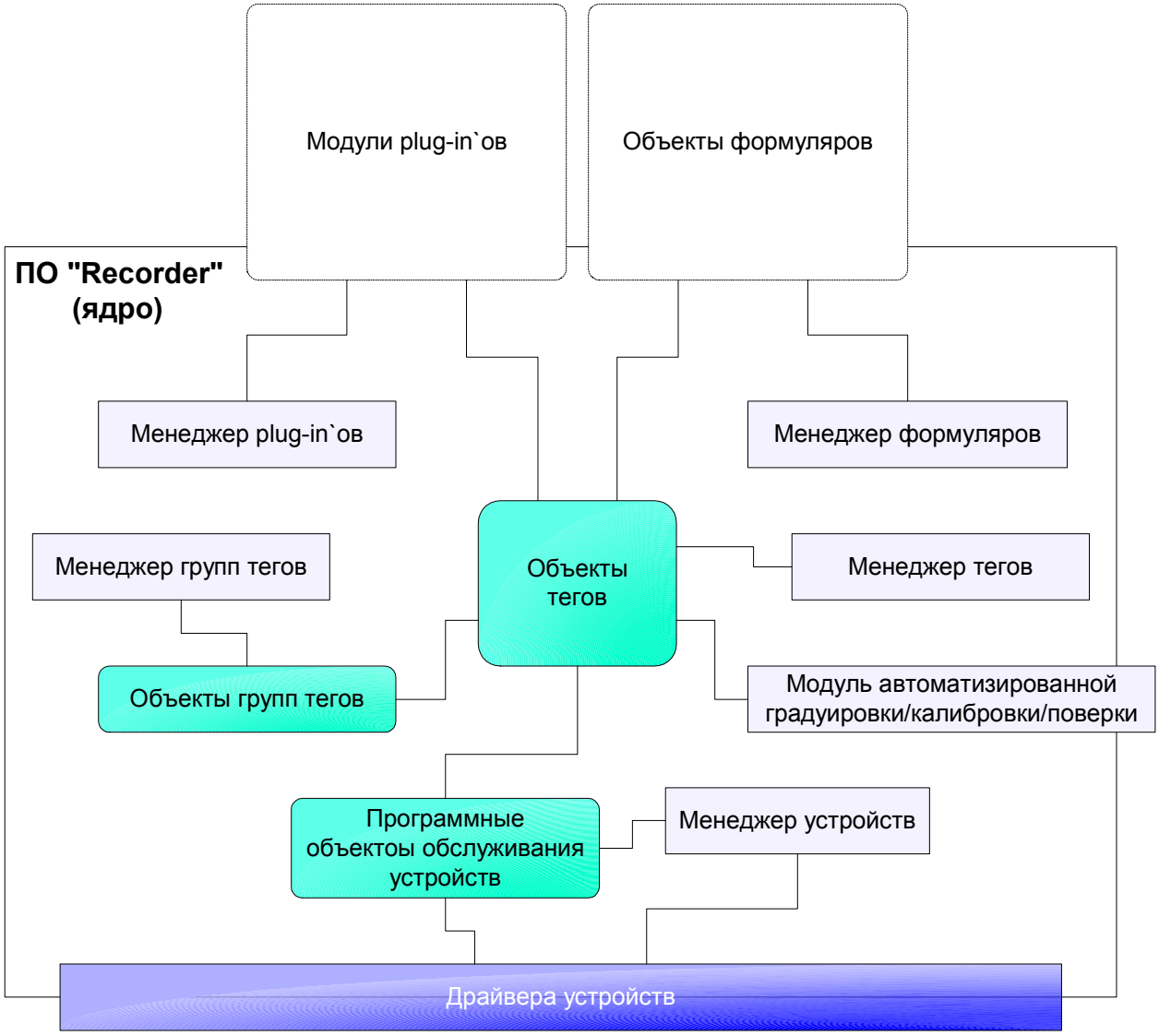


Рисунок 6. Структура ядра ПО «Recorder», взаимодействие объектов

Как видно из рисунка 6, ПО «Recorder» содержит функциональный блок менеджера устройств. Менеджер устройств взаимодействует с драйверами, создает программные объекты для обслуживания отдельных устройств (модулей, измерительных плат…). Объекты обслуживания устройств управляют аппаратными устройствами по средствам функций драйверов. С каждым физическим каналом измерительного и исполнительного устройства ассоциируется отдельный тег. Тег – это программный объект, при помощи ко-

торого осуществляется управление физическим каналом, который выполняет хранение данных канала в буфере. Управление тегами осуществляет функциональный блок менеджера тегов, который выполняет: создание и удаление тегов. Для единой установки свойств тегов используются объекты групп тегов. Объектами групп тегов управляет менеджер групп. ПО «Recorder» содержит менеджер plug-in`ов, который выполняет управление модулями plug-in`ов, выполняет загрузку модуля, активизацию, используется для передачи сообщений plug-in`ам и т.п. Модули plug-in`ов имеют доступ к объектам тегов, к объектам групп, к объектам устройств, к менеджеру тегов, к менеджеру групп. Модули plug-in`ов имеют возможность создавать собственные объекты формуляров, для отображения данных из буферов тегов. Объекты формуляров регистрируются у менеджера формуляров. Менеджер формуляров позволяет ПО «Recorder» управлять формулярами, активизировать их, показывать, скрывать, передавать сообщения.

ПО «Recorder» выполняет обработку и регистрацию измеренных данных, объекты, реализующие данную функциональность, отображены на рисунке 7.

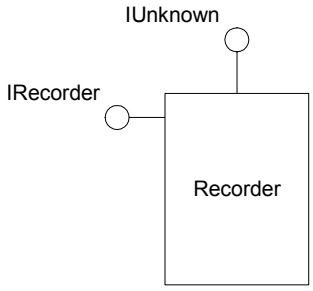
Рисунок 7. Структура ядра ПО «Recorder», обработка и регистрация данных

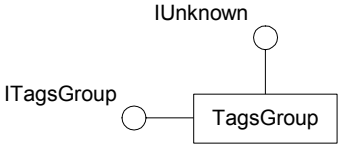
Данные, полученные тегами, обрабатываются при помощи градуировочных функций, записываются в буферы тегов. При получении новых данных в буферах тегов, выполняется регистрация – сохранение данных в файлы.

3.1.2 Объекты ядра

Перечень программных объектов ядра ПО «Recorder», доступ к которым предоставляется интерфейсом PluginAPI приведен ниже.

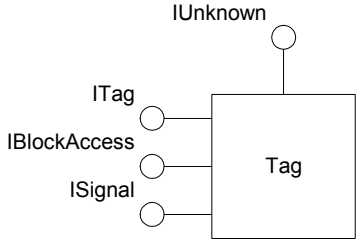
Объект Recorder. Реализует интерфейс IRecorder. Интерфейс IRecorder позволяет получить доступ к целой группе объектов ядра ПО «Recorder». Функции интерфейса IRecorder служат для управления режимом работы, для смены параметров режима измерения, для получения доступа к менеджеру тегов, для получения доступа к менеджеру групп тегов, для получения доступа к менеджеру формуляров, для получения доступа к менеджеру устройств.

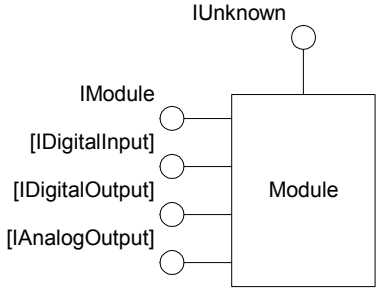




Объект TagsGroup. Данный объект является объектом группы тегов ядра, реализует интерфейс ITagsGroup. Интерфейс ITagsGroup предназначен для установки параметров группы и получения списка тегов, которые принадлежат группе.

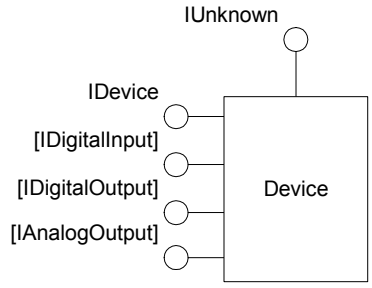
Объект Tag. Данный объект является объектом тега ядра. Объект Tag реализует интерфейсы ITag, IBlockAccess и ISignal. Интерфейс ITag предназначен для получения информации о теге. Интерфейсы IBlockAccess и ISignal предоставляют доустп к буферу данных тега.

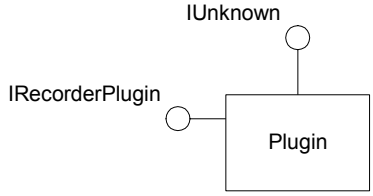




Объект Module. При помощи данного объекта предоставляется доступ к объекту обслуживания устройства ядра. Объект Module реализует интерфейсы IModule, IDigitalInput, IDigitalOutput, IAnalogOutput. Интерфейс IModule позволяет получить свойства устройств. Интерфейсы IDigitalInput, IDigitalOutput, IAnalogOutput реализованы не для каждого устройства, эти интерфейсы позволяют производить цифровой ввод/вывод и аналоговый вывод.

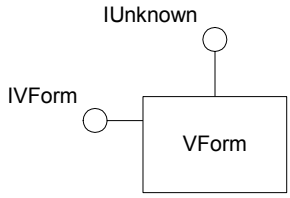
Объект Device. При помощи данного объекта предоставляется доступ к объекту обслуживания устройства ядра. Объект Device реализует интерфейсы IDevice, IDigitalInput, IDigitalOutput, IAnalogOutput. Интерфейс IDevice позволяет получить свойства устройств. Интерфейсы IDigitalInput, IDigitalOutput, IAnalogOutput реализованы не для каждого устройства, эти интерфейсы позволяют производить цифровой ввод/вывод и аналоговый вывод.

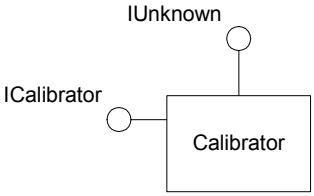




Объект Plugin. Этот объект должен быть реализован в модуле plug-in`а. При помощи интерфейса IRecorderPlugin ядро ПО «Recorder» выполняет управление plug-in`ом.

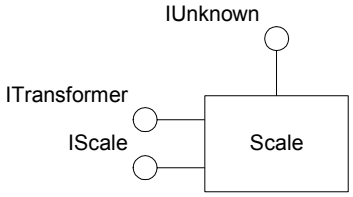
Объект VForm. Это объект может быть реализован в модуле plug-in`a. При помощи интерфейса IVForm ядро ПО «Recorder» производит управление формулярами отображения.

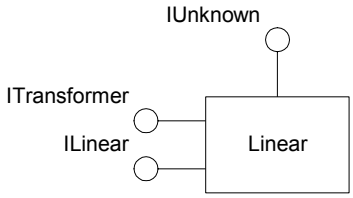




Объект Calibrator. Данный объект используется ядром для выполнения автоматизированной калибровки, градуировки, поверки.

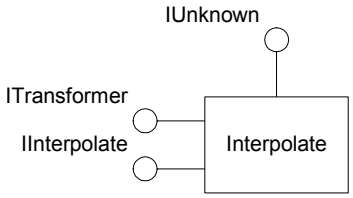
Объект Scale. Это объект градуировочной функции «Масштабный коэффициент».

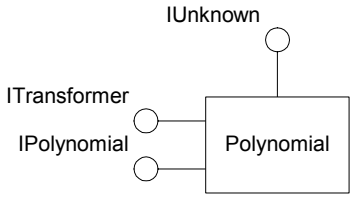




Объект Linear. Это объект градуировочной функции «Линейная функция».

Объект Interpolate. Это объект градуировочной функции «Таблица линейной интерполяции».





Объект Polynomial. Это объект градуировочной функции «Полином n-го порядка».

Связи объектов отображены на рисунке в приложении В.

3.1.3 Режимы работы

ПО «Recorder» в процессе работы пребывает в одном из режимов:

1. Режим «Готовность», или «Останов». Этот режим устанавливается сразу после запуска приложения или после входа из любого другого режима.
2. Режим «Измерение». В этом режиме выполняется сбор измеренных данных, обработка и отображение.
3. Режим «Запись», или «Регистрация». В этом режиме выполняется сбор измеренных данных, отображение и сохранение в файлы.

4. Режим «Настройка». В этом режиме производится изменение настройки ПО «Recorder». Изменение настройки может выполняться как при помощи диалогов самого ПО «Recorder», так и каким-либо из plug-in`ов.

Смена режима работы ПО «Recorder» выполняется по команде оператора, либо по запросу любого plug-in`а.

Переход от режима «Готовность» к режиму «Измерение» или «Запись» может быть осуществлен только в том случае если настройка аппаратных измерительных и исполнительных устройств выполнена корректно.

Изменение настройки – добавление тегов, изменение параметров тегов и т.п. должно выполняться plug-in`ами только в режиме «Настройки» и может выполняться только тем plug-in`ом, который инициировал режим «Настройки». ПО «Recorder» выполняет контроль изменения состояний, для того, чтобы исключить одновременное изменение настройки двумя или несколькими plug-in`ами.

3.1.4 Потоки выполнения программы

ПО «Recorder» является многопоточным приложением, в рамках приложения используется три независимых потока.

Один поток создается драйверами устройств, в контексте этого потока выполняется получение измеренных данных от устройств и передача этих данных в теги.

Рисунок 8. Потоки ПО «Recorder»

В ядре ПО «Recorder» создан отдельный поток, который используется для обращения к драйверам устройств, для работы с модулями plug-in`ов.

Для оконного интерфейса создается отдельный поток, он служит для обработки действий оператора, в контексте этого потока выполняется отображение данных.

Обращение к модулям plug-in`ов выполняется в контексте каждого из трех потоков. Соответственно plug-in должен быть разработан таким образом, чтобы его работа в многопоточной среде была безопасна. В модуле plug-in`а должна быть реализована защита, если в нем выполняется обработка сообщений или вызовов из более чем из одного потока.

Загрузка библиотеки plug-in`а выполняется в контексте потока ядра. В этом же потоке выполняется создание, инициализация и запуск экземпляра plug-in`а. Сообщения об изменении (обновлении) данных в тегах передаются экземплярам plug-in`ов в контексте

потока драйверов устройств. Сообщения об изменении режима работы, инициированного оператором, передаются в plug-in`ы в контексте оконного потока.

Существует ряд сообщений, которые могут генерироваться в результате каких-либо действий plug-in`ов. К примеру, это могут быть сообщения о переходе в режим настройки и выходе из него. Подобные сообщения генерируются в контексте того потока, в котором были вызваны функции перехода в режим настройки и соответственно выход из режима настройки.

3.1.5 Буфер данных тега

Каждый тег ПО «Recorder» обладает внутренним буфером для хранения данных. Данные в этот буфер могут записываться plug-in`ами, либо могут поступать от драйверов аппаратных устройств. Буферы тегов являются кольцевыми и двойными.

Аппаратные измерительные устройства ПО «Recorder» настраивает таким образом, что данные от них поступают порциями – блоками. Блок это массив измеренных данных. Размер блока данных одного канала, и соответственно тега, зависит от частоты дискретизации канала и периода обновления данных. Частота дискретизации устанавливается для каждого тега индивидуально, и соответственно для канала. Период обновления – это тот период, с которым ПО «Recorder» получает блоки данных от драйверов. Период обновления данных указывается в настройке ПО «Recorder». Для того, чтобы получить размер одного блока необходимо частоту дискретизации умножить на время обновления.

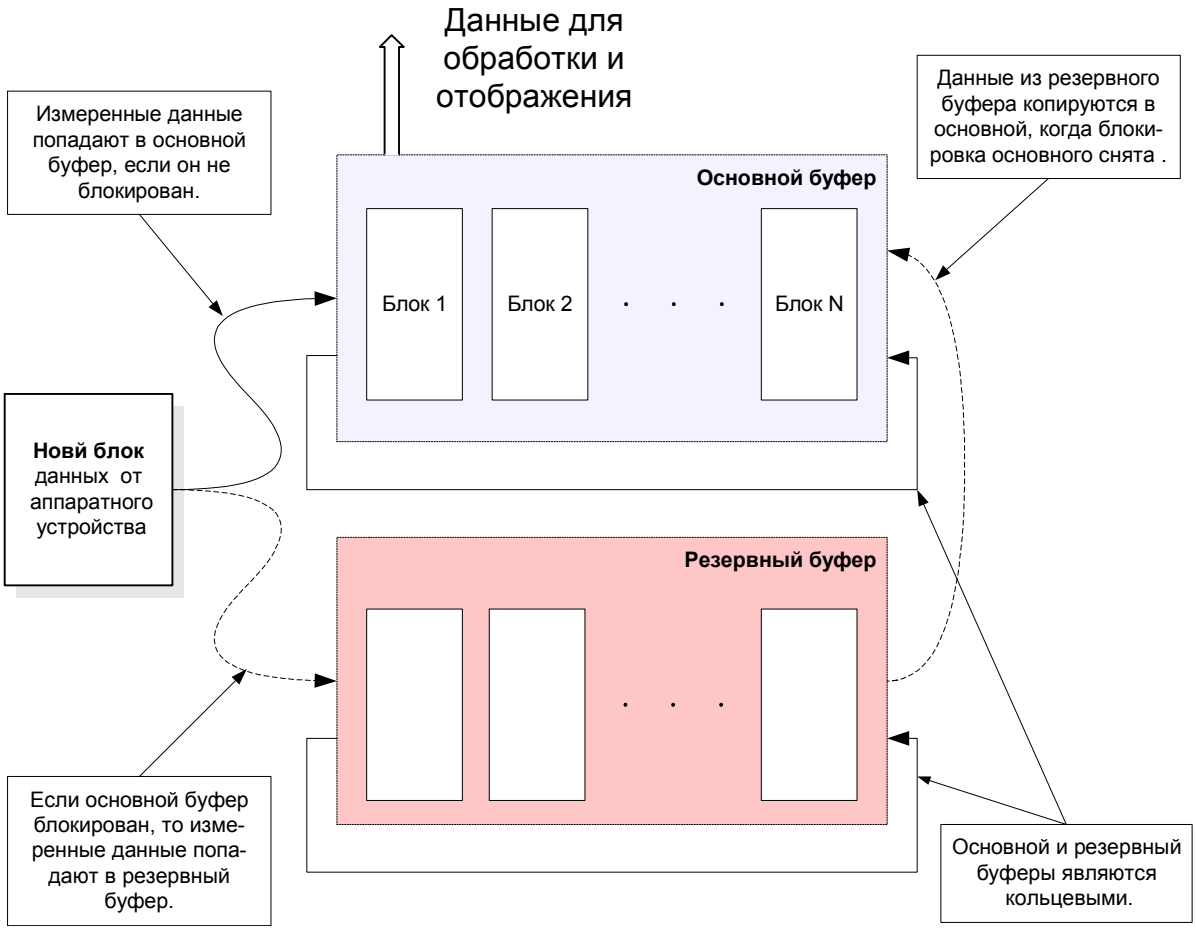


Рисунок 9. Механизм работы двойного буфера тега

В Recorder`е используется двойная буферизация измеренных данных. Двойная буферизация необходима потому, что Recorder является многопоточным приложением, дан-

ные тегов могут использоваться различными plug-in`ами и каждый plug-in может работать в собственном потоке.

Для каждого тега определен основной и резервный буферы данных. Измеренные данные, от аппаратных драйверов, поступают в основной буфер тега. Из основного буфера данные извлекаются любым plug-in`ом. В тот момент, когда plug-in читает данные из буфера тега, данные в буфере не должны меняться. Для того, чтобы монополизировать буфер на время операции чтения, plug-in использует специальный механизм блокировки буфера. Когда plug-in завершит операцию чтения данных, буфер должен быть разблокирован. В то время когда один из plug-in`ов выполняет чтение и основной буфер блокирован, могут поступить данные от измерительного устройства. Чтобы эти данные не были потеряны, они записываются в резервный буфер. После того как основной буфер будет разблокирован данные из резервного буфера перепишутся в основной.

Для управления буферами используется специальный контроллер двойного буфера, он выполняет блокирование и разблокирование основного буфера, по запросам plug-in`ов, а также автоматически переписывает данные после разблокирования, если таковые есть, из резервного буфера в основной.

Оба буфера и основной и резервный являются кольцевыми. Каждый «новый» поступивший блок с данными «перетирает» самый «старый блок». Размер буферов зависит от периода отображения данных. Период отображения указывается при настройке ПО «Recorder». Размер основного буфера устанавливается таким, чтобы буфер вмещал данные на время одного периода отображения.

Механизм работы двойного буфера тега схематично отображен на рисунке 9. Управление буфером тега, получение данных, запись данных, блокировка, разблокирование производится при помощи интерфейсов тега – объекта ядра ПО «Recorder».

3.1.6 Журнал отладочных событий.

Журнал отладочных событий создан для облегчения разработки и отладки plu-in`ов – для диагностики ошибок. Физически журнал хранится в файле. Файл расположен в каталоге “C:\var\”, называется файл: “recorder.log”.

ПО «Recorder» автоматически фиксирует в журнале основные внутренние события, такие как запуск, инициализация аппаратуры, запуск plug-in`ов и т.п.

Для добавления сообщений в журнал из plug-in`а можно использовать метод IRecorder::LogMessage().

Для просмотра событий, во время работы Recorder`а, создано специальное окно, которое отображает весь список событий. По умолчанию, при запуске Recorder`а окно журнала отладочных событий скрыто, а становится видимым при нажатии символа “~” в главном окне. Содержимое окна журнала обновляется при изменении файла.

3.1.7 Использование COM технологий

Объекты ядра ПО «Recorder», объекты, описанные в пп. 3.1.2, не являются полноценными COM объектами. Программный интерфейс PluginAPI разработан с некоторыми отступлениями от спецификации COM. Особенность интерфейса PluginAPI, в том, что в нем не используются принципы управления памяти принятые для COM. Интерфейс PluginAPI не использует менеджер памяти COM. Данное отступление от стандарта имеет целью повышение общей производительности ПО «Recorder», служит для минимизации накладных расходов при обмене данными между объектами ядра ПО «Recorder» и plug-in`ами.

Объекты ядра Recorder, TagsGroup, Tag, Module создаются самим приложением ПО «Recorder», и не могут быть созданы с использованием системных функций CoCreateIn-

stance(), CoGetClassObject(), и т.п. Экземпляры plug-in`ов получают ссылку на объект Recorder при инициализации. Модули plug-in`ов могут не быть зарегистрированы в реестре как COM сервера, потому, что ПО «Recorder» загружает их без использования функций CoCreateInstance(), CoGetClassObject(), и т.п.

Объекты Calibrator, Scale, Linear, Interpolate, Polynomial регистрируются в реестре, являются COM серверами, но могут работать только в рамках одного процесса и одного COM апартамента. Создание этих объектов должно выполняться при помощи системных функций CoCreateInstance(), CoGetClassObject() и т.п.

Все объекты ядра ПО «Recorder» и используемые яром, описанные в спецификации PluginAPI, реализуют интерфейс IUnknown. Это означает, что они реализуют счетчик ссылок, некоторые только в отладочных целях, и функцию получения интерфейса по идентификатору.

3.2 Разработка модулей plug-in`ов

Модуль plug-in`а – это динамически линкуемая библиотека DLL. Подобный модуль можно создать при помощи инструмента разработки программ, такого как Borland® Delphi™, Borland® C++ Builder™, Microsoft® Visual C++.

3.2.1 Библиотека plug-in`а

Динамически линкуемая библиотека plug-in`а должна реализовывать и экспортировать следующие функции:

1. GetPluginType() - функция получения типа plug-in`а, в данный момент используется только один типа plug-in`а;
2. CreatePluginClass() – функция создания экземпляра класса plug-in`а;
3. GetPluginDescription() – функция получения строки описания модуля plug-in`а, эта функция необходима для того, чтобы ПО «Recorder» могло отобразить информацию о plug-in`е оператору;
4. DestroyPluginClass() – функция удаления экземпляра класса plug-in`а;
5. GetPluginInfo() – функция получения расширенной информации о модуле plug-in`а, расширенная информация включает наименования plug-in`а, строку описания, наименование фирмы разработчика и номер версии.

В библиотеке должен быть описан и реализован класс plug-in`а.

3.2.2 Класс plug-in`а

Класс plug-in`а может реализовывать любую необходимую функциональность: получение данных, обработку, вывод, регистрацию и т.п. Для взаимодействия с ядром ПО «Recorder» класс может использовать объекты Recorder, TagsGroup, Tag и должен реализовывать интерфейс IRecorderPlugin. Если класс plug-in`а содержит собственный формуляр для отображения данных, то он может дополнительно реализовать интерфейс IVForm.

Создание экземпляра класса plug-in`а, удаление, управление временем его жизни, выполняется при помощи экспортируемых функций CreatePluginClass() и DestroyPluginClass().

3.2.3 Загрузка, запуск и завершение работы Plug-in`ов

ПО «Recorder» выполняет загрузку библиотек plug-in`ов, которые указаны в его конфигурационном файле, загрузка библиотек, создание экземпляров plug-in`ов производится следующим образом:

1. Производится загрузка библиотеки plug-in`а.

2. Производится вызов функции GetPluginType(), для проверки того, что библиотека реализует объект plug-in именно для ПО «Recorder».
3. Создается экземпляр plug-in`а. Объект plug-in`а создается при помощи функции библиотеки CreatePluginClass(), ПО «Recorder» получает ссылку на интерфейс, реализуемый plug-in`ом.
4. ПО «Recorder» вызывает для объекта plug-in`а метод IRecorderPlugin::create(). Параметром метода create() в plug-in передается ссылка на объект Recorder. В этом методе plug-in может инициализировать собственные данные и произвести выделение необходимых системных ресурсов. Если инициализация plug-in`а по какой-то причине завершилась ошибкой, то метод create() должен вернуть «false». Следующие шаги загрузки выполняются, только если метод create() вернул «true».
5. ПО «Recorder» вызывает для объекта plug-in`а метод IRecorderPlugin::config(). В этом методе в plug-in`е может быть реализовано чтение собственной конфигурации из файлов и настройка.
6. Recorder вызывает для объекта plug-in`а метод IRecorderPlugin::execut(). Вызов этого метода производится, когда ПО «Recorder» полностью завершило загрузку и собственный запуск (запуск приложения), все plug-in`ы созданы и настроены. В plug-in`е в этом методе производится запуск – это может быть, к примеру, создание собственного рабочего потока либо отображение диалоговых окон и т.п.

После 6-го шага plug-in`ы запущены, ПО «Recorder» в процессе работы передает в plug-in`ы сообщения, при помощи метода IRecorderPlugin::notify().

Завершение работы Recorder`а происходит следующим образом:

1. Для каждого plug-in`а вызывается метод IRecorderPlugin::canclose(). В этом методе в plug-in`е должна производится проверка возможности завершения работы. Если по каким-то причинам в данный момент работу plug-in`а завершить невозможно, то метод canclose() должен вернуть «false». Если закрытие plug-in`а возможно, то метод должен вернуть «true». О том, что какой-либо plug-in не позволяет завершить работу сообщается оператору. Оператор имеет возможность принудительно прервать работу plug-in`а.
2. Если все plug-in`ы допускают закрытие, либо оператор принудительно завершает работу, то ПО «Recorder» вызывает метод plug-in`а IRecorderPlugin::close(). В методе close() в plug-in`е необходимо освободить все выделенные ранее ресурсы, если plug-in создавал собственные потоки, то выполнение потоков необходимо завершить.
3. После закрытия plug-in`а Recorder производит его удаление при помощи функции DestroyPluginClass() (функция библиотеки plug-in`а).
4. Производится выгрузка библиотеки plug-in`а.

3.2.4 Режим ПО «Recorder», изменение режима

Текущий режим работы ПО «Recorder» хранит во внутренней переменной, режим кодируется при помощи битовых флагов состояния. Получить значение переменной можно при помощи метода IRecorder::GetState(). ПО «Recorder» оповещает все plug-in`ы об изменении режима.

Переход от режима «Готовность» к режиму «Измерение» производится следующим образом:

1. Перед запуском режима «Измерение» ПО «Recorder» рассылает всем plug-in`ам сообщение с кодом PN_BEFORE_RCSTART.
2. Производится подготовка и запуск измерения. Если запуск осуществлен успешно, то выполняется шаг 3, иначе шаг 4.
3. После запуска выполняется оповещение всех plug-in`ов об успешном запуске. Plug-in`ам рассылается сообщение с кодом PN_RCSTART. Производится измерение.
4. Если процесс запуска не выполнен, к примеру, по причине ошибок в настройке, то все plug-in`ы получают сообщение с кодом PN_ABORT_RCSTART. Запуск отменен.

Переход от режима «Измерение» к режиму «Готовность» выполняется следующим образом:

1. Все plug-in`ы получают сообщение о необходимости завершения измерения - сообщение с кодом PN_BEFORE_RCSTOP.
2. Выполняется останов процесса измерения.
3. Все plug-in`ы оповещаются о останове – сообщение PN_RCSTOP.

Модули plug-in`ов могут инициировать изменение режима работы ПО «Recorder» при помощи команд RCN_VIEW, RCN_REC, RCN_STOP. Передача команд в ПО «Recorder» выполняется при помощи вызова метода IRecorder::Notify().

О переходе в режим «Настройка» ПО «Recorder» оповещает все plug-in`ы при помощи сообщения с кодом PN_ENTERRCCONFIG, при выходе из режима все plug-in`ы получают сообщение с кодом PN_LEAVERCCONFIG.
Запрос на переход к режиму «Настройка» и выход из него, plug-in может выполнить при помощи методов IRecorder::EnterConfigMode() и IRecorder::LeaveConfigMode() соответственно.

3.2.5 Доступ к объектам ядра ПО «Recorder»

Объект plug-in`а получает ссылку на объект ядра Recorder в методе IRecorderPlugin::create(), при создании и инициализации. При помощи объекта Recorder и его интерфейсов plug-in может получить доступ ко всем объектам ядра.

Методы IRecorder::GetModuleByIndex() и IRecorder::GetModulesCount() позволяют получить список объектов Module. Методы IRecorder::GetIFormByName() и IRecorder::GetIFormByIndex() позволяют получить ссылки на объекты формуляров. Методы IRecorder::GetGroupByIndex() и IRecorder::GetGroupsCount() позволяют получить ссылки на объекты групп тегов TagsGroup. Методы IRecorder::GetTagByName(), IRecorder::GetTagByIndex() и IRecorder::GetTagsCount() позволяют получить список ссылок на теги.

Интерфейсы объекта TagsGroup позволяют получить список тегов, которые принадлежат группе, для этого используются функции ITagsGroup::GetTagsCount(), ITagsGroup::GetTagByIndex(), ITagsGroup::GetTagByName().

Методы интерфейсов тегов позволяют получить ссылки на объекты градуировочных функций, для этого используется метод ITag::GetProperty().

Таким образом, методы объекта Recorder позволяют получить доступ к любому объекту ядра ПО «Recorder».

3.2.6 Plug-in`ы с формулярами отображения

Одна из целей разработки модулей plug-in`ов – это создание специализированного формуляра для отображения измеренных либо обработанных данных. Формуляр – это ок-

но, в котором для оператора, отображаются различные рисунки, гистограммы, графики, таблицы и т.п. Формуляр может быть реализован как независимое окно, а может являться частью главного окна ПО «Recorder».

Реализация формуляра, как независимого окна, может быть необходима, к примеру, в том случае, если требуется, чтобы оператор не имел непосредственного доступа к главному окну ПО «Recorder». Независимое окно формуляра, одно или несколько, могут создаваться plug-in`ом, удаляться, отображаться, скрываться тогда, когда это необходимо plug-in`у.

Если необходимо, чтобы окно формуляра присутствовало в списке формуляров ПО «Recorder», если необходимо иметь доступ к формуляру из главного окна, то plug-in должен:

1. реализовать интерфейс IVForm,
2. регистрировать себя (свой формуляр, один или несколько) в менеджере формуляров ядра ПО «Recorder».
3. Установить собственному окну формуляра главное окно ПО «Recorder» как родительское окно.

Регистрация формуляра в ПО «Recorder» выполняется после запуска приложения и инициализации plug-in`а, при помощи функции IRecorder:: RegisterIForm(). После регистрации, управление формуляром принимает на себя менеджер формуляров. Менеджер передает в формуляр, при помощи интерфейса IVForm, сообщения для отображения формуляра, сокрытия формуляра, изменения размера формуляра, обновления отображаемых данных на формуляре и т.п. Окно формуляра должно быть настроено как дочернее окно без рамки. Управление формуляром выполняется менеджером по запросам оператора и по внутренним событиям ПО «Recorder».

3.2.7 Событие обновления данных

В режиме «Измерение» или «Регистрация» от драйверов устройств периодически поступают порции измеренных данных. Эти порции – блоки попадают в буферы тегов. При получении очередной порции измеренных данных ПО «Recorder» рассылает plug-in`ам оповещения. Эти оповещения plug-in`ы могут использовать для обработки и отображения поступивших вновь данных. Оповещения передаются следующим образом: ПО «Recorder» вызывает метод IRecorderPlugin::notify(), с кодом события PN_UPDATEDATA. Это событие рассылается в контексте потока драйверов устройств (см. пп. 3.1.4.).

Класс plug-in`а должен обрабатывать сообщение с кодом PN_UPDATEDATA только в том случае, если для работы plug-in`а необходим полный объем измеренных данных, если обработка измеренных данных производится в реальном масштабе времени. Класс plug-in`а может не обрабатывать сообщение с кодом PN_UPDATEDATA, если для его работы необходимы измеренные данные реже, чем они поступают от измерительных устройств.

3.2.8 Чтение измеренных данных, блочный доступ

Объект ядра тег реализует несколько интерфейсов. Для получения из буфера тега измеренных данных блоками используется интерфейс IBlockAccess. При помощи этого интерфейса plug-in имеет возможность получать полный объем измеренных данных, тот объем, который поступает непосредственно от драйверов устройств. Использование интерфейса IBlockAccess целесообразно в режиме «Измерение» либо «Регистрация».

Интерфейс IBlockAccess позволяет работать непосредственно с основным буфером тега, описывает функции:

1. функция блокирования буфера тега;

2. функция снятия блокировки буфера;
3. функция получения количества блоков в буфере;
4. функция получения общего количества блоков, которые прошли через буфер, подсчет количества блоков производится от начала режима «Измерение» или «Регистрация»;
5. функция определения размера одного блока;
6. группа функций для чтения блока измеренных данных из буфера, интерфейс описывает несколько функций чтения измеренных данных – для чтения измеренных данных как массива действительных чисел либо как массива целых чисел.

Класс plug-in`а может работать с буфером тега в любой момент времени, либо непосредственно в методе обработчике сообщения с кодом PN_UPDATEDATA. Получение сообщения с кодом PN_UPDATEDATA гарантирует, что хотя бы в один тег из всего списка тегов ПО «Recorder» содержит в буфере блок еще не обработанных данных.

Для определения того, есть ли в буфере конкретного тега не обработанный plug-in`ом блок, необходимо использовать счетчик общего количества обработанных блоков. В коде plug-in`а следует запоминать общее количество блоков прошедших через буфер.

Типовой алгоритм чтения блоков измеренных данных описан ниже.

1. Вызов метода IBlockAccess::LockVector(); – этот вызов необходим для блокирования буфера тега, на все время работы с данными буфера.
2. Вызов метода IBlockAccess::GetBlocksCount(), получение количества блоков, которые в данный момент находятся в буфере. Если буфер пуст, то перейти к пп. 6.
3. Вызов метода IBlockAccess::GetReadyBlocksCount() – получение общего количества блоков, которые прошли через буфер. Если это значение изменилось, для данного тега, от момента предыдущего обращения к буферу, значит, буфер содержит блок с измеренными данными, которые plug-in еще не обработал. Если буфер не содержит «новых» блоков, то переход к пп. 6.
4. Вызов метода IBlockAccess::GetBlocksSize() – для получения размера блока. Эта процедура может быть выполнена в процессе подготовки plug-in`а к измерению.
5. Вызов одного из методов GetVectorR4(),GetVectorI2(),GetVectorR8() для чтения данных блока. В зависимости от количества необходимых блоков методы чтения вызываются один или несколько раз.
6. Вызов метода IBlockAccess::UnlockVector(); - необходим для разблокирования буфера тега.

Программно в plug-in`е можно получить доступ к любому из блоков основного буфера тега, номер требуемого блока указывается параметром при вызове методов чтения. Общее количество блоков в буфере можно узнать при вызове метода IBlockAccess::GetBlocksCount().

3.2.9 Чтение измеренных данных, синхронное чтение

Существует способ чтения измерительных данных, минуя буфер тега – это синхронное чтение блока. Такое чтение можно производить только когда ПО «Recorder» находится в режиме «Готовность».

Синхронное чтение может быть произведено при помощи одного из двух методов: IRecorder::MultiTagSynchroReadDataBlock() либо ITag::SynchroReadDataBlock().

Методом ITag::SynchroReadDataBlock() выполняется операция чтения измеренных данных у одного тега. Метод IRecorder::MultiTagSynchroReadDataBlock выполняет синхронное одновременное чтение измеренных данных у нескольких тегов.

Эти методы обращаются непосредственно к драйверу аппаратного устройства, при помощи функций драйвера, производят чтение блоков данных. Функции драйвера выполняют подготовку и запуск режима измерения для аппаратного устройства, ожидают, пока требуемое количество данных не будет получено от устройства, далее режим измерения останавливается, и функции драйвера завершают свою работу. То есть выполнение функций синхронного чтения длится столько времени, сколько занимает процесс измерения и сбора необходимого объема данных.

Метод синхронного чтения измеренных данных может применяться при калибровке или поверке.

3.2.10 Обработка данных ПО «Recorder»

ПО «Recorder» может автоматически, в зависимости от настройки, выполнять обработку каждого блока измеренных данных. В процессе обработки измеренных данных может быть выполнена одна или более оценок:

1. расчет среднего арифметического;
2. расчет среднеквадратического значения;
3. расчет среднеквадратического отклонения;
4. расчет значения амплитуды;
5. расчет значения размаха.

Перечень расчетов, которые будет производить, ПО «Recorder» можно установить индивидуально для каждого тега. Программно это выполняется при помощи вызова метода ITag::SetEstimatorsMask(), либо при помощи свойства с кодом TAGPROP_ESTIMATOR.

В любой момент времени ПО «Recorder» использует одну из оценок для отображения в табличном формуляре. Тип, отображаемой в табличном формуляре оценки, можно получить и изменить при помощи свойства с кодом TAGPROP_DEFAULTESTIMATOR.

3.2.11 Чтение измеренных данных, чтение оценок

Для чтения обработанных ПО «Recorder» данных можно воспользоваться одном из трех методов:

1. ITag::GetEstimate() – простое получение оценки.
2. ITag::GetScalarEstimate() – получение оценки и кода результата выполнения операции. Этот метод позволяет обнаружить недостоверные данные.
3. Получение оценки при помощи свойства с кодом TAGPROP_ESTIMATE. Последнее допустимо, если ПО «Recorder» вычисляет только одну из оценок.

3.2.12 Запись данных в теги

Для записи данных в тег может быть использован один из трех методов:

1. ITag::PushData() – добавление одного блока данных в циклический буфер тега.
2. ITag::PushDataEx() – добавление одного блока данных, со штампом времени в буфер тега.
3. ITag::PushValue() – добавление одного значения в буфер тега.

Метод ITag::PushData() и ITag::PushDataEx() отличаются лишь тем, что ITag::PushDataEx() позволяет к записываемому блоку данных указать штамп времени. Эти методы используются для записи данных, как в виртуальные теги, так и в теги связанные с аппаратными каналами вывода.

Метод ITag::PushValue() позволяет запасть в тег одно значение, этот метод предназначен только для записи данных в теги, которые связанны с аппаратными каналами вывода.

3.2.13 Вывод данных в исполнительные устройства

Интерфейс Recorder`а дает возможность получить ссылки на «низкоуровневые» интерфейсы управления измерительными модулями. Эти интерфейсы позволяют, минуя теги, получать измерительные данные из аппаратных модулей ввода и устанавливать выходные сигналы модулей вывода.

Объект модуля реализует интерфейс IModule, при помощи этого интерфейса можно получить описание соответствующего аппаратного модуля. Если аппаратный модуль поддерживает аналоговый вывод, то из интерфейса IModule, этого модуля, можно получить ссылку на интерфейс IAnalogOutput, если модуль поддерживает цифровой вывод, то интерфейс IDigitalOutput, если модуль поддерживает цифровой ввод, то интерфейс IDigitalInput.

При помощи интерфейса IRecorder можно получить список ссылок на интерфейсы всех аппаратных модулей. Для этого используются методы IRecorder::GetModuleByIndex(), IRecorder::GetModulesCount().

3.2.14 Изменение настройки ПО «Recorder»

Модуль plug-in`а имеет возможность менять настройку ПО «Recorder», создавать теги и группы тегов, менять значения параметров тегов и групп тегов, менять параметры самого приложения и т.п. Для изменения настройки ПО «Recorder» должно находиться в режиме «Настройка». Для перехода в режим настройки и выхода из него программно в интерфейсе объекта Recorder определено два метода IRecorder::EnterConfigMode() и IRecorder::LeaveConfigMode().

Методом IRecorder::EnterConfigMode() инициирует запрос на выход в режим настройки. ПО «Recorder» допускает переход в режим настройки, только если находится в режиме «Готовность». В любой момент времени настройку может производить только один из plug-in`ов либо оператор при помощи оконного интерфейса ПО «Recorder». Метод IRecorder::EnterConfigMode() возвращает значение «true», если запрос на переход в режим настройки удовлетворен. Метод IRecorder::EnterConfigMode() возвращает значение «false», если ПО «Recorder» не может быть переведено в режим «Настройки», либо уже производится настройка другим plug-in`ов или оператором. Для того, чтобы объекты ядра ПО «Recorder» могли определить, какой из plug-in`ов инициирует переход в режима «Настройки», plug-in параметром метода IRecorder::EnterConfigMode() должен передать ссылку на свой интерфейс.

Если запрос plug-in`а на вход в режим «Настройка» был удовлетворен, то plug-in может выполнять любое изменение настройки ПО «Recorder», какое только допускается интерфейсом PluginAPI. По завершению настройки plug-in должен оповестить ПО «Recorder». Для завершения режима настройки используется метод IRecorder::LeaveConfigMode().

Как описано выше (см. пп. 3.2.4.), при смене режима ПО «Recorder» рассылает plug-in`ам оповещения. При входе и выходе из режима «Настройка» каждый plug-in получает сообщение с кодом PN_ENTERRCCONFIG и PN_LEAVERCCONFIG соответственно. Если plug-in не производил изменение настройки и получил сообщение с кодом

PN_LEAVERCCONFIG, значит какой-либо из других plug-in`ов либо оператор изменил конфигурацию ПО «Recorder». В этой ситуации plug-in должен прочесть новую настройку ПО «Recorder», ту её часть, с которой plug-in работает, и обновить собственную настройку.

3.2.15 Виртуальные теги

Основная функция виртуальных тегов состоит в том, чтобы хранить данные в памяти. Виртуальные теги обладают буфером, и всеми свойствами тега, данные виртуального тега регистрируются в файлы, но виртуальный тег не имеет связи с аппаратным каналом устройства. Данные в виртуальные теги могут записываться plug-in`ами.

Назначение виртуальных тегов:

1. Виртуальные теги могут быть использованы для генерирования тестовых данных. Использование тестовых данных вместо реальных измеренных, позволяет без аппаратного обеспечения моделировать процесс измерения и обработки.
2. Виртуальные теги могут быть использованы для хранения обработанных данных.

Создание виртуальных тегов производится при помощи функции IRecorder::CreateTag(), должно выполняться в режиме «Настройки». Параметрами метода необходимо указать имя нового тега и признак того, что новый тег будет виртуальным.

Для определения является ли тег виртуальным можно использовать функцию ITag::GetLinkState().

3.3 Программный модуль автоматизированной градуировки, калибровки, поверки

3.3.1 Градуировка, калибровка чувствительности, градуировочная, калибровочная характеристики

Схема типового процесса измерения отображена на рисунке 10. Датчик под воздействием измеряемой физической величины генерирует электрические сигналы, которые поступают на первичные преобразователи. Первичный преобразователь усиливает и фильтрует электрический сигнал и передает его в измерительный модуль. Измерительный модуль формирует цифровое представление аналогового электрического сигнала, результатом преобразования являются цифровые коды. Для преобразования цифровых кодов в значения измеренной электрической величины, используется калибровочная функция модуля (калибровка чувствительности). В результате из аппаратного модуля можно получить цифровое значение измеренного электрического сигнала, это может быть сопротивление, напряжение и т.п. Для преобразования измеренной электрической величины в физическую, используется градуировочная функция канала. Калибровочная функция модуля зависит от типа измерительного устройства (модуля). Градуировочная функция канала зависит от типа используемого датчика.

Процесс калибровки чувствительности и градуировки предназначен для определения опытным путем, калибровочной и градуировочной функций соответственно. Процесс заключается в следующем: на вход измерительной системы подаются эталонные значения измеряемых величин, с выхода снимаются соответствующие измеренные значения. На ос-

новании эталонных и полученных измеренных значений определяются параметры калибровочной и (или) градуировочной функции.

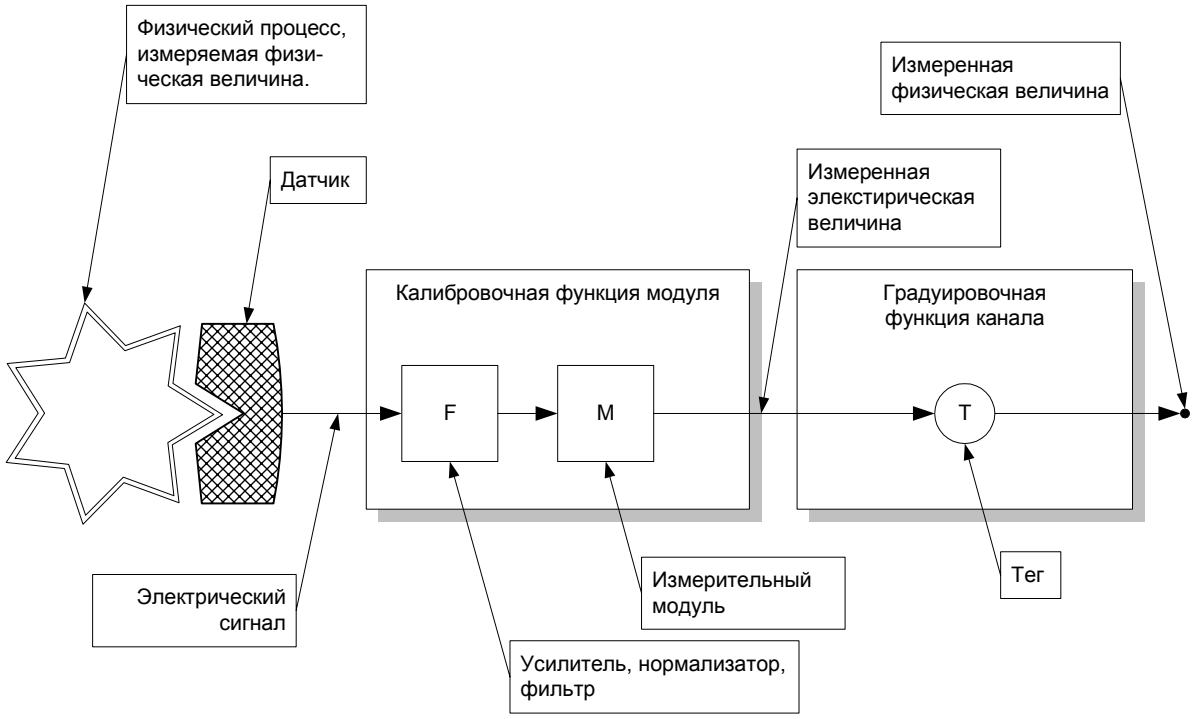


Рисунок 10. Упрощенная схема процесса измерения

Процесс поверки выполняется аналогично калибровке чувствительности или градуировке, за исключением того, что после измерений на поверке выполняется расчет погрешностей, выявление тех ошибок, которые допускает в процессе измерения измерительный тракт.

3.3.2 Типы ГХ/КХ

В ПО «Recorder» реализованы четыре основные функции:

1. функция масштабирования,
2. линейная функция,
3. таблица линейной интерполяции,
4. функция полином n-го порядка.

Все эти функции реализованы в виде COM объектов, все они реализуют один общий интерфейс ITransformer. COM объект каждой функции реализует специализированный интерфейс для собственной настройки.

При помощи методов интерфейса ITransformer можно:

1. Сохранить и прочесть настройку функции из потока в области памяти.
2. Экспортировать и импортировать параметры функции (таблицу или точки) из текстового файла.
3. Обработать измеренное значение при помощи функции.
4. Отобразить диалог редактирования параметров функции.

COM объект функции масштабирования реализует интерфейс IScale, при помощи которого можно установить и получить коэффициент масштаба. Формула, по которой функция производит вычисления:

f(x) = a*x, где f(x) – выходное значение, x – входное значение, a – масштабный коэффициент.

COM Объект линейной функции реализует интерфейс ILinear, при помощи которого можно установить и получить коэффициенты линейной функции. Формула, по которой функция производит обработку данных:

f(x) = a*x + b, где f(x) – выходное значение, x – входное значение, a, b – коэффициенты.

COM Объект функции таблица линейной интерполяции реализует интерфейс IInterpolate. Этот интерфейс позволяет формировать таблицу. Таблица содержит группу входных значений и соответствующие им выходные значения.

COM объект функции полинома n-го порядка реализует интерфейс IPolynomial, при помощи которого можно указать степень полинома и массив коэффициентов полинома. Формула, по которой функция производит вычисления:

$f(x) = \sum_{i=0}^{n} a_i \cdot x^i$, где f(x) – выходное значение, x – входное значение, a – список коэффициентов, n – степень полинома.

3.3.3 Алгоритм процесса градуировки (калибровки чувствительности), калибровки и поверки

В состав ПО «Recorder» входит модуль калибровки. Модуль калибровки реализован в виде отдельного «in-proc» COM сервера. COM объект калибровки реализует интерфейс ICalibrator.

Калибровка запускается ПО «Recorder» по запросу оператора из окна редактирования свойств тегов. Процесс калибровки выполняется следующим образом:

1. ПО «Recorder» создает COM объект калибровки.
2. ПО «Recorder» инициализирует объект калибровки – вызывает метод ICalibrator::Create().
3. ПО «Recorder» вызывает метод чтения конфигурации и передает блок данных конфигурации в объект калибровки – вызывается метод ICalibrator::Load().
4. ПО «Recorder» устанавливает параметры калибровки – устанавливает свойства при помощи метода ICalibrator::SetProperty().
5. ПО «Recorder» передает в объект калибровки список ссылок на теги, которые необходимо калибровать – вызывается метод ICalibrator::AddTag()
6. ПО «Recorder» запускает процесс калибровки – вызывается метод ICalibrator::Run().
7. Объект калибровки выполняет подготовку к измерению, выполняет ряд измерений, выполняет обработку измеренных данных и рассчитывает параметры ГФ/КФ.

Как правило, объект калибровки выполняет синхронное чтение данных, запоминает соответствующие эталонные значения, а потом производит определение необходимой функции. После того как ГФ/КФ определена, создан, для каждого тега или модуля, экземпляр COM объекта функции, ссылка на этот объект передается в тег.

Ссылка на COM объект ГФ/КФ устанавливается в тег в качестве свойства. Код идентификатора свойства TAGPROP_TARE и TAGPROP_DEVTARE для градуировочной функции канала и для калибровочной функции модуля соответственно.

Необходимо отметить, что калибровочная функция модуля присутствует всегда, когда в ней нет необходимости, её можно отключить.

Калибровать теги можно и из любого plug-in`a, особенность специализированного модуля калибровки заключается в том, что этот модуль управляется из ПО «Recorder» и вызывается из ПО «Recorder» по запросу оператора.

Процесс поверки выполняется аналогично калибровке чувствительности или градуировке.

4 Приложение. Список используемых документов

1. «Программный интерфейс DevAPI. Руководство программиста».
2. «Программное обеспечение сбора данных MR-DAQ. Руководство программиста».
3. «Программное обеспечение сбора измерительных данных «Recorder». Руководство программиста».
4. «Программное обеспечение сбора измерительных данных Recorder. Разработка plug-in`ов. Руководство программиста».
5. «Программный интерфейс настройки ПО «Recorder». Руководство программиста».
6. «Разработка ПО ОС ИИС для автоматизации испытаний».

5 Перечень используемых сокращений

COM – Component Object Model (модель компонентных объектов).
DCOM – Distributed COM, распределенная COM.
DLL – Dynamic-Link Library (динамически подсоединяемая библиотека).
ГФ (ГХ) – Градуировочная функция (градуировочная характеристика).
КФ (КХ) – Калибровочная функция (калибровочная характеристика).
ОС – операторская станция.
ПО – программное обеспечение.

A Приложение. Описание интерфейсов

A.1 Функции модуля plug-in`a

A.1.1 Функции, экспортируемые библиотекой plug-in`a

A.1.1.1 GetPluginType

int GetPluginType(void); - получение идентификатора типа plug-in`a. В настоящий момент поддерживаются plug-in`ы с идентификатором PLUGIN_CLASS. Результатом выполнения функции должен быть код типа plug-in`a.

A.1.1.2 CreatePluginClass

IRecorderPlugin * CreatePluginClass(void); - создание экземпляра класса plug-in`a. Результатом выполнения функции должна быть ссылка на интерфейс IRecorderPlugin объекта plug-in`a.

A.1.1.3 GetPluginDescription

const char* GetPluginDescription(void); - Получить строку описания plug-in`a. Результатом выполнения функции должна быть ссылка на строку с описанием plug-in`a. Строка формата языка «C», с завершающим нулевым символом.

A.1.1.4 DestroyPluginClass

DestroyPluginClass(IRecorderPlugin* a_piPlg); - Уничтожение экземпляра plug-in`a. Параметром функции является ссылка на интерфейс IRecorderPlugin удаляемого объекта.

A.1.1.5 GetPluginInfo

void GetPluginInfo(LPPLUGININFO lpPluginInfo); - Получение расширенного описания plug-in`a. Параметром функции является ссылка на структуру типа PLUGININFO, поля которой необходимо заполнить.

A.1.2 Структура PLUGININFO

В структуре PLUGININFO содержится расширенное описание plug-in`a.
Поля структуры:

№ п/п	Тип данных	Наименование	Описание
1	char[101]	name	строка наименования plug-in`a.
2	char[201]	describe	строка описания plug-in`a.
3	char[201]	vendor	строка имени разработчика plug-in`a, наименования фирмы разработчика.
4	short	version	номер версии plug-in`a.
5	short	subversion	номер подверсии plug-in`a.

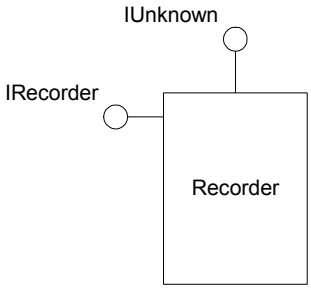
Строки в структуре должны иметь формат языка «C», то есть завершаться нулевым символом.

A.1.3 Коды типов plug-in`ов.

№	Наименование	Описание
1	PLUGIN_CLASS	Основной тип plug-in`а.

A.2 Объекты

A.2.1 Объект Recorder

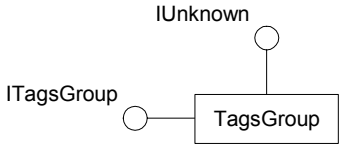


Объект Recorder выполняет общее управление ядром ПО «Recorder» и самим приложением. Реализует интерфейс IRecorder. Функции объекта Recorder интерфейса IRecorder:

1. Управление состоянием ПО «Recorder». Получение, изменение режима работы и т.п.
2. Управление тегами. Получение списка тегов, создание тегов, удаление тегов. Доступ к объектам тегов.
3. Управление группами тегов. Получение списка групп, создание, удаление. Доступ к объектам групп тегов.
4. Получение списка модулей. Доступ к объектам модулей.
5. Получение информации о физических каналах, данная информация может быть использована для создания тегов.
6. Постановка на регистрацию и снятие с регистрации формуляров, получение списка формуляров. Доступ к объектам формуляров.
7. Передача сообщений.

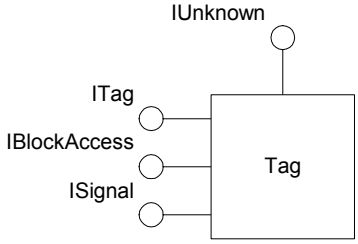
A.2.2 Объект группы тегов

Объект TagsGroup предназначен для единого управления списком тегов, реализует интерфейс ITagsGroup. Основные функции объекта:



1. Управление параметрами регистрации измеренных данных тегов группы. Установка признака необходимости регистрации.
2. Управление процессом регистрации. Запуск процесса регистрации и останов процесса регистрации для отдельно взятой группы, без останова общего процесса измерения.
3. Изменение параметров формирования файлов регистрации данных. Установка имени каталога для регистрации измеренных данных.
4. Управление списком тегов группы. Добавление и удаление тега из группы.

A.2.3 Объект тег

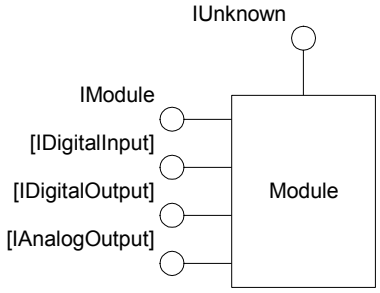


Объект Tag предназначен для управления тегом и для доступа к измеренным данным. Объект Tag реализует интерфейсы ITag, IBlockAccess и ISignal. Функции объекта тега:

1. Управление параметрами тега и аппаратного канала (если тег не является виртуальным). Получение и изменение параметров таких как имя, строка описания, частота дискретизации и т.п. Получение

минимума и максимума, получение типа данных буфера тега и т.п.
2. Управление ГФ/КФ. Получение и установка функции. Доступ к объекту ГФ/КФ.
3. Доступ к буферу данных тега, чтение, запись данных.

A.2.4 Объект модуль



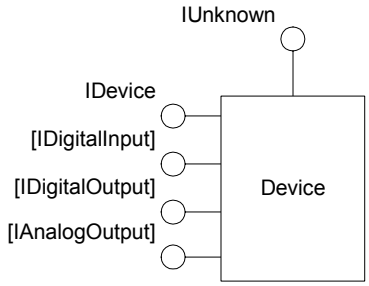
Объект Module. При помощи данного объекта предоставляется доступ к объекту обслуживания устройства ядра. Объект Module реализует интерфейс IModule. В зависимости от типа аппаратного устройства объект Module может реализовывать интерфейсы IDigitalInput, IDigitalOutput, IAnalogOutput. Интерфейс IDigitalInput реализуется для цифрового ввода. Интерфейс IDigitalOutput реализуется для цифрового вывода. Интерфейс IAnalogOutput реализуется для аналогового вывода.

Функции объекта Module:

1. Управление аппаратным устройством. Получение свойств устройства, таких как серийный номер, наименование модуля, версия и т.п.
2. «Низкоуровневые» операции по вводу и выводу, если аппаратное устройство таковые поддерживает

A.2.5 Объект устройство

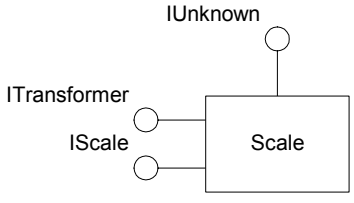
Объект Device. При помощи данного объекта предоставляется доступ к объекту обслуживания устройства ядра. Объект Device реализует интерфейсы IDevice, IDigitalInput, IDigitalOutput, IAnalogOutput. Интерфейс IDevice позволяет получить свойства устройств. Интерфейсы IDigitalInput, IDigitalOutput, IAnalogOutput реализованы не для каждого устройства, эти интерфейсы позволяют производить цифровой ввод/вывод и аналоговый вывод.



Функции объекта Device:

1. Управление аппаратным устройством. Получение свойств устройства, таких как серийный номер, наименование устройства, версия и т.п.
2. «Низкоуровневые» операции по вводу и выводу, если аппаратное устройство таковые поддерживает

A.2.6 Объекты Калибровочной и Градуировочной Функции (КФ/ГФ)

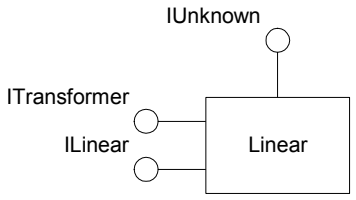


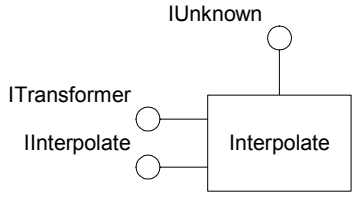
Объект Scale - объект функции «Масштабный коэффициент». Объект предназначен для обработки измеренных данных. Функции объекта:

1. Обработка значений.
2. Управление параметрами функции. Получение и изменение значений коэффициента.

Объект Linear - объект функции «Линейная функция». Объект предназначен для обработки измеренных данных. Функции объекта:

1. Обработка значений.
2. Управление параметрами функции. Получение и изменение значений коэффициентов.



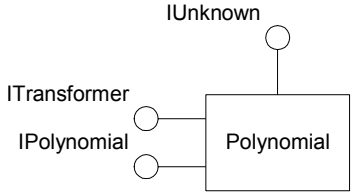


Объект Interpolate – объект функции «Таблица линейной интерполяции».
Объект предназначен для обработки измеренных данных. Функции объекта:

1. Обработка значений.
2. Управление параметрами функции. Получение и изменение значений таблицы.

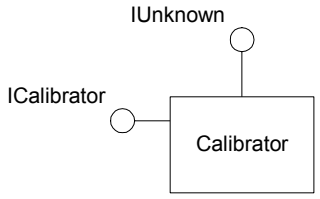
Объект Polynomial – объект функции «Полином n-го порядка». Объект предназначен для обработки измеренных данных. Функции объекта:

1. Обработка значений.
2. Управление параметрами функции. Получение и изменение массива коэффициентов.



A.2.7 Объект автоматизированной калибровки

Объект Calibrator. Данный объект используется ядром для выполнения автоматизированной калибровки, градуировки, поверки. Объект реализует интерфейс ICalibrator. Функции объекта:



1. Предоставление оконного интерфейса для настройки, измерения и отображения, обработанных данных, оператору.
2. Выполнение измерения и обработки измеренных данных, формирование отчетов.
3. Выполнение расчетов для определения параметров ГФ/КФ, установка ГФ/КФ для тегов.

A.3 Интерфейс IRecorder

После создания экземпляра plug-in`а ПО «Recorder» передает в объект plug-in`а ссылку на свой интерфейс IRecorder (см. IRecorderPlugin). Интерфейс IRecorder позволяет plug-in`у управлять ПО «Recorder», менять его состояние, получать список тегов, создавать виртуальные теги и т.п.

Интерфейс IRecorder наследует интерфейсу IUnknown. В объекте Recorder корректно реализованы методы подсчета ссылок. Поэтому необходимо контролировать счетчик ссылок интерфейса IRecorder – своевременно вызывать методы AddRef() и Release(). Перед удалением plug-in должен освободить все ссылки на интерфейс IRecorder.

A.3.1 Интерфейс IRecorder

Список функций интерфейса приведен ниже.

A.3.1.1 IRecorder::RegisterIForm

bool RegisterIForm(IVForm* a_pIVForm, long a_lParam); - Зарегистрировать формуляр отображения. В результате выполнения этой функции ПО «Recorder» получает ссылку на новый формуляр, добавляет его в общий список формуляров и имеет возможность управлять формуляром. Оператору предоставляется возможность выбрать новый формуляр для отображения.

Параметр	Описание
a_pIVForm	Ссылка на интерфейс IVForm, регистрируемого формуляра.
a_lParam	Значение этого параметра будет передано в объект формуляра параметро метода инициализации IVForm::Init().

Код результата	Описание
true	Формуляр успешно зарегистрирован.
false	Ошибка регистрации формуляра. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.2 IRecorder::UnregisterIForm

bool UnregisterIForm(IVForm* m_pIVForm); - Исключить формуляр отображения из зарегистрированных объектов. Формуляр перестает быть доступным оператору.

Параметр	Описание
a_pIVForm	Ссылка на интерфейс IVForm.

Код результата	Описание
true	Формуляр успешно снят с регистрации.
false	Операция снятия формуляра с регистрации завершена с ошибкой. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.3 IRecorder::GetIFormByName

IVForm* GetIFormByName(const char* a_pchName); - Получить ссылку на интерфейс формуляра отображения по имени формуляра.

Параметр	Описание
a_pchName	Строка с именем искомого формуляра.

Результатом выполнения функции является не нулевая ссылка на интерфейс IVForm искомого формуляра. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.4 IRecorder::GetIFormByIndex

IVForm* GetIFormByIndex(ULONG a_nIndex); - Получить ссылку на интерфейс формуляра отображения по индексу. Формуляры индексируются с 0, то есть первый формуляр списка в ПО «Recorder» имеет индекс 0.

Параметр	Описание
a_nIndex	Значение индекса требуемого формуляра

Результатом выполнения функции является не нулевая ссылка на интерфейс IVForm искомого формуляра. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.5 IRecorder::GetTagByName

ITag* GetTagByName(const char* a_pchName); - Получить ссылку на интерфейс тега по имени тега.

Параметр	Описание
a_pchName	Строка имени искомого тега.

Результатом выполнения функции является не нулевая ссылка на интерфейс ITag искомого тега. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.6 IRecorder::GetTagByIndex

ITag* GetTagByIndex(ULONG a_nIndex); - Получить ссылку на интерфейс тега по индексу тега в общем списке. Теги в списке ПО «Recorder» индексируются, начиная с 0, то есть первый тег списка имеет индекс 0.

Параметр	Описание
a_nIndex	Индекс искомого тега.

Результатом выполнения функции является не нулевая ссылка на интерфейс ITag искомого тега. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.7 IRecorder::GetTagsCount

ULONG GetTagsCount(); - Получить общее количество тегов в списке ПО «Recorder». Результатом выполнения функции является число- количество тегов.

A.3.1.8 IRecorder::GetModuleByIndex

IModule* GetModuleByIndex(ULONG a_nHostDeviceNum, ULONG a_nIndex); - Получить ссылку на интерфейс объекта Module по индексу.

Параметр	Описание
a_nHostDeviceNum	Номер измерительного крейта, либо номер измерительного устройства. Номер первого устройства в системе равен 0.
a_nIndex	Номер измерительного модуля в крейте. Нумерация модулей начинается с 0.

Результатом выполнения функции является не нулевая ссылка на интерфейс IModule искомого объекта Module. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.9 IRecorder::GetModulesCount

ULONG GetModulesCount(ULONG a_nHostDeviceNum); - Получить общее количество объектов Module в измерительном устройстве, крейте.

Параметр	Описание
a_nHostDeviceNum	Номер измерительного крейта, либо номер измерительного устройства. Номер первого устройства в системе равен 0.

Результатом выполнения функции является число- количество объектов Module.

A.3.1.10 IRecorder::GetDeviceByIndex

HRESULT GetDeviceByIndex(IDevice ** a_pDevice, ULONG a_nIndex); - Получить ссылку на интерфейс объекта Device по индексу.

Параметр	Описание
a_pDevice	Адрес переменной, в которую возвращается ссылка на интерфейс IDevice объекта Device.
a_nIndex	Номер измерительного объекта Device в общем списке. Нумерация объектов устройств начинается с 0.

Код результата	Описание
E_POINTER	
E_INVALIDARG	
S_OK	Ссылка на интерфейс IDevice объекта устройство успешно получена.

A.3.1.11 IRecorder::GetDevicesCount

int GetDevicesCount(void); - Получить общее количество объектов Device в системе.
Результатом выполнения функции является число- количество объектов Device.

A.3.1.12 IRecorder::GetGroupByIndex

ITagsGroup* GetGroupByIndex(ULONG a_nIndex); - Получить ссылку на объект группы тегов по индексу. Группы тегов в списках ПО «Recorder» нумеруются начиная с 0.

Параметр	Описание
a_nIndex	Индекс требуемого объекта группы тегов

Результатом выполнения функции является не нулевая ссылка на интерфейс ITagsGroup искомого объекта группы тегов. Если результат функции равен 0, значит, в процессе поиска, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.13 IRecorder::GetGroupsCount

ULONG GetGroupsCount(); - Получить число групп тегов
Результатом выполнения функции является число- количество объектов групп тегов.

A.3.1.14 IRecorder::GetState

RECORDERSTATE GetState(RECORDERSTATE a_rsMask); - Получить состояние ПО «Recorder». Данный метод накладывает маску на слово состоянии и возвращает результат.

Параметр	Описание
a_rsMask	Маска для слова состояния. Маска может включать любой набор флагов состояния.

Результатом выполнения функции является слово состояния с установленными флагами.

A.3.1.15 IRecorder::CheckState

bool CheckState(RECORDERSTATE a_rsState); - Проверить состояние ПО «Recorder».

Параметр	Описание
a_rsState	Список флагов состояния для проверки.

Код результата	Описание
true	Если значение параметра a_rsState совпадает с состоянием ПО «Recorder».
false	Если значение параметра a_rsState не совпадает с состоянием ПО «Recorder».

A.3.1.16 IRecorder::GetSignalFrameName

LPCSTR GetSignalFrameName(); - Получить имя подкаталога, в котором формируются файлы с измеренными данными.
В качестве результата выполнения функция возвращает ссылку на строку с полным путем и именем каталога.

A.3.1.17 IRecorder::GetSignalFolderName

LPCSTR GetSignalFolderName(); - Получить имя каталога для данных, каталога в котором создаются подкаталоги с файлами.
В качестве результата выполнения функция возвращает ссылку на строку с полным путем и именем каталога.

A.3.1.18 IRecorder::CreateTag

ITag* CreateTag(const char* a_pchName, LINKSTATE a_ls, void* a_pParams); - Создать новый тег.

Параметр	Описание
a_pchName	Строка с именем нового тега
a_ls	Код типа создаваемого тега. Допустимым значением является один из флагов: LS_HARDWARE или LS_VIRTUAL.
a_pParams	Дополнительный параметр для создания тега. При создании виртуального тега этот параметр должен быть равен 0. При создании тега, связанного с аппаратным каналом, значение этого параметра должно быть равно адресу переменной, в которой хранится идентификатор аппаратного канала.

Результатом выполнения функции является не нулевая ссылка на интерфейс ITag созданного объекта тега. Если результат функции равен 0, значит, в процессе создания нового тега, произошла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.19 IRecorder::CloseTag

bool CloseTag(ITag* a_piTag); - Удаление тега по ссылке на тег. Объект тега удаляется из списков ядра ПО «Recorder», однако, существует в памяти до тех пор, пока все его интерфейсы не будут освобождены (пока счетчик ссылок не станет равным 0).

Параметр	Описание
a_piTag	Ссылка на интерфейс удаляемого тега.

Код результата	Описание
true	Операция удаления тега была успешно выполнена.
false	В результате операции удаления возникла ошибка. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.20 IRecorder::Notify

bool Notify(DWORD a_dwCommand, DWORD a_dwParam); - Уведомить рекордер о неком событии, код события указывается параметром функции.

Параметр	Описание
a_dwCommand	Код события или сообщения.
a_dwParam	Значение данного параметра зависит от кода сообщения.

Код результата	Описание
true	Сообщение было корректно обработано.
false	Сообщение не было обработано, возможно, код сообщения некорректен либо возникла ошибка во время обработки. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.21 IRecorder::SetLastError

void SetLastError(DWORD a_dwErrorCode); - Установить код результата выполнения последней операции. Эта функция необходима для того, чтобы модули plug-in`ов имели возможность устанавливать собственные коды ошибок.

Параметр	Описание
a_dwErrorCode	Код ошибки

A.3.1.22 IRecorder::GetLastError

DWORD GetLastError(); - Метод получения кода результата выполнения последней операции ПО «Recorder» или одного из его plug-in`ов.

Код результата	Описание

RCERROR_NOERROR	Ошибки нет, последняя операция была выполнена успешно.
RCERROR_UNKNOWNERROR	Неизвестная ошибка, источник ошибки определить не удалось.
RCERROR_NOTIMPLEMENT	Вызванная функция не реализована.
RCERROR_PLUGINNOTCREATED	В процессе создания объекта plug-in`а возникла ошибка.
RCERROR_OUTOFRANGE	Индекс искомого объекта некорректен.
RCERROR_NOTFOUND	Объект не найден.
RCERROR_DLLNOTFOUND	Библиотека не найдена.
RCERROR_FUNCTIONNOTFOUND	Библиотека, указанная в качестве plug-in`а, не содержит (не реализует, не экспортирует) требуемую функцию.
RCERROR_CANTOPEN	Файл не может быть открыт.
RCERROR_CANTCREATE	Метод инициализации plug-in`а завершен с ошибкой.
RCERROR_SYSTEMBUSY	Система не может выполнить указанную операцию, к примеру, смену режима, система занята.
RCERROR_FILENOTFOUND	Файл, указанный в качестве библиотеки plug-in`а не найден.
RCERROR_INVALIDARGUMENT	Недопустимые значения одного или нескольких аргументов.
RCERROR_INVALIDTYPECAST	Ошибка приведения типа.
RCERROR_ALREADYEXIST	Объект с подобным именем уже существует
RCERROR_ABNORMALTERMINATION	Операция работы с plug-in`ом была завершена аварийно.
RCERROR_NOTSUPPORTED	Операция не поддерживается.
RCERROR_ACCESSVIOLATION	Операция привела к внутренней фтальной ошибке.
RCERROR_USERABORT	Операция была отменена оператором.
RCERROR_ACCESSDENIED	Доступ к объекту запрещен.
RCERROR_DB_EMPTY	Устройство не содержит ГФ/КФ.
RCERROR_UNKOWN_FORMAT	Неизвестный формат ГФ/КФ
RCERROR_INSUFFICIENT_SPACE	Не достаточно ресурсов для получения ГФ/КФ

A.3.1.23 IRecorder::ConvertErrorCodeToString

LPCSTR ConvertErrorCodeToString(DWORD a_dwErrorCode); - Формирование строки описания ошибки по коду ошибки.

Параметр	Описание
a_dwErrorCode	Код ошибки

В качестве результата функция возвращает ссылку на строку с описанием ошибки.

A.3.1.24 IRecorder::GetProperty

HRESULT GetProperty(DWORD a_dwPropertyID, VARIANT& a_Value); - Получить свойство объекта Recorder.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо получить.
a_Value	Параметр, в который функция вернет требуемое значение свойства.

Код результата	Описание
S_OK	Операция получения свойства выполнена успешно.
E_FAIL	Операция получения свойства завершена с ошибкой, значение параметра a_Value не изменено.

A.3.1.25 IRecorder::SetProperty

HRESULT SetProperty(DWORD a_dwPropertyID, VARIANT a_Value); - Установить новое значение свойства объекта Recorder.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо изменить.
a_Value	Параметр, в котором находится новое значение для свойства

Код результата	Описание
S_OK	Значение свойства было корректно изменено.
E_FAIL	Операция установки значения свойства завершена с ошибкой.

A.3.1.26 IRecorder::SetEnvironmentCurDir

bool SetEnvironmentCurDir(const char* a_pchDir) – Установить текущий каталог переменных окружения.

Параметр	Описание
a_pchDir	Адрес строки с именем каталога переменных окружения.

Код результата	Описание
true	Текущий каталог переменных окружения успешно устновлен.
false	Ошибка установки текущего каталога переменных окружения.

A.3.1.27 IRecorder::GetEnvironmentCurDir

bool GetEnvironmentCurDir(char* a_pchDir, int a_nLength); - Получить текущий каталог переменных окружения

Параметр	Описание
a_pchDir	Адрес буфера, в который функция перепишет строку с именем текущего каталога переменных окружения.
a_nLength	Размер буфера.

Код результата	Описание

true	Строка имени текущего каталога переменных окружения успешно возвращена.
false	Операция привела к ошибке.

A.3.1.28 IRecorder::GetEnvironmentVar

bool GetEnvironmentVar(const char* a_pchVarID, VARIANT& a_varVal) – Получить значение переменной окружения в формате типа VARIANT.

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо получить.
a_varVal	Ссылка на переменную, в которую функция вернет значение.

Код результата	Описание
true	Значение переменной окружения успешно получено функцией.
false	Операция привела к ошибке.

A.3.1.29 IRecorder::SetEnvironmentVar

bool SetEnvironmentVar(const char* a_pchVarID, VARIANT& a_varVal) – Установить значение переменной окружения в формате типа VARIANT.

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо установить.
a_varVal	Ссылка на переменную, которая содержит новое значение.

Код результата	Описание
true	Новое значение переменной окружения успешно установлено.
false	Операция привела к ошибке.

A.3.1.30 IRecorder::GetEnvironmentLong

bool GetEnvironmentLong(const char* a_pchVarID, long &a_lVal) – Получить значение переменной окружения в формате типа long (32-х битное целое)

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо получить.
a_lVal	Ссылка на переменную, в которую функция вернет значение.

Код результата	Описание
true	Значение переменной окружения успешно получено функцией.
false	Операция привела к ошибке.

A.3.1.31 IRecorder::SetEnvironmentLong

bool SetEnvironmentLong(const char* a_pchVarID, long a_lVal) – Установить значение переменной окружения в формате типа long (32-х битное целое)

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо установить.
a_lVal	Ссылка на переменную, которая содержит новое значение.

Код результата	Описание
true	Новое значение переменной окружения успешно установлено.
false	Операция привела к ошибке.

A.3.1.32 IRecorder::GetEnvironmentDouble

bool GetEnvironmentDouble(const char* a_pchVarID, double &a_dblVal) - Получить значение переменной окружения в формате типа double (64-х битное действительное)

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо получить.
a_dblVal	Ссылка на переменную, в которую функция вернет значение.

Код результата	Описание
true	Значение переменной окружения успешно получено функцией.
false	Операция привела к ошибке.

A.3.1.33 IRecorder::SetEnvironmentDouble

bool SetEnvironmentDouble(const char* a_pchVarID, double a_dblVal) - Установить значение переменной окружения в формате типа long (64-х битное действительное)

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо установить.
a_dblVal	Ссылка на переменную, которая содержит новое значение.

Код результата	Описание
true	Новое значение переменной окружения успешно установлено.
false	Операция привела к ошибке.

A.3.1.34 IRecorder::GetEnvironmentString

bool GetEnvironmentString(const char* a_pchVarID, char* a_pchVal, int* a_pnLength) - Получить значение переменной окружения в виде строки.

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо получить.
a_pchVal	Адрес буфера, в который функция должна переписать строку.
a_pnLength	Размер буфера.

Код результата	Описание

true	Значение переменной окружения успешно получено функцией.
false	Операция привела к ошибке.

A.3.1.35 IRecorder::SetEnvironmentString

bool SetEnvironmentString(const char* a_pchVarID, const char* a_pchlVal); - Установить значение переменной окружения в виде char*

Параметр	Описание
a_pchVarID	Строка с именем переменной окружения, значение которой необходимо установить.
a_pchlVal	Адрес строки – нового значения переменной окружения.

Код результата	Описание
true	Новое значение переменной окружения успешно установлено.
false	Операция привела к ошибке.

A.3.1.36 IRecorder::LogMessage

bool LogMessage(const char* a_pchMessage) – Добавить в отладочный журнал событий ПО «Recorder» строку с переводом каретки.

Параметр	Описание
a_pchMessage	Адрес строки с сообщением.

Код результата	Описание
true	Строка успешно добавлена в отладочный журнал.
false	Операция привела к ошибке. Код ошибки можно получить при помощи метода IRecorder::GetLastError().

A.3.1.37 IRecorder::GetHWND

HWND GetHWND(); - Получить идентификатор главного окна ПО «Recorder». Эта функция используется для создания plug-in`ом дочерних окон.
В качестве результата функция возвращает системный идентификатор (HWND) окна ПО «Recorder».

A.3.1.38 IRecorder::GetCurrentTag

ITag* GetCurrentTag(); - Получить ссылку на текущий, выбранный для отображения данных тег.

A.3.1.39 IRecorder::SetCurrentTag

HRESULT SetCurrentTag(ULONG a_nIndex); - Установить текущий тег для отображения данных.

Параметр	Описание
a_nIndex	Индекс тега в общем списке тегов.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.3.2 Флаги состояния Recorder`a

Эти флаги можно получить при помощи метода интерфейса Recorder`a IRecorder::GetState().

№	Наименование	Описание
1	RS_STOP	Recorder остановлен, сбор данных не производится. Режим «Готовность»
2	RS_VIEW	Recorder находится в режиме «Измерение».
3	RS_REC	Recorder находится в режиме «Регистрация».
4	RS_PLAYING	Recorder находится в режиме просмотра ранее произведенных записей. Режим «Воспроизведение», данный режим в текущей версии ПО «Recorder» не реализован.
5	RS_BASESTATE	Маска на базовые состояния ПО «Recorder» – это RS_REC | RS_VIEW | RS_STOP | RS_PLAYING.
6	RS_HARDWAREFAULT	Инициализация системы не прошла успешно, ошибка инициализации аппаратных средств.
7	RS_INITFAULT	Инициализация системы не прошла успешно, ошибка инициализации программных средств.
8	RS_NEEDDEVICERESET	Требуется сброс аппаратных средств.
9	RS_NEEDHARDWARERESET	Требуется сброс аппаратных средств.
10	RS_NEEDSOFTWARERESET	Требуется сброс программной части драйвера аппаратуры.
11	RS_NEEDLINKSREFRESH	Требуется обновление ссылок. Это состояние может возникать после переконфигурирования программы.
12	RS_FAULT	Маска на состояния ошибок инициализации системы RS_HARDWAREFAULT | RS_INITFAULT.
13	RS_PLAYMODE	Настройка и работа в режиме «Воспроизведение».
14	RS_SIGNALLOADED	Файлы для воспроизведения загружены, конфигурирование успешно завершено. Настройка перед режимом «Воспроизведение».
15	RS_CONFIGCHANGED	Конфигурация ПО «Recorder» была изменена, но не была сохранена в файл, требуется сохранение конфигурации.
16	RS_CONFIGMODE	ПО «Recorder» находится в режиме «Настройка»
17	RS_PAUSE	Регистрация измеренных данных временно приостановлена, пауза
18	RS_PACKETLOST	Флаг, обозначающий, что от начала режима «Измерение» или «Регистрация», были обнаружены потери измеренных данных.
19	RS_RECEIVEERROR	Флаг, обозначающий потери измеренных данных, сбрасывается при обновлении отображения измеренных данных.
20	RS_ENABEDPAUSE	Флаг допустимости регистрации измеренных данных с паузами.
21	RS_TERMINATION	Состояние завершения работы.

№	Наименование	Описание
22	RS_FULLMASK	Маска на все информационные биты флагового слова.

A.3.3 Коды команд сообщений

Коды команд сообщений

№	Наименование	Описание
1	RCN_VIEW	Запрос на переход в режим «Измерение»
2	RCN_REC	Запрос на переход в режим «Регистрация»
3	RCN_STOP	Запрос на переход в режим «Готовность»
4	RCN_SHOW	Сделать главное окно видимым
5	RCN_HIDE	Скрыть главное окно
6	RCN_CLOSE	Завершить работу ПО «Recorder»

A.3.4 Коды идентификаторов свойств

Коды идентификаторов свойств

№	Наименование	Доступ	Тип значения	Описание
1	RCPROP_DATAFOLDER	чтение и запись	VT_BSTR	Рабочий каталог, в котором формируются подкаталоги с измеренными данными.
2	RCPROP_STARTRECLEVCTRLPROPS	чтение и запись	см. описание	Параметры механизма запуска процесса «Регистрации» по сигналу. Идентификатор типа данных, поле vt, содержит значение VT_BYREF, поле byref содержит ссылку на структуру RCLEVELCONTROLPROPS. При чтении свойства необходимо освобождать память, выделенную под структуру. При записи свойства память под структуру необходимо выделить, её освободит объект Recorder. Выделение и освобождение памяти производится при помощи функций управления памяти COM служб.
3	RCPROP_STARTTIME	чтение	VT_FILETIME	Системное время, когда был запущен режим «Измерение» или «Регистрация». В данной версии ПО «Recorder», это свойство зарезервировано.
4	RCPROP_CALIBRDBNAME		VT_BSTR	Имя подкаталога базы ГФ/КФ, в корневом каталоге ПО «Recorder».
5	RCPROP_REFRESHPERIOD	чтение и запись	VT_R8	Значение периода обновления (размер блока данных в буфере тега) в секундах.
6	RCPROP_VIEWTIME	чтение и запись	VT_R8	Значение времени отображения (размер буфера данных тега) в секундах.
7	RCPROP_VERSION	чтение	VT_UI4	Код номера версии ПО «Recorder». Формат кода описан типом VERSION_DWORD.
8	RCPROP_TAGSSORTMODE	чтение и запись	VT_I4	Значение кода метода сортировки тегов, это может быть один из флагов SRTMD_BY_NAME, SRTMD_BY_ADDRESS, SRTMD_BY_DEVICENAME.
9	RCPROP_TIMERTICPERIOD	чтение и запись	VT_R8	Период обновления отображения измеренных данных в главном окне ПО «Recorder»

№	Наименование	Доступ	Тип значения	Описание
10	RCPROP_MODIFYFRAMENAME	чтение и запись	VT_BOOL	Признак автоматического изменения имени подкаталога, в котором сохраняются измеренные данные. Для каждого отдельного процесса регистрации создается отдельный каталог.
11	RCPROP_ENABLEDLEVELSTARTREC	чтение и запись	VT_BOOL	Разрешен старт записи по уровню
12	RCPROP_TRIGGERSTART	чтение и запись	VT_BOOL	Состояние триггерног старта
13	RCPROP_ENABLEDAUTOSTOPTIME	чтение и запись	VT_BOOL	Признак автоматического перехода в режим «Останов» из режима «Измерение» или «Регистрация» по истечении времени.
14	RCPROP_AUTOSTOPTIME	чтение и запись	VT_R8	Временной промежуток, через который произойдет переход в режим «Останов». Значение в секундах.
15	RCPROP_CONFIGNAME	чтение	VT_BSTR	Имя файла конфигурации
16	RCPROP_UISERVERLINK	чтение и запись	VT_UNKNOWN	Ссылка на объект сервера оконного интерфейса.
17	RCPROP_SUMMARYDATAFLOW	чтение	VT_I4	Суммарный поток данных
18	RCPROP_PROJECTNAME	запись	VT_BSTR	Имя файла проекта
19	RCPROP_CURRENTTAGNAME	чтение и запись	VT_BSTR	Строка с именем текущего тега, тега для которого отображаются измеренные данные.

A.3.5 Код метода сортировки тегов

Коды метода сортировки

№	Наименование	Описание
1	SRTMD_BY_NAME	Сортировать по имени
2	SRTMD_BY_ADDRESS	Сортировка по строке адреса физического канала
3	SRTMD_BY_DEVICENAME	Сортировка по строке имени устройства владельца

A.3.6 Структура RCLEVELCONTROLPROPS

Коды типов анализа уровня сигнала, перечисление LEVEL.

№	Наименование	Описание
1	LV_EQUAL	Уровень равен пороговому
2	LV_ABOVE	Уровень превышает пороговый
3	LV_BELOW	Уровень меньше порогового

Структура RCLEVELCONTROLPROPS.

№	Имя поля	Тип поля	Описание
1	pTag	ITag*	Ссылка на объект тега, данные которого служит для определения условий перехода в режим «Регистрация».
2	pchTagName	char [256]	Строка имени объект тега, данные которого служит для определения условий перехода в режим «Регистрация».
3	dblLevel	double	Пороговое значение
4	level	LEVEL	Тип анализа уровня сигнала, любое из значений перечисления LEVEL.

A.3.7 Коды типов тегов, LINKSTATE

Коды типов тегов

№	Наименование	Описание
1	LS_HARDWARE	Тег связан с физическим каналом.
2	LS_VIRTUAL	Тег является виртуальным, не имеет связи с физическим каналом
3	LS_LOST	Тег был связан с физическим каналом, однако в результате изменения настройки физический канал (измерительное устройство) был удален.

A.3.8 Структура VERSION_DWORD

№	Имя поля	Номера битов (размер поля)	Описание
1	w	0..3 (4)	Код типа текущей версии, это: альфа - тестовая версия (VERSION_ALPHA), бета – тестовая версия (VERSION_BETTA), окончательная версия (VERSION_RELEASE).
2	bn	4..11 (8)	Номер сборки.
3	bf	12..19 (8)	Номер исправлений ошибок
4	min	20..27 (8)	Подномер версии
5	maj	28..31 (4)	Порядковый номер версии

A.4 Интерфейс IRecorderPlugin

A.4.1 Интерфейс IRecorderPlugin

A.4.1.1 IRecorderPlugin::create

bool create(IRecorder* a_pOwner); - Инициализация внутренних данных объекта plug-in`a, выделение необходимых ресурсов.

Параметр	Описание
a_pOwner	Ссылка на интерфейс IRecorder объекта Recorder.

Код результата	Описание
true	Объект plug-in`a возвращает это значение в том случае, если инициализация выполнена успешно.
false	Объект plug-in`a возвращает это значение в том случае, если инициализация plug-in`a завершена ошибкой.

A.4.1.2 IRecorderPlugin::config

bool config(); - Метод настройки объекта plug-in`a перед запуском штатного режима.

Код результата	Описание
true	Объект plug-in`a возвращает это значение в том случае, если настройка выполнена успешно.
false	Объект plug-in`a возвращает это значение в том случае, если настройка завершена ошибкой.

A.4.1.3 IRecorderPlugin::edit

bool edit(); - Вызов окна настройки. В окне настройки plug-in может отображать значения любых своих параметров и предоставлять оператору возможность их изменения.

Код результата	Описание
true	Окно настройки успешно отображено.
false	Окно настройки отображено не было, plug-in не имеет окна настройки.

A.4.1.4 IRecorderPlugin::execute

bool execute(); - Запуск plug-in`а. Переход plug-in`а в штатный режим работы, создание необходимых окон, потоков и пр.

Код результата	Описание
true	Запуск произведен успешно.
false	Запуск произведен с ошибкой. Объект plug-in`а не смог выполнить необходимые ему действия.

A.4.1.5 IRecorderPlugin::suspend

bool suspend(); - Приостановка работы plug-in`а. Данная функция в текущей версии ПО «Recorder» не используется.

A.4.1.6 IRecorderPlugin::resume

bool resume(); - Возобновление работы, после её приостанови. Данная функция в текущей версии ПО «Recorder» не используется.

A.4.1.7 IRecorderPlugin::notify

bool notify(DWORD a_dwCommand, DWORD a_dwData); - Уведомление о внешних событиях. При помощи этого метода Recorder оповещает plug-in о событиях, таких как обновление данных, начало изменения конфигурации и т.п.

Параметр	Описание
a_dwCommand	Код команды сообщения
a_dwData	Данные, формат данных зависит от типа команды

Код результата	Описание
true	Объект plug-in`а обработал сообщение.
false	Объект plug-in`а не обработал сообщение.

A.4.1.8 IRecorderPlugin::getname

LPCSTR getname(); - Метод получения наименования plug-in`а. Метод используется ПО «Recorder», для отображения списка имен запущенных plug-in`ов.
В качестве результата метод должен вернуть адрес строки с именем plug-in`а.

A.4.1.9 IRecorderPlugin::getproperty

bool getproperty(DWORD a_dwPropertyID, VARIANT& a_Value); - Получить свойство plug-in`a.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо получить.
a_Value	Параметр, в котором функция должна вернуть значение свойства.

Код результата	Описание
true	Значение свойства корректно получено.
false	Значение свойства не было получено корректно.

A.4.1.10 IRecorderPlugin::setproperty

bool setproperty(DWORD a_dwPropertyID, VARIANT a_Value); - Установить значение свойства объекта plug-in`a.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо изменить.
a_Value	Новое значение свойства

Код результата	Описание
true	Значение свойства корректно установлено.
false	Значение свойства не было установлено корректно.

A.4.1.11 IRecorderPlugin::canclose

bool canclose(); - Проверка допустимости завершения работы объекта plug-in`a. Метод вызывается ПО «Recorder» перед завершением работы plug-in`a.

Код результата	Описание
true	Работа объекта plug-in`a может быть корректно завершена, и объект plug-in`a может быть удален.
false	Объект plug-in`a занят, завершение его работы в данный момент не может быть выполнено.

A.4.1.12 IRecorderPlugin::close

bool close(); - Завершить работу plug-in`a. В этом методе plug-in должен освободить все выделенные ранее системные ресурсы и завершить свою работу, при необходимости корректно завершить работу собственных потоков.

A.4.2 Коды сообщений

Коды команд сообщений

№	Наименование	Описание
1	PN_ENTERRCCONFIG,	ПО «Recorder» перешло в режим «Настройка»
2	PN_LEAVERCCONFIG	ПО «Recorder» вышло из режима «Настройка»
3	PN_CHANGECURTAG	Текущий тег был изменен, теперь текущим является другой тег, ссылка на новый текущий тег находится в параметре a_dwData, тип – ссылка на интерфейс ITag.
4	PN_RCSTART	ПО «Recorder» перешло в режим «Измерения» или «Регистрации»
5	PN_RCPLAY	ПО «Recorder» перешло в режим «Воспроизведения», данный режим в текущей версии не реализован.
6	PN_RCSTOP	ПО «Recorder» перешло в режим «Готовность»
7	PN_UPDATEDATA	Для некоторых тегов были получены новые порции измеренных или обработанных данных.
8	PN_SHOWINFO	Команда на отображение объектом plug-in`а собственного окна с описанием. ПО «Recorder» генерирует данное сообщение по запросу оператора, по нажатию кнопки «Инфо» в окне настройки.
9	PN_RCSAVECONFIG	ПО «Recorder» выполнило сохранение собственной настройки в файлы.
10	PN_RCLOADCONFIG	ПО «Recorder» выполнило чтение собственной настройки из файлов.
11	PN_SYNCHRO_READ_DATA_BLOCK	Для виртуального тега выполняется синхронное чтение данных. Параметр a_dwData содержит ссылку на структуру RCNOTIFY. Поле pSender содержит ссылку на интерфейс IRecorder объекта Recorder, поле lParam содержит количество требуемых данных, поле pvParam содержит ссылку на буфер который необходимо заполнить данными.
12	PN_EDIT_VIRTUAL_TAG_PROPS	Команда на отображение специализированного диалога редактора свойств виртуального тега, если таковой существует. Параметр a_dwData содержит ссылку на структуру RCNOTIFY. Поле pvParam равно ссылке на интерфейс ITag редактируемого объекта тега.
13	PN_IMPORTSETTINGS	Команда на загрузку настроек объекта plug-in`а из определенного файла.
14	PN_EXPORTSETTINGS	Команда на сохранение настроек объекта plug-in`а в определенный файл.
15	PN_BEFORE_RCSTART	Команда на подготовку к переходу в режим «Измерение» или «Регистрация».
16	PN_BEFORE_RCSTOP	Команда на подготовку выходу из режима «Измерение» или «Регистрация».
17	PN_ABORT_RCSTART	Оповещение о том, что переход в режим «Измерение» или «Регистрация» не был выполнен.

Структура RCNOTIFY

№	Имя поля	Тип поля	Описание
1	pSender	IUnknown*	Ссылка на интерфейс инициатора сообщения или команды.
2	nSubCommand	UINT	Дополнительный код характеризующий сообще-

			ние либо команду.
3	lParam	long	Значение поля зависит от кода сообщение или команды.
4	dblParam	double	Значение поля зависит от кода сообщение или команды.
5	pvParam	void*	Значение поля зависит от кода сообщение или команды.

A.4.3 Коды идентификаторов свойств

№	Наименование	Доступ	Тип значения	Описание
1	PLGPROP_INFOSTRING	чтение	VT_BSTR	Строка описания модуля plug-in`a.

A.5 Интерфейс IVForm

A.5.1 Интерфейс IVForm

A.5.1.1 IVForm::getname

LPCSTR getname(); - Получить имя формуляра должно быть уникальным, используется при регистрации
Метод должен возвращать адрес строки с именем формуляра. Имя формуляра отображается оператору при выборе формуляра в ПО «Recorder».

A.5.1.2 IVForm::init

bool init(IRecorder* a_pParent, HWND a_hParent, long int a_lParam); - Инициализация формы

Параметр	Описание
a_pParent	Ссылка на интерфейс IRecorder объекта Recorder.
a_hParent	Идентификатор главного окна ПО «Recorder». Формуляр должен использовать главное окно ПО «Recorder» в качестве родительского.
a_lParam	Резервный параметр.

Код результата	Описание
true	Инициализация окна формуляра и самого объекта формуляра выполнена успешно.
false	Инициализация формуляра завершена с ошибкой.

A.5.1.3 IVForm::getHWND

HWND getHWND(); - Получить идентификатор окна формуляра.

A.5.1.4 IVForm::close

bool close(); - Закрыть окно формуляра. ПО «Recorder» завершает работу и дает команду на закрытие окна формуляра. Результат функции всегда должен быть равен «true».

A.5.1.5 IVForm::prepare

bool prepare() – Метод для подготовки данных к отображению. ПО «Recorder» вызывает данный метод периодически. Период равен периоду обновления отображения. Результат функции всегда должен быть равен «true».

A.5.1.6 IVForm::update

bool update() – Данный метод является резервным, создан для применения в будущих разработках.

A.5.1.7 IVForm::repaint

bool repaint(); - Метод для отображения (обновления) данных. ПО «Recorder» вызывает данный метод периодически. Период равен периоду обновления отображения. Результат функции всегда должен быть равен «true».

A.5.1.8 IVForm::linktags

bool linktags(ITag** a_pTagsList, ULONG a_nTagsCount); - Данный метод является резервным, создан для применения в будущих разработках.

A.5.1.9 IVForm::activate

bool activate() – Активизация формуляра. Этот метод вызывается, когда оператор выбрал формуляр для отображения. Объект формуляра должен отобразить (сделать видимым) окно формуляра.

Код результата	Описание
true	Окно формуляра успешно отображено.
false	Отображение окна формуляра завершено с ошибкой.

A.5.1.10 IVForm::deactivate

bool deactivate() – Деактивация формуляра. Это метод вызывается ПО «Recorder», в том случае, если формуляр был активным и оператор выбрал другой формуляр. На вызов метода deactivate() объект формуляра должен скрывать свое окно (делать его невидимым). И так как окно не активно, то и обновление данных в окне нет необходимости выполнять. Результат функции всегда должен быть равен «true».

A.5.1.11 IVForm::edit

bool edit() - Отображение окна редактирования свойств формуляра. В данной версии ПО «Recorder» свойства формуляров не редактируются.

A.5.1.12 IVForm::notify

bool notify(DWORD a_dwCommand, DWORD a_dwData); - События, уведомления, команды

Параметр	Описание
a_dwCommand	Идентификатор команды или сообщения.
a_dwData	Параметр с дополнительными данными к команде, значение параметра зависит от кода команды.

Код результата	Описание
true	Операция выполнена успешно, команда или сообщение обработана.
false	Операция завершена с ошибкой.

A.5.2 Коды команд и сообщений

Коды команд оповещений формуляров отображения, ПО «Recorder» вызывает метод IVForm::notify() для того, чтобы передать форме отображения эти команды.

№	Наименование	Описание
1	VSN_ENTERRCCONFIG,	ПО «Recorder» перешел в режим «Настройка»
2	VSN_LEAVERCCONFIG	ПО «Recorder» вышел из режим «Настройка»
3	VSN_CHANGECURTAG	Текущий тег был изменен, теперь текущим является другой тег, ссылка на новый текущий тег находится в параметре a_dwData, тип – ссылка на интерфейс ITag.
4	VSN_RCSAVECONFIG	ПО «Recorder» выполнило сохранение собственной настройки в файлы.
5	VSN_RCLOADCONFIG	ПО «Recorder» выполнило чтение собственной настройки из файлов.
6	VSN_RCSTART	ПО «Recorder» перешло в режим «Измерения» или «Регистрации»
7	VSN_RCPLAY	ПО «Recorder» перешло в режим «Воспроизведения», данный режим в текущей версии не реализован.
8	VSN_RCSTOP	ПО «Recorder» перешло в режим «Готовность»
9	VSN_BEFORE_RCSTART	Команда на подготовку к переходу в режим «Измерение» или «Регистрация».
10	VSN_BEFORE_RCSTOP	Команда на подготовку выходу из режима «Измерение» или «Регистрация».
11	VSN_ABORT_RCSTART	Оповещение о том, что переход в режим «Измерение» или «Регистрация» не был выполнен.
12	VSN_RESIZE	Команда на изменение размеров окна формуляра. Параметр a_dwData содержит ссылку на объект типа RECT, в котором находятся новые координаты для окна. Структура RECT описана в спецификации Win32.
13	VSN_CHANGEVIEWTIME	Оповещение о том, что интервал отображения был изменен. Размер нового интервала находится в параметре a_dwData, значение в процентах от максимального интервала отображения.

№	Наименование	Описание
14	VSN_CHANGETIMESHIFT	Оповещение о том, что смещение интервала отображения был изменен. Величина смещения находится в параметре a_dwData, значение в процентах от максимального интервала отображения.
15	VSN_CHANGERCSTATE	Оповещение о том, что состояние ПО «Recorder» изменилось. Новое значение слова состояния находится в параметре dwData.

A.6 Интерфейс ITag

A.6.1 Интерфейс ITag

A.6.1.1 ITag::GetLinkState

LINKSTATE GetLinkState() – Код состояния тега или код типа тега.
В качестве результата функция возвращает один из флагов перечисления LINKSTATE, это может быть LS_HARDWARE, LS_VIRTUAL, LS_LOST.

A.6.1.2 ITag::Edit

bool Edit() – Активировать окно редактирования свойств тега.
В качестве результата функция возвращает признак успешности редактирования свойств тега.

Код результата	Описание
true	Функция выполнена успешно.
false	Функция выполнена с ошибкой.

A.6.1.3 ITag::SetName

bool SetName(const char* a_pchName) – Установить новое имя тега. Имя тега должно быть уникальным в списке имен тегов ПО «Recorder».

Параметр	Описание
a_pchName	Адрес строки с новым именем тега.

Код результата	Описание
true	Имя тега успешно изменено.
false	Операция привела к ошибке, имя тега не было изменено.

A.6.1.4 ITag::GetName

LPCSTR GetName() – Получить имя тега. В качестве результата функция возвращает адрес строки с именем тега.

A.6.1.5 ITag::SetFreq

void SetFreq(double a_dblFreq) – Установить частоту дискретизации тега.

Функция устанавливает частоту тега и частоту дискретизации физического канала, если тег связан с физическим каналом. В соответствии со свойствами физического канала частота дискретизации устанавливается наиболее близкая к требуемому значению.

Параметр	Описание
a_dblFreq	Требуемая частота дискретизации тега.

A.6.1.6 ITag::GetFreq

double GetFreq() – Получить частоту дискретизации тега. В качестве результата функция возвращает значение частоты дискретизации тега. Если тег связан с физическим каналом, то функция возвращает значение частоты дискретизации физического канала.

A.6.1.7 ITag::Notify

bool Notify (DWORD a_dwCommand, DWORD a_dwParam) – Передать программному объекту тегу команду или сообщение.

Параметр	Описание
a_dwCommand	Код команды или сообщения.
a_dwParam	Дополнительный параметр, значение которого зависит от команды.

Код результата	Описание
true	Операция выполнена успешно, сообщение обработано объектом.
false	Операция привела к ошибке.

A.6.1.8 ITag::SetProperty

bool SetProperty(DWORD a_dwPropertyID, VARIANT a_Value) – Установить значение свойства тега по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства.
a_Value	Новое значение свойства.

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.9 ITag::GetProperty

bool GetProperty(DWORD a_dwPropertyID, VARIANT& a_Value) – Получить значение свойства тега по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства.
a_Value	Ссылка на переменную, в которую функция вернет значение свойства.

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.10 ITag::GetData

ISignal* GetData() – Получить ссылку на интерфейс ISignal для получения данных тега. В качестве результата функция возвращает ссылку на интерфейс.

A.6.1.11 ITag::SynchroReadDataBlock

HRESULT SynchroReadDataBlock(double a_dblFreq, DWORD a_dwPortionLen, double* a_pdblBuffer,BOOL m_boolViaTransformer) – Функция чтения измеренных данных. Функция выполняет следующие действия:

1. Запуск процесса измерения (с требуемой частотой дискретизации).
2. Чтение измеренных данных, до тех пор, пока не будет получено требуемое количество измеренных данных.
3. Останов процесса измерения.

Функция может использоваться в режиме «Останов».

Параметр	Описание
a_dblFreq	Частота дискретизации, которая должна быть использована при получении в процессе измерения.
a_dwPortionLen	Количество измеренных данных.
a_pdblBuffer	Буфер для получения измеренных данных. Размер буфера должен соответствовать значению a_dwPortionLen.
m_boolViaTransformer	Признак необходимости обработки измеренных данных при помощи градуировочной функции тега (если таковая имеется).

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция привела к ошибке.

A.6.1.12 ITag::isCfgWritable

bool isCfgWritable() – Функция получения признака необходимости сохранения параметров тега в файлы конфигурации ПО «Recorder». В качестве результата функция возвращает значение признака.

A.6.1.13 ITag::CfgWritable

bool CfgWritable(bool a_fNewState) – Функция установки признака необходимости сохранения параметров тега в файлы конфигурации ПО «Recorder».

Параметр	Описание
a_fNewState	Требуемое значение признака сохранения параметров тега в файлы конфигурации ПО «Recorder»

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.14 ITag::PushData

bool PushData(void* a_pData) – Функция добавления блока данных в буфер тега. Функция используется для виртуальных тегов и для тегов, связанных с выходными физическими каналами.

Параметр	Описание
a_pData	Адрес буфера с блоком данных. Количество значений в блоке должно соответствовать размеру блока буфера тега. Тип данных соответствует типу данных блоков буфера тега.

Код результата	Описание
True	Операция выполнена успешно.
False	Операция привела к ошибке.

A.6.1.15 ITag::PushValue

bool PushValue(double dValue, double dXValue) – Функция установки тегу одного значения. Функция может быть использована только для тегов, связанных с выходными физическими каналами.

Параметр	Описание
dValue	Значение для вывода в тег
dXValue	Резервный параметр функции, значение параметра должно быть равно 0.

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.16 ITag::SetYRange

bool SetYRange(float a_fltRange) – Функция установки масштаба по оси Y, в процентах от диапазона отображения.

Параметр	Описание
a_fltRange	Значение масштаба

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.17 ITag::GetYRange

float GetYRange() – Функция получения масштаба по оси Y. В качестве результата функция возвращает значение масштаба.

A.6.1.18 ITag::SetYShift

bool SetYShift(float a_fltShift) – Функция установки смещения окна отображения по оси Y, в процентах от диапазона отображения.

Параметр	Описание
a_fltShift	Значение смещения

Код результата	Описание
true	Операция выполнена успешно.
false	Операция привела к ошибке.

A.6.1.19 ITag::GetYShift

float GetYShift() – Функция получения смещения по оси Y. В качестве результата функция возвращает значение смещения.

A.6.1.20 ITag::LinkGroup

HRESULT LinkGroup(ITagsGroup* a_pGroup) – Функция позволяет указать (изменить) группу, которой тег принадлежит.

Параметр	Описание
a_pGroup	Ссылка на интерфейс ITagsGroup объекта группы, которой должен принадлежать тег.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция привела к ошибке.

A.6.1.21 ITag::GetGroup

HRESULT GetGroup(ITagsGroup** a_ppGroup) – Функция получения ссылки на интерфейс объекта группы, которой принадлежит тег.

Параметр	Описание
a_ppGroup	Адрес переменной в которую функция вернет ссылку на интерфейс ITagsGroup объекта группы тегов.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция привела к ошибке.

A.6.1.22 ITag::GetDataType

VARTYPE GetDataType() – Код типа данных тега. В качестве результата функция возвращает код типа данных, аналогичный коду типа, используемому в переменных типа VARIANT.

A.6.1.23 ITag::SetEstimate

void SetEstimate(double a_dblValue, ULONG a_nMask) – Функция установки значения математической оценки по идентификатору оценочной функции.

Параметр	Описание
a_dblValue	Значение оценки
a_nMask	Флаг типа математической функции вычисления оценки, допустимым является один из флагов ESTIMATOR_EMPTY, ESTIMATOR_MEAN, ESTIMATOR_RMSV, ESTIMATOR_RMSD, ESTIMATOR_PEAK, ESTIMATOR_P2P.

A.6.1.24 ITag::GetEstimate

double GetEstimate(ULONG a_nEstimator) – Функция получения математической оценки по флагу типа математической оценочной функции.

Параметр	Описание
a_nEstimator	Флаг типа математической функции вычисления оценки, допустимым является один из флагов ESTIMATOR_EMPTY, ESTIMATOR_MEAN, ESTIMATOR_RMSV, ESTIMATOR_RMSD, ESTIMATOR_PEAK, ESTIMATOR_P2P.

A.6.1.25 ITag::SetEstimatorsMask

HRESULT SetEstimatorsMask(ULONG a_nMask) – Функция установки флагов типов математических функций, используемых для обработки данных тега.

Параметр	Описание
a_nMask	Флаг типа математической функции вычисления оценки, допустимым является один или несколько флагов ESTIMATOR_EMPTY, ESTIMATOR_MEAN, ESTIMATOR_RMSV, ESTIMATOR_RMSD, ESTIMATOR_PEAK, ESTIMATOR_P2P.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция привела к ошибке.

A.6.1.26 ITag::GetEstimatorsMask

ULONG GetEstimatorsMask() – Функция получения флагов типов математических функций, используемых для обработки данных тега. В качестве результата функция возвращает группу флагов.

A.6.1.27 ITag::Eval

double Eval(double a_dblValue) – Обработать значение при помощи функции ГФ тега. В качестве параметра функция получает исходное значение, в качестве результата возвращает обработанное значение.

A.6.2 Коды сообщений для тега

Сообщения тегу передаются через метод ITag::Notify().

№	Наименование	Описание
1	TN_UPDATETAGTX	Обновить TX
2	TN_ERASETAGTX	Удалить тарировочную характеристику Тега
3	TN_SETVECTORDELTA	Задать dX для вектора данных, используется для Tags
4	TN_WRITETAGTXINFOFILE	Прописать тарировки тега (служебное)
5	TN_PREPARETARGETS	Подготовить таргеты dwParams==MASK (служебное)
6	TN_FINISHTARGETS	Завершить работу таргетов dwParams==MASK (служебное)
7	TN_STARTREC	Перевести тег в режим записи (служебное)
8	TN_STOPREC	Остановить режим записи тега (служебное)
9	TN_ERASEDEVTX	Удалить тарировочную характеристику аппаратного канала

A.6.3 Тип тега

Таблица флагов типов тегов

№	Наименование	Описание
1	TTAG_VECTOR	Векторный тег
2	TTAG_SCALAR	Скалярный тег
3	TTAG_OUTPUT	Тег предназначен для вывода сигналов

A.6.4 Типы функций оценки данных для тегов

Таблица флагов типов оценочных функций

№	Наименование	Описание
1	ESTIMATOR_EMPTY	Оценка (обработка) данных не производится.
2	ESTIMATOR_MEAN	Функция вычисления среднего арифметического (МО).
3	ESTIMATOR_RMSV	Функция вычисления среднеквадратического значения (СКЗ).
4	ESTIMATOR_RMSD	Функция вычисления среднеквадратического отклонения (СКО).
5	ESTIMATOR_PEAK	Функция определения амплитуды сигнала (Пик).
6	ESTIMATOR_P2P	Функция определения размаха сигнала (Пик-Пик).

A.6.5 Идентификаторы свойств тегов

В таблице описаны идентификаторы свойств тегов, значеbния свойств можно получить и установить при помощи методов ITag::GetProperty() и ITag::SetProperty().

№	Наименование	Доступ	Тип значения	Описание
1	TAGPROP_BUFFSIZE	чтение	VT_I4	Размер одного блока в буфере данных тега. Размер блока в элементах.
2	TAGPROP_ESTIMATOR	чтение и запись	VT_I4	Слово с флагами оценочных функций, при помощи которых обрабатываются данные тега. Слово может содержать один из флагов ESTIMATOR_MEAN, ESTIMATOR_RMSV, ESTIMATOR_RMSD, ESTIMATOR_PEAK, ESTIMATOR_P2P, а также слово может быть равно значению ESTIMATOR_EMPTY.
3	TAGPROP_DATATYPE	чтение и запись	-	Код типа данных измерительного канала. Тип возвращаемого значения соответствует тому типу данных, который используется буфером тега. Свойство доступно для записи только для виртуальных тегов.
4	TAGPROP_HARDWAREADRESS	чтение	VT_BSTR	Строка физического адреса канала связанного с тегом, для виртуального тега возвращается строка "Virtual".
5	TAGPROP_TYPE	чтение и запись	VT_I4	Слово с флагами типа тега. Слово может содержать флаг TTAG_OUTPUT для выходных тегов и может содержать один из флагов TTAG_VECTOR, TTAG_SCALAR. Свойство доступно для записи только для виртуального тега.
6	TAGPROP_DEVCHANID	Чтение	VT_UI4	Идентификатор физического канала, связанного с тегом. Свойство имеет корректное значение только для тегов, связанных с физическими каналами.
7	TAGPROP_MINVALUE	чтение и запись	VT_R4	Минимальное значение данных, которое может быть получено от тега. Для тега, связанного с физическим каналом, минимальное значение зависит от типа измерительного устройства и от параметров калибровочной и градуировочной функций. Свойство доступно для записи только для виртуальных тегов.

№	Наименование	Доступ	Тип значения	Описание
8	TAGPROP_MAXVALUE	чтение и запись	VT_R4	Максимальное значение данных, которое может быть получено от тега. Для тега, связанного с физическим каналом, максимальное значение зависит от типа измерительного устройства и от параметров калибровочной и градуировочной функций. Свойство доступно для записи только для виртуальных тегов.
9	TAGPROP_TARE	чтение и запись	VT_UNKNOWN	Ссылка на интерфейс IUnknown объекта градуирвочной функции тега.
10	TAGPROP_OWNER	чтение и запись	VT_UNKNOWN	Ссылка на интерфейс IRecorderPlugin объекта plug-in`a, создателя тега. Свойство имеет корректное значение только для виртуальных тегов, в том случае если оно было установлено объектом plug-in`a.
11	TAGPROP_MODULE	чтение	VT_UNKNOWN	Ссылка на интерфейс IUnknown объекта Module, того объекта, которому принадлежит физический канал тега. Свойство имеет корректное значение только для тегов, связанных с физическим каналом.
12	TAGPROP_DEVSIGNALTYPE	чтение	VT_I4	Тип сигнала, свойство, предназначенное только для использования объектами ядра ПО «Recorder».
13	TAGPROP_UNITS	чтение и запись	VT_BSTR	Строковое обозначение единиц измерения сигнала, данные которого хранятся в теге.
14	TAGPROP_RECENABLED	чтение	VT_BOOL	Признак сохранения данных тега в файл, соответствует признаку регистрации данных группы, которой принадлежит тег.
15	TAGPROP_DESCRIBE	чтение и запись	VT_BSTR	Строка описания тега
16	TAGPROP_MODULENAME	чтение	VT_BSTR	Строка наименования аппаратного модуля, которому принадлежит физический канал тега. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
17	TAGPROP_MODULESERNUM	чтение	VT_I4	Серийный номер аппаратного модуля, которому принадлежит физический канал тега. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
18	TAGPROP_DEVTARE	чтение и запись	VT_UNKNOWN	Ссылка на интерфейс IUnknown объекта градуировочной функции тега. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.

№	Наименование	Доступ	Тип значения	Описание
19	TAGPROP_MINUSERVALUE	чтение и запись	VT_R4	Минимальное значение в шкале осциллограммы при отображении данных тега.
20	TAGPROP_MAXUSERVALUE	чтение и запись	VT_R4	Максимальное значение в шкале осциллограммы при отображении данных тега.
21	TAGPROP_AUTORANGE	чтение и запись	VT_BOOL	Признак автоматического расчета границ шкалы осциллограммы. При автоматическом расчете границ осциллограмм свойства TAGPROP_MINUSERVALUE, TAGPROP_MAXUSERVALUE имеют значения равные значениям свойств TAGPROP_MINVALUE и TAGPROP_MAXVALUE соответственно.
22	TAGPROP_IRECORDER	чтение	VT_UNKNOWN	Ссылка на интерфейс IUnknown объекта Recorder.
24	TAGPROP_VIEWRANGE	запись		Диапазон отображения, в процентах от диапазона измерения
26	TAGPROP_MAXXVIEWRANGE	чтение и запись	VT_R4	Максимальный диапазон отображения по оси X
27	TAGPROP_MODCHANNUMBER	чтение	VT_I4	Номер физического канала тега в модуле (порядковый номер в измерительном устройстве). Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
28	TAGPROP_DEFAULTESTIMATOR	чтение и запись	VT_UI4	Оценочная функция по умолчанию для отображения в цифровом формуляре
29	TAGPROP_DEVCHAN	чтение	VT_UI4	Уникальный цифровой идентификатор физического канала, связанного с тегом. Этот идентификатор действителен только на время работы ПО «Recorder», уникален в рамках одной программы. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
30	TAGPROP_DEVCHANINFO	чтение	VT_BSTR	Строка описания аппаратного измерительного устройства, канал которого связан с тегом. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
31	TAGPROP_FREQCOUNT	чтение	VT_I4	Количество допустимых значений частоты дискретизации тега, зависит от свойств физического канала тега. Свойство имеет корректное значение только для тегов, связанных с аппаратным каналом.
32	TAGPROP_HASTARE	чтение	VT_BOOL	Признак наличия у тега градуировочной функции.

A.7 Интерфейс ISignal

A.7.1 Интерфейс ISignal

A.7.1.1 ISignal::GetX

double GetX(int index) – Функция получения времени по индексу значения. В качестве результата функция возвращает промежуток времени прошедший от момента получения первого значения в буфере до момента получения значения с индексом index.

A.7.1.2 ISignal::GetY

double GetY(int index, bool viaTransformer) – Функция получения значения Y по индексу. В качестве результата функция возвращает значение из буфера по индексу.

A.7.1.3 ISignal::GetYX

double GetYX(double x, int pow) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.4 ISignal::SetX

void SetX(int index, double x) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.5 ISignal::SetY

void SetY(int index, double y, bool viaTransformer) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.6 ISignal::GetSize

int GetSize() - Функция получения количества данных в буфере. В качестве результата функция возвращает размер буфера.

A.7.1.7 ISignal::Resize

void Resize(int size) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.8 ISignal::GetTransformerType

int GetTransformerType() – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.9 ISignal::GetTransformer

void* GetTransformer() – Функция получения ссылки на интерфейс IUnknown объекта функции ГФ тега.

A.7.1.10 ISignal::SetTransformer

void SetTransformer(void* newtr) – Функция установки объекта ГФ для тега.

A.7.1.11 ISignal::GetDataType

int GetDataType() – Функция получения кода типа данных буфера, допустимыми значениями являются: VT_I2, VT_R4, VT_R8 и т.п.

A.7.1.12 ISignal::GetDeltaX

double GetDeltaX(void) – Функция получения периода измерения значений из буфера.

A.7.1.13 ISignal::SetDeltaX

void SetDeltaX(double deltaX) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.14 ISignal::GetStartX

double GetStartX(void) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.15 ISignal::SetStartX

void SetStartX(double startX) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.16 ISignal::GetStartTime

void GetStartTime(LPSYSTEMTIME lpStartTime) – Функция получения времени перехода в режим «Измерение» либо в режим «Регистрация». Функция возвращает локальное компьютерное время.

A.7.1.17 ISignal::GetTimeCounter

DWORD GetTimeCounter(void) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.18 ISignal::GetRangeX

void GetRangeX(double* minX, double* maxX) – Функция получения длительности сигнала, хранимого в буфере. Значение, возвращаемое по адресу из параметра minX, всегда равно 0. Значение, возвращаемое по адресу из параметра maxX и есть длительность сигнала.

A.7.1.19 ISignal::GetMinMax

int GetMinMax(double* min, double* max, void (*pNotify)(int)) – Функция получения минимального и максимального значения данных в буфере.

A.7.1.20 ISignal::SearchMinMax

int SearchMinMax(void (*pNotify)(int)) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.21 ISignal::IsMinMaxCalculated

bool IsMinMaxCalculated() – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.22 ISignal::GetK1K0

void GetK1K0(double* k1, double* k0) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.23 ISignal::SetK1K0

void SetK1K0(double k1 = 1, double k0) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.24 ISignal::SetSName

void SetSName(const char* newSName) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.25 ISignal::GetNameX

const char* GetNameX()– Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.26 ISignal::SetNameX

void SetNameX(const char* newNameX) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.27 ISignal::GetNameY

const char* GetNameY() – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.28 ISignal::SetNameY

void SetNameY(const char* newNameY) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.29 ISignal::GetComment

const char* GetComment() – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.30 ISignal::SetComment

void SetComment(const char* newComment) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.31 ISignal::GetCharacteristic

unsigned __int16 GetCharacteristic() – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.32 ISignal::SetCharacteristic

void SetCharacteristic(const unsigned __int16 newCharacteristic) – Данная функция в текущей версии ПО «Recorder» не реализована, оставлена для совместимости с ранними версиями.

A.7.1.33 ISignal::IndexOf

int IndexOf(double x) – Функция поиска номера значения наиболее близкого к значению x. В качестве результата функция возвращает индекс найденного значения.

A.7.1.34 ISignal::lockdata

int lockdata() – Функция блокирования буфера данных, для выполнения операций чтения или записи. В качестве результата функция возвращает количество блокирований вектора.

A.7.1.35 ISignal::unlockdata

int unlockdata() – Функция снятии блокировки буфера данных. В качестве результата функция возвращает количество блокирований вектора.

A.7.1.36 ISignal::lockcount

int lockcount() – Функция получения количество блокировок буфера данных. В качестве результата функция возвращает количество блокирований вектора.

A.8 Интерфейс IBlockAccess

A.8.1 Интерфейс IBlockAccess

A.8.1.1 IBlockAccess::GetReadyBlocksCount

ULONG GetReadyBlocksCount() – Получить количество блоков которые были записаны в буфер от начала режима «Измерение» или режима «Регистрация».

A.8.1.2 IBlockAccess::GetBlocksCount

ULONG GetBlocksCount() – Получить количество блоков, которые в данный момент находятся в буфере тега.

A.8.1.3 IBlockAccess::GetBlocksSize

ULONG GetBlocksSize() - Получить размер одного блока в элементах (измеренные значения).

A.8.1.4 IBlockAccess::GetVectorSize

ULONG GetVectorSize() – Получить полное количество значений в буфере.

A.8.1.5 IBlockAccess::GetVectorCapacity

ULONG GetVectorCapacity() – Получить полное количество значений которые могут быть записаны в буфер.

A.8.1.6 IBlockAccess::LockVector

ULONG LockVector() – Блокировать буфер для выполнения операции записи или чтения данных.

A.8.1.7 IBlockAccess::UnlockVector

ULONG UnlockVector() – Снять блокировку вектора.

A.8.1.8 IBlockAccess::GetLockCount

ULONG GetLockCount() – Получить количество блокирований буфера. Только один поток имеет возможность блокировать буфер несколько раз.

A.8.1.9 IBlockAccess::GetVectorR4

HRESULT GetVectorR4(float* a_pBlock, long int a_nFirstBlockIndex, long int a_nSize, BOOL a_fViaTranformer) – Чтение одного или нескольких блоков из буфера в область памяти, каждое значение данных представляет из себя четырехбайтовое действительное число.

Параметр	Описание
a_pBlock	Адрес области памяти, в которую будут прочтены данные из буфера.
a_nFirstBlockIndex	Индекс блока, начиная с которого будут прочтены данные.
a_nSize	Количество значений, которые необходимо прочесть.
a_fViaTranformer	Признак необходимости обработки данных вектора при помощи градуировочной функции тега.

Код результата	Описание
S_OK	Операция чтения выполнена успешно.
E_FAIL	Операция чтения завершена с ошибкой.

A.8.1.10 IBlockAccess::GetVectorI2

HRESULT GetVectorI2(short* a_pBlock, long int a_nFirstBlockIndex, long int a_nSize, BOOL a_fViaTranformer) – Чтение одного или нескольких блоков из буфера в область памяти, каждое значение данных представляет из себя двухбайтовое целое число со знаком.

Параметр	Описание
a_pBlock	Адрес области памяти, в которую будут прочтены данные из буфера.
a_nFirstBlockIndex	Индекс блока, начиная с которого будут прочтены данные.
a_nSize	Количество значений, которые необходимо прочесть.
a_fViaTranformer	Признак необходимости обработки данных вектора при помощи градуировочной функции тега.

Код результата	Описание
S_OK	Операция чтения выполнена успешно.
E_FAIL	Операция чтения завершена с ошибкой.

A.8.1.11 IBlockAccess::GetVectorU2

HRESULT GetVectorU2(unsigned short* a_pBlock, long int a_nFirstBlockIndex, long int a_nSize, BOOL a_fViaTranformer) – Чтение одного или нескольких блоков из буфера в область памяти, каждое значение данных представляет из себя двухбайтовое целое число без знаком.

Параметр	Описание
a_pBlock	Адрес области памяти, в которую будут прочтены данные из буфера.
a_nFirstBlockIndex	Индекс блока, начиная с которого будут прочтены данные.
a_nSize	Количество значений, которые необходимо прочесть.
a_fViaTranformer	Признак необходимости обработки данных вектора при помощи градуировочной функции тега.

Код результата	Описание
S_OK	Операция чтения выполнена успешно.
E_FAIL	Операция чтения завершена с ошибкой.

A.8.1.12 IBlockAccess::GetVectorR8

HRESULT GetVectorR8(double* a_pBlock, long int a_nFirstBlockIndex, long int a_nSize, BOOL a_fViaTranformer) – Чтение одного или нескольких блоков из буфера в область памяти, каждое значение данных представляет из себя восьми байтовое действительное число.

Параметр	Описание
a_pBlock	Адрес области памяти, в которую будут прочтены данные из буфера.
a_nFirstBlockIndex	Индекс блока, начиная с которого будут прочтены данные.
a_nSize	Количество значений, которые необходимо прочесть.
a_fViaTranformer	Признак необходимости обработки данных вектора при помощи градуировочной функции тега.

Код результата	Описание
S_OK	Операция чтения выполнена успешно.
E_FAIL	Операция чтения завершена с ошибкой.

A.9 Интерфейс ITagsGroup

A.9.1 Интерфейс ITagsGroup.

A.9.1.1 ITagsGroup::GetName

HRESULT GetName(CHAR* a_pchName) – Получение строки имени объекта группы тегов.

Параметр	Описание
a_pchName	Адрес буфера, в который будет переписана строка с именем объекта группы тегов. Максимальный размер строки с именем группы 255 символов.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.2 ITagsGroup::SetName

HRESULT SetName(CHAR* a_pchName) – Переименование объекта группы тегов.

Параметр	Описание
a_pchName	Адрес строки с новым именем группы тегов

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.3 ITagsGroup::AddTag

HRESULT AddTag(ITag* a_pTag) – Добавление тега в группу. Данная функция является внутренней, непосредственный вызов её из plug-in`ов недопустим.

Параметр	Описание
a_pTag	Ссылка на интерфейс ITag объекта тега, который необходимо добавить в группу.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.4 ITagsGroup::RemoveTag

HRESULT RemoveTag(ITag* a_pTag) – Удаление тега из группы. Данная функция является внутренней, непосредственный вызов её из plug-in`ов недопустим.

Параметр	Описание
ITag	Ссылка на интерфейс ITag объекта тега, который необходимо удалить из группы.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.5 ITagsGroup::GetTagsCount

ULONG GetTagsCount() Получить общее количество тегов в группе.
Результатом выполнения функции является количество тегов в группе.

A.9.1.6 ITagsGroup::GetTagByIndex

HRESULT GetTagByIndex(ITag** a_pTag, ULONG a_nIndex) – Получить ссылку на интерфейс ITag объекта тега по индексу тега в группе. Теги в группе индексируются, начиная с 0.

Параметр	Описание
a_pTag	Адрес переменной, в которую функция вернет ссылку на интерфейс тега.
a_nIndex	Индекс требуемого тега.

Код результата	Описание
S_OK	Операция выполнена успешно.
RPC_X_ENUM_VALUE_OUT_OF_RANGE	Значение индекса превышает допустимое

A.9.1.7 ITagsGroup::GetTagByName

HRESULT GetTagByName(ITag** a_pTag, CHAR* a_pchName) – Получить ссылку на интерфейс ITag объекта тега по имени. В данной версии ПО «Recorder» эта функция не реализована.

Параметр	Описание
a_pTag	Адрес переменной, в которую функция вернет ссылку на интерфейс тега.
a_pchName	Адрес строки с именем тега

Код результата	Описание
E_NOTIMPL	В данной версии ПО «Recorder» функция не реализована

A.9.1.8 ITagsGroup::StartRec

HRESULT StartRec() – Запуск процесса записи измеренных данных для тегов отдельно взятой группы, без останова общего процесса измерения. Для успешного выполнения этой функции ПО «Recorder» должно находится в режиме «Измерение» либо «Регистрация».

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.9 ITagsGroup::StopRec

HRESULT StopRec() – Останов процесса регистрации измеренных данных для отдельно взятой группы, без останова общего процесса измерения.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.10 ITagsGroup::GetRecEnabled

BOOL GetRecEnabled() – Получить признак необходимости регистрации измеренных данных группы в режиме «Регистрация». В качестве результата метод возвращает значение признака необходимости регистрации данных для группы.

A.9.1.11 ITagsGroup::SetRecEnabled

void SetRecEnabled(BOOL a_bValue) – Установить признак необходимости регистрации измеренных данных группы в режиме «Регистрация».

Параметр	Описание
a_bValue	Признак необходимости регистрации данных группы.

A.9.1.12 ITagsGroup::Notify

HRESULT Notify(DWORD a_dwCommand, DWORD a_dwParam) – Передать объекту группы сообщение.

Параметр	Описание
a_dwCommand	Код команды сообщения. В данной версии ПО «Recorder» не предусмотрено команд, которые можно было бы передать группе тегов.
a_dwParam	Дополнительный параметр для передачи данных вместе с сообщением.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, либо сообщение не было обработано.

A.9.1.13 ITagsGroup::SetProperty

HRESULT SetProperty(DWORD a_dwPropertyID, VARIANT a_Value) – Установить значение свойства объекта группы тегов, по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства
a_Value	Новое значение свойства

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.1.14 ITagsGroup::GetProperty

HRESULT GetProperty(DWORD a_dwPropertyID, VARIANT& a_Value) – Получить значение свойство группы тегов по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства
a_Value	Переменная, в которой функция вернет значение свойства

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.9.2 Иидентификаторы свойств групп тегов

№	Наименование	Доступ	Тип значения	Описание
1	TGRPROP_WRITETOPERSONALFRAME	чтение и запись	VT_BOOL	Признак необходимости сохранения файлов с измеренными данными группы в отдельном каталоге.
2	TGRPROP_PERSONALFRAMENAME	чтение и запись	VT_BSTR	Имя специализированного каталога, в котором должны располагаться файлы с измеренными данными.
3	TGRPROP_ENABLEREC	-	-	Признак необходимости регистрации измеренных данных группы. В текущей версии ПО «Recorder» это свойство доступно только через методы GetRecEnabled(),SetRecEnabled.
4	TGRPROP_DESCRIBE	чтение и запись	VT_BSTR	Строка описания объекта группы тегов.

A.10 Интерфейс IModule

A.10.1 Интерфейс IModule

A.10.1.1 IModule::GetType

WORD GetType() – Получить код типа аппаратного устройства.
В качестве результата функция возвращает код тип измерительного или исполнительного устройства.

A.10.1.2 IModule::_GetProperty

HRESULT _GetProperty(DWORD a_dwPropertyID, VARIANT& a_Value) - Получить значение свойства объекта по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства.
a_Value	Переменная, в которую функция вернет значение свойства.

Код результата	Описание
S_OK	Значение свойства корректно получено.
E_NOTIMPL	Свойство не поддерживается данным объектом, возможна ошибка с кодом свойства.

A.10.1.3 IModule::_SetProperty

HRESULT _SetProperty(DWORD a_dwPropertyID, VARIANT a_Value) – Установить значение свойства по идентификатору свойства.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо изменить.
a_Value	Новое значение свойства, которое необходимо установить.

Код результата	Описание
S_OK	Значение свойства корректно установлено.
E_NOTIMPL	Свойство не поддерживается данным объектом, возможна ошибка с кодом свойства.

A.10.2 Коды идентификаторов свойств

№	Наименование	Описание
1	MDPROP_SLOT	Номер слота крейт контроллера, в котором установлен аппаратный модуль.
2	MDPROP_TYPE	Код типа измерительного модуля.
3	MDPROP_SERIALNUMBER	Серийный номер измерительного модуля.
4	MDPROP_VERSIONMAJOR	Старший номер версии измерительного модуля.

№	Наименование	Описание
5	MDPROP_VERSIONMINOR	Младший номер версии измерительного модуля.

A.10.3 Коды типов измерительных и исполнительных устройств

№	Наименование	Описание
1	MOD_MC114_TYPE	Модуль MC114
2	MOD_MC201_TYPE	Модуль MC201
3	MOD_MC212_TYPE	Модуль MC212
4	MOD_MC227_TYPE	Модуль MC227
5	MOD_LC301_TYPE	Модуль LC301
6	MOD_MC451_TYPE	Модуль MC451
7	MOD_MC451V1_TYPE	Модуль MC451V1
8	MOD_MC451GV1_TYPE	Модуль MC451GV1
9	MOD_LC101_TYPE	Модуль LC101
10	MOD_LC102H_TYPE	Модуль LC102H
11	MOD_LC102HT_TYPE	Модуль LC102HT
12	MOD_LC102C_TYPE	Модуль LC102C
13	MOD_MC227C_TYPE	Модуль MC227C
14	MOD_MC227K_TYPE	Модуль MC227K
15	MOD_MC227U_TYPE	Модуль MC227U
16	MOD_MC227T_TYPE	Модуль MC227T
17	MOD_MC227W_TYPE	Модуль MC227W
18	MOD_MC227R_TYPE	Модуль MC227R
19	MOD_MC201A_TYPE	Модуль MC201A
20	MOD_LC302_TYPE	Модуль LC302
21	MOD_MC401_TYPE	Модуль MC401
22	MOD_MC402_TYPE	Модуль MC402
23	MOD_MC403_TYPE	Модуль MC403
24	MOD_MC801_TYPE	Модуль MC801
25	MOD_MAC_TYPE	Модуль MAC
26	MOD_MC114W31_TYPE	Модуль MC114W31
27	MOD_MC212W60_TYPE	Модуль MC212W60
28	MOD_LC227C_TYPE	Модуль LC227C
29	MOD_LC227K_TYPE	Модуль LC227K
30	MOD_LC227U_TYPE	Модуль LC227U
31	MOD_LC227T_TYPE	Модуль LC227T
32	MOD_LC227W_TYPE	Модуль LC227W
33	MOD_LC227R_TYPE	Модуль LC227R
34	MOD_MC227UP_TYPE	Модуль MC227UP
35	M2408_TYPE	Плата M2408
36	M2081_TYPE	Плата M2081
37	M2181_TYPE	Плата M2181
38	M2070_TYPE	Плата M2070
39	M20100_TYPE	Плата M20100
40	M21156_TYPE	Плата M21156

A.11 Интерфейс IDigitalOutput.

A.11.1 Интерфейс IDigitalOutput

A.11.1.1 IDigitalOutput::OutputWord

HRESULT OutputWord(DWORD a_dwValue) – Вывод цифрового значения.

Параметр	Описание
a_dwValue	Значение, которое необходимо выдать в исполнительное устройство.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.12 Интерфейс IDigitalInput

A.12.1 Интерфейс IDigitalInput

A.12.1.1 IDigitalInput::InputWord

HRESULT InputWord(DWORD* a_pdwValue) – Прочесть значение с цифрового входа.

Параметр	Описание
a_pdwValue	Переменная, в которую будет прочитана информация с цифрового входа.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13 Интерфейс IAnalogOutput

A.13.1 Интерфейс IAnalogOutput

A.13.1.1 IAnalogOutput::LoadPlayDACData

HRESULT LoadPlayDACData(DWORD size, WORD* data) – Передача устройству блока данных для генерации сигнала.

Параметр	Описание
size	Размер блока данных, в значениях.
data	Адрес блок с данными, для генерации сигнала.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.2 IAnalogOutput::StopPlayDAC

HRESULT StopPlayDAC() – Остановить процесс генерации сигнала.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.3 IAnalogOutput::StartPlayDAC

HRESULT StartPlayDAC(double* freq) – Запустить генерацию сигнала.

Параметр	Описание
freq	Частота дискретизации для генерируемого сигнала.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.4 IAnalogOutput::GetPlayDACDataMaxCount

HRESULT GetPlayDACDataMaxCount(DWORD* count) – Получить максимальный объем данных, которые можно использовать для генерации сигнала.

Параметр	Описание
count	Адрес переменной, в которую функция вернет размер буфера для данных.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.5 IAnalogOutput::GetPlayDACCodeRange

HRESULT GetPlayDACCodeRange(long *min, long *max) – Получить минимум и максимум генерируемого сигнала в кодах.

Параметр	Описание
min	Адрес переменной, в которую функция вернет минимальное значение.
max	Адрес переменной, в которую функция вернет максимальное значение.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.6 IAnalogOutput::GetPlayDACVoltRange

HRESULT GetPlayDACVoltRange(double *min, double *max) – Получить минимум и максимум генерируемого сигнала в вольтах.

Параметр	Описание
min	Адрес переменной, в которую функция вернет минимальное значение.
max	Адрес переменной, в которую функция вернет максимальное значение.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.13.1.7 IAnalogOutput::GetPlayDACFreqRange

HRESULT GetPlayDACFreqRange(double* min, double *max) – Получить диапазон доступных частот дискретизации.

Параметр	Описание
min	Адрес переменной, в которую функция вернет минимальное значение.
max	Адрес переменной, в которую функция вернет максимальное значение.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14 Интерфейс ICalibrator

A.14.1 Интерфейс ICalibrator

A.14.1.1 ICalibrator::Create

HRESULT Create() - Инициализировать объект калибровки.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14.1.2 ICalibrator::Run

HRESULT Run() - Запустить процесс калибровки. Предварительно в объект калибровки должны быть установлены: список тегов и параметры калибровки.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14.1.3 ICalibrator::AddTag

HRESULT AddTag(ITag* a_piTag) - Добавить тег в список для калибровки.

Параметр	Описание
a_piTag	Ссылка на интерфейс ITag объекта тега, который будет использоваться в процессе градуировки, калибровки или поверки.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14.1.4 ICalibrator::Save

ULONG Save(LPVOID a_lpvStream) - Сохранение настроек калибровки в область памяти.

Параметр	Описание
a_lpvStream	Адрес области памяти, в которую будут записаны параметры объекта автоматизированной калибровки.

Если параметр a_lpvStream содержит адрес области памяти, то в качестве результата функция возвращает количество записанных байт. Если параметр a_lpvStream равен 0, то функция возвращает требуемый объем памяти для сохранения настройки объекта автоматизированной калибровки.

A.14.1.5 ICalibrator::Load

ULONG Load(LPVOID a_lpvStream) - Чтение из области памяти настроек калибровки, если параметр a_lpvStream равен NULL, то чтение не производится.

Параметр	Описание
a_lpvStream	Адрес области памяти, из которой необходимо прочесть настройки объекта автоматизированной калибровки.

В качестве результата функция возвращает количество прочитанных байт.

A.14.1.6 ICalibrator::GetProperty

HRESULT GetProperty(DWORD a_dwPropID, VARIANT& a_varProp) - Получить значение свойства объекта калибровки по идентификатору свойства.

Параметр	Описание
a_dwPropID	Идентификатор свойства, значение которого необходимо получить.
a_varProp	Переменная, в которую функция вернет значение свойства.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14.1.7 ICalibrator::SetProperty

HRESULT SetProperty(DWORD a_dwPropID, VARIANT a_varProp) - Установить значение свойства объекта калибровки по идентификатору свойства.

Параметр	Описание
a_dwPropID	Идентификатор свойства, значение которого необходимо изменить.
a_varProp	Переменная, которая содержит новое значение для свойства.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.14.2 Идентификаторы свойств объекта калибровки

№	Наименование	Доступ	Тип значения	Описание
1	CLBPROP_FLAGS	чтение и запись	VT_I4	Флаги процесса градуировки, калибровки и поверки. ПО «Recorder» устанавливает в объект автоматизированной калибровки один из флагов CF_DEVICECALIBR, CF_TAGCALIBR.

A.14.3 Коды флагов автоматизированной градуировки, калибровки и поверки

Значения битовых флагов калибровки.

№	Наименование	Описание
1	CF_DEVICECALIBR	Выполнить калибровку чувствительности для модуля, либо поверку.
2	CF_TAGCALIBR	Выполнить градуировку для тега, либо поверку.

A.15 Интерфейс ITransformer

Интерфейс ITransformer является базовым для всех объектов градуировочных функций (КФ/ГФ).

A.15.1 Интерфейс ITransformer

A.15.1.1 ITransformer::GetEditor

HANDLE GetEditor() – Получить идентификатор окна редактора свойств объекта КФ/ГФ. Данная функция в текущей версии ПО «Recorder» не используется, зарезервирована для будущих проектов.

В качестве результата функция возвращает идентификатор окна редактора свойств.

A.15.1.2 ITransformer::GetEmbeddedEditor

HANDLE GetEmbeddedEditor(HANDLE a_pParent) – Получить идентификатор встроенного окна редактора свойств объекта КФ/ГФ. Встроенное окно не имеет рамки и не имеет заголовка. Встроенное окно должно являться частью какого-либо диалогового окна редактора свойств.

Параметр	Описание
a_pParent	Идентификатор окна, которое будет являться родительским для встроенного окна редактора.

В качестве результата функция возвращает идентификатор встроенного окна редактора свойств.

A.15.1.3 ITransformer::Save

int Save(HANDLE a_pStream) – Сохранить настройки объекта КФ/ГФ в область памяти.

Параметр	Описание
a_pStream	Адрес буфера памяти, в который необходимо записать настройки объекта.

Если параметр a_pStream содержит корректный адрес области памяти, то функция в качестве результата вернет количество записанных байт. Если параметр a_pStream имеет значение 0, то функция вернет требуемый размер области памяти для сохранения настроек объекта.

A.15.1.4 ITransformer::Load

int Load(HANDLE a_pStream) – Чтение настройки объекта КФ/ГФ из области памяти.

Параметр	Описание
a_pStream	Адрес области памяти, из которой необходимо прочесть настройки объекта.

В качестве результата функция возвращает количество прочитанных байт.

A.15.1.5 ITransformer::GetInfoStrings

int GetInfoStrings(int* a_pnIndex, char* a_pchBuffer, const char* a_pchPath, const char* a_pchTagName) – Получить описание объекта КФ/ГФ для формирования файла типа «*.mera». Данная функция возвращает строку, описывающую объект КФ/ГФ и при необходимости создает дополнительный файл, в котором сохраняются параметры объекта КФ/ГФ.

Параметр	Описание
a_pnIndex	Адрес переменной, которая должна содержать общий номер объекта ГФ/КФ, для которого выполняется получение описания. В результате выполнения функции значение по адресу a_pnIndex увеличивается на единицу.
a_pchBuffer	Адрес буфера для формирования описания. В результате выполнения данной функции буфер будет содержать строку, которую можно добавить в файл «*.mera». Строка описания будет содержать либо параметры объекта ГФ/КФ, либо имя файла, который содержит параметры объекта ГФ/КФ.
a_pchPath	Адрес строки с именем каталога, в котором формируется файл «*.mera».

Параметр	Описание
a_pchTagName	Наименование тега, которому принадлежит данный объект ГФ/КФ.

A.15.1.6 ITransformer::Edit

bool Edit() – Отобразить самостоятельный модальный диалог редактора свойств объекта КФ/ГФ.
В качестве результата функция возвращает признак того, что диалог был успешно отображен.

A.15.1.7 ITransformer::Eval

double Eval(double a_dblValue) – Произвести обработку измеренного значения при помощи функции объекта КФ/ГФ.

Параметр	Описание
a_dblValue	Значение, которое необходимо обработать при помощи КФ/ГФ.

В качестве результата функция возвращает обработанное значение.

A.15.1.8 ITransformer::GetTXType

TXTYPE GetTXType() - Получить идентификатор типа объекта КФ/ГФ.

В качестве результата функция возвращает код типа объекта КФ/ГФ, допустимым значением является: TXT_SCALE, TXT_LINE, TXT_INTEPOLATETABLE либо TXT_POLYNOME.

A.15.1.9 ITransformer::UpdateEditor

HRESULT UpdateEditor() – Обновить содержимое редактора свойств объекта КФ/ГФ.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.15.1.10 ITransformer::Reset

HRESULT Reset() – Сбросить состояние объекта КФ/ГФ. Параметры объекта инициализируются значениями по умолчанию.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.15.1.11 ITransformer::GetName

DWORD GetName(char* a_pchName) - Получить строку имени объекта КФ/ГФ.

Параметр	Описание
a_pchName	Адрес буфера, в который будет записано имя объекта.

Если параметр a_pchName содержит корректный адрес буфера, то функция возвращает количество переписанных в буфер символов. Если значение параметра a_pchName 0, то функция возвращает

A.15.1.12 ITransformer::SetName

HRESULT SetName(const char* a_pchName) - Установить наименование объекта КФ/ГФ.

Параметр	Описание
a_pchName	Адрес строки с именем объекта КФ/ГФ.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.15.1.13 ITransformer::ImportFromFile

HRESULT ImportFromFile(const char* a_pchName, const char* a_pchSeparator) - Функция импортирования настройки объекта КФ/ГФ из файла типа «.csv».

Параметр	Описание
a_pchName	Адрес строки с именем файла, из которого необходимо импортировать параметры объекта КФ/ГФ.
a_pchSeparator	Адрес строки с символом, который используется в файле для разделения значений параметров объекта КФ/ГФ.

Код результата	Описание
S_OK	Операция выполнена успешно, параметры объекта КФ/ГФ прочитаны из файла.
E_FAIL	Операция завершена с ошибкой.

A.15.1.14 ITransformer::ExportToFile

HRESULT ExportToFile(const char* a_pchName, const char* a_pchSeparator); - экспортировать настройки трансформера в файл типа «..csv».

Параметр	Описание
a_pchName	Адрес строки с именем файла, в который необходимо экспортировать параметры объекта КФ/ГФ.
a_pchSeparator	Адрес строки с символом, который будет используется в файле для разделения значений параметров объекта КФ/ГФ.

Код результата	Описание
S_OK	Операция выполнена успешно, параметры объекта КФ/ГФ сохранены в файл.
E_FAIL	Операция завершена с ошибкой.

A.15.1.15 ITransformer::GetProgID

LPCSTR GetProgID() – Функция получения строки с идентификатором COM объекта КФ/ГФ. В качестве результата функция возвращает адрес строки.

A.15.1.16 ITransformer::GetTypeName

DWORD GetTypeName(char* a_pchName) – Функция получения строки имени объекта КФ/ГФ.

Параметр	Описание
a_pchName	Адрес буфера для получения строки имени объекта КФ/ГФ.

Если значение параметра a_pchName равно 0, то функция возвращает требуемый размер буфера. Если a_pchName содержит корректный адрес буфера, то в этот буфер копируется строка с именем объекта КФ/ГФ, в качестве результата функция возвращает количество скопированных символов.

A.15.1.17 ITransformer::Check

HRESULT Check() – Функция проверки корректности значений параметров объекта КФ/ГФ. В текущей версии ПО «Recorder» данная функция зарезервирована для последующих использований.

Код результата	Описание
S_OK	Функция зарезервирована.

A.15.1.18 ITransformer::GetProperty

HRESULT GetProperty(DWORD a_dwPropertyID, VARIANT& a_Value) – Получить значение свойства по идентификатору свойства. В текущей версии ПО «Recorder» объекты функций ГФ/КФ не содержат специфических свойств.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо получить.
a_Value	Адрес переменной, в которую функция вернет значение свойства.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, значение свойства не получено.

A.15.1.19 ITransformer::SetProperty

HRESULT SetProperty(DWORD a_dwPropertyID, VARIANT a_Value) – Установить значение свойства по идентификатору свойства. В текущей версии ПО «Recorder» объекты функций ГФ/КФ не содержат специфических свойств.

Параметр	Описание
a_dwPropertyID	Идентификатор свойства, значение которого необходимо установить.
a_Value	Новое значение свойства.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, значение свойства не было из-

	менено.

A.15.1.20 ITransformer::Notify

bool Notify(DWORD a_dwCommand, DWORD a_dwParam) – Обработка команды или сообщения. В текущей версии ПО «Recorder» объекты функций ГФ/КФ не содержат специфических команд или сообщений.

Параметр	Описание
a_dwCommand	Идентификатор команды или сообщения
a_dwParam	Значение этого параметра специфично для каждой команды и сообщения.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, команда или сообщение не обработано, либо обработано с ошибкой.

A.15.1.21 ITransformer::TestImportFile

bool TestImportFile(LPCSTR a_pchName) – Проверить файл на возможность загрузки параметров объекта функции ГФ/КФ. В текущей версии ПО «Recorder» эта функция не реализована, зарезервирована для модернизации в будущих версиях.
В качестве результата функция возвращает значение false.

A.15.2 Коды типов градуировочных функций

Тип перечисление TXTYPE.

№	Наименование	Описание
1.	TXT_NULL	Не определен, отсутствует. Это тип предназначен только для внутреннего использования в ПО «Recorder».
2.	TXT_SCALE	Масштабный коэффициент.
3.	TXT_LINE	Линейная функция.
4.	TXT_INTEPOLATETABLE	Таблица линейной интерполяции.
5.	TXT_POLYNOME	Полином n-ой степени

A.16 Интерфейс IScale

A.16.1 Интерфейс IScale

A.16.1.1 IScale::PutScale

HRESULT PutScale(double a_dblValue) – Функция установки значения масштабного коэффициента.

Параметр	Описание
a_dblValue	Значение масштабного коэффициента

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, значение коэффициента не было изменено.

A.16.1.2 IScale::GetScale

HRESULT GetScale(double* a_pdblValue); - получить значение масштабного коэффициента.

Параметр	Описание
a_pdblValue	Адрес переменной, в которую функция вернет значение масштабного коэффициента.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой, значение коэффициента не было получено.

A.17 Интерфейс IInterpolate

A.17.1 Интерфейс IInterpolate

A.17.1.1 IInterpolate::SetNode

HRESULT SetNode(double a_dblValX, double a_dblValY, int a_nIndex) – Функция добавления строки в таблицу.

Параметр	Описание
a_dblValX	Значение X.
a_dblValY	Значение Y.
a_nIndex	Номер значения в таблице

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.17.1.2 IInterpolate::GetNode

HRESULT GetNode(double* a_pdblValX, double* a_pdblValY, int a_nIndex); - получение элемента из таблицы, функция возвращает значение x, y по номеру в таблице.

Параметр	Описание
a_pdblValX	Адрес переменной, в которую функция вернет значение X
a_pdblValY	Адрес переменной, в которую функция вернет значение Y
a_nIndex	Номер значения в таблице

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.17.1.3 IInterpolate::GetNodesCounter

ULONG GetNodesCounter() – Функция получения количества строк таблицы.

A.17.1.4 IInterpolate::SetExtrapolateMode

HRESULT SetExtrapolateMode(BOOL a_fExtrapolate) – Функция установки признака интерполяции за крайними точками таблицы.

Параметр	Описание
a_fExtrapolate	Значение признака необходимости экстраполяции

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.17.1.5 IInterpolate::GetExtrapolateMode

BOOL GetExtrapolateMode() – Функция получения признака экстраполяции за крайними точками таблицы.
В качестве результата функция возвращает значение признака.

A.18 Интерфейс ILinear

A.18.1 Интерфейс ILinear

A.18.1.1 ILinear::PutA

HRESULT PutA(double a_dblA) – Функция установки значения коэффициента a.

Параметр	Описание
a_dblA	Значение коэффициента a.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.18.1.2 ILinear::PutB

HRESULT PutB(double a_dblB) - Функция установки значения коэффициента b.

Параметр	Описание
	Значение коэффициента b.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.18.1.3 ILinear::GetA

HRESULT GetA(double* a_pdblA) – Функция получения значения коэффициента a.

Параметр	Описание

a_pdblA	Адрес переменной, в которую функция вернет значение коэффициента.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

A.18.1.4 ILinear::GetB

HRESULT GetB(double* a_pdblB) – Функция получения значения коэффициента b.

Параметр	Описание
a_pdblB	Адрес переменной, в которую функция вернет значение коэффициента.

Код результата	Описание
S_OK	Операция выполнена успешно.
E_FAIL	Операция завершена с ошибкой.

В Приложение. Схемы и рисунки