Руководство пользователя

gtlab

Программное обеспечение GTLd

Содержание

1. Принятые сокращения и определения

АФЧХ – амплитудно-фазовая частотная характеристика (APFC)

СКЗ – среднее квадратическое значение (RMS)

СКО – среднее квадратическое отклонение

ЛКМ – левая кнопка мыши

ПКМ – правая кнопка мыши

БПФ – быстрое преобразование Фурье (FFT)

АРМ – автоматизированное рабочее место

JS API – интерфейс прикладного программирования Java Script

2. Назначение программного обеспечения GTLd

Программное обеспечение GTLd предназначено для конфигурирования проектов диагностики с объектами любой разветвленной структуры, формирования маршрутов измерений параметров вибрации для внешних устройств, а также отладки методик диагностики и мониторинга с использованием собственного интерфейса JS API. Отличительной особенностью GTLd является возможность реализации проведения вибрационной диагностики и мониторинга состояния объектов с применением любой доступной методики (в том числе и авторской), используя широкий функционал JS API, документация по которому размещена на странице https://docs.gtlab.pro/index.php/Справочник_GTLD:JS-API, которая также постоянно дополняется по мере развития программного обеспечения. Дополнительно пользователь GTLd получает возможность проводить собственные исследования вибрационных сигналов.

3. Окно программы

3.1 Запуск программы

Для запуска программы GTLd необходимо открыть соответствующий ярлык на рабочем столе.

3.2 Окно программы «Создание и открытие проекта»

Окно «Создание и открытие проекта» служит пользователю интерфейсом для управления проекта в программе GTLd.

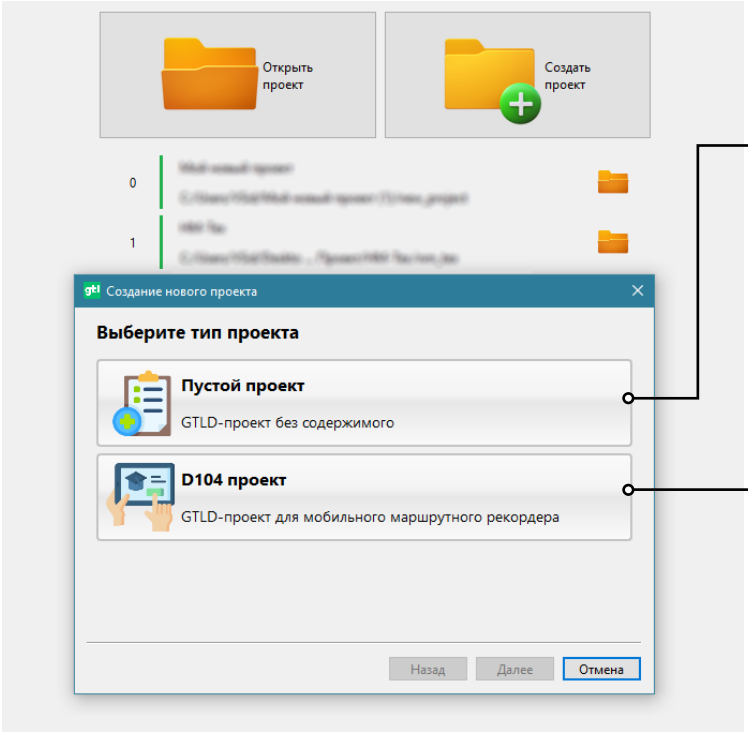
Открытие ранее созданного проекта
Позволяет пользователю быстро загружать и продолжать работу с уже существующими проектами

Создание нового проекта
Предоставляет возможность начать новый проект, используя доступные инструменты

История ранее открытых проектов
Упрощает навигацию между недавно использованными проектами, позволяя быстро возвращаться к ним

3.3 Создание нового проекта в GTLd

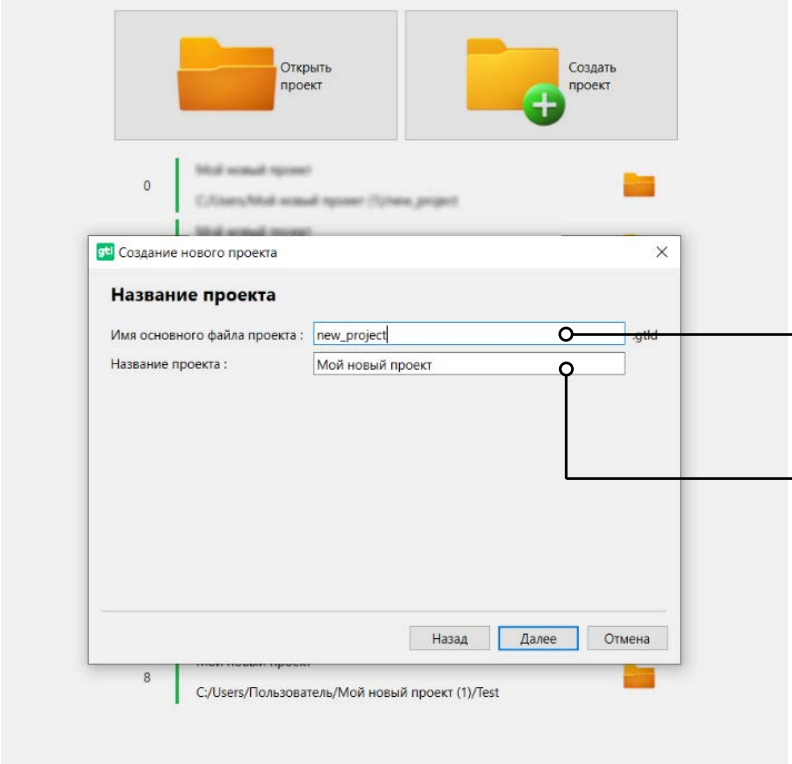
При нажатии кнопки «Создать новый проект» пользователю будет доступно окно «Выберите тип проекта».



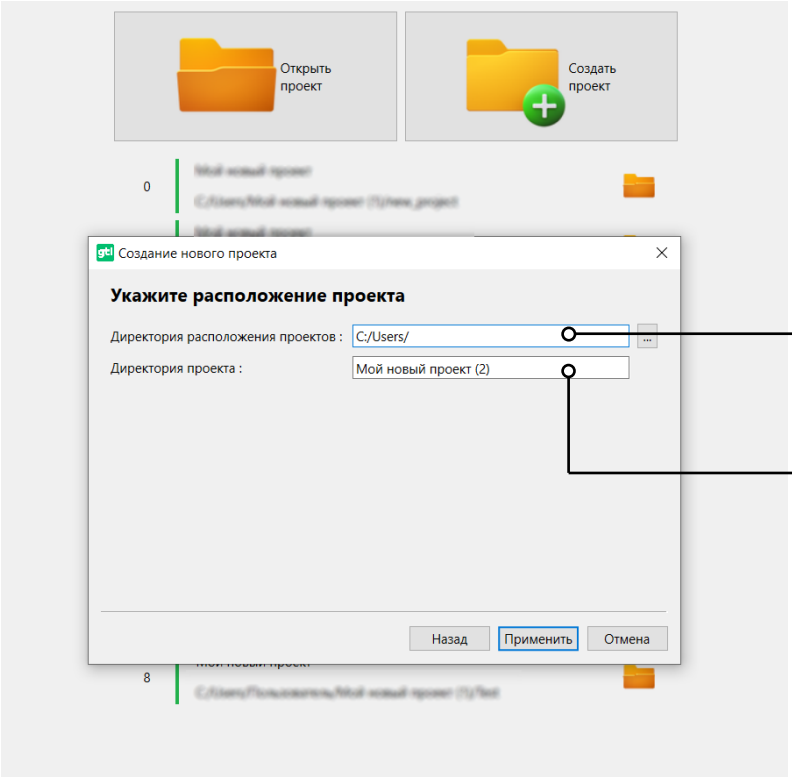
«Пустой проект»
Позволяет создать новый проект в GTLd без заранее заданного содержимого, предоставляя пользователю полную свободу для дальнейшей настройки и изменения

«D104 проект»
Создает проект GTLd, специально предназначенный для мобильного маршрутного рекордера. При этом автоматически добавляется одна группа, объект и «отправная» маршрутная точка, а также включается виброанализатор D104 в качестве измерительного устройства

При выборе любого типа проекта откроется окно «Название проекта», где пользователь сможет задать имя для нового проекта.



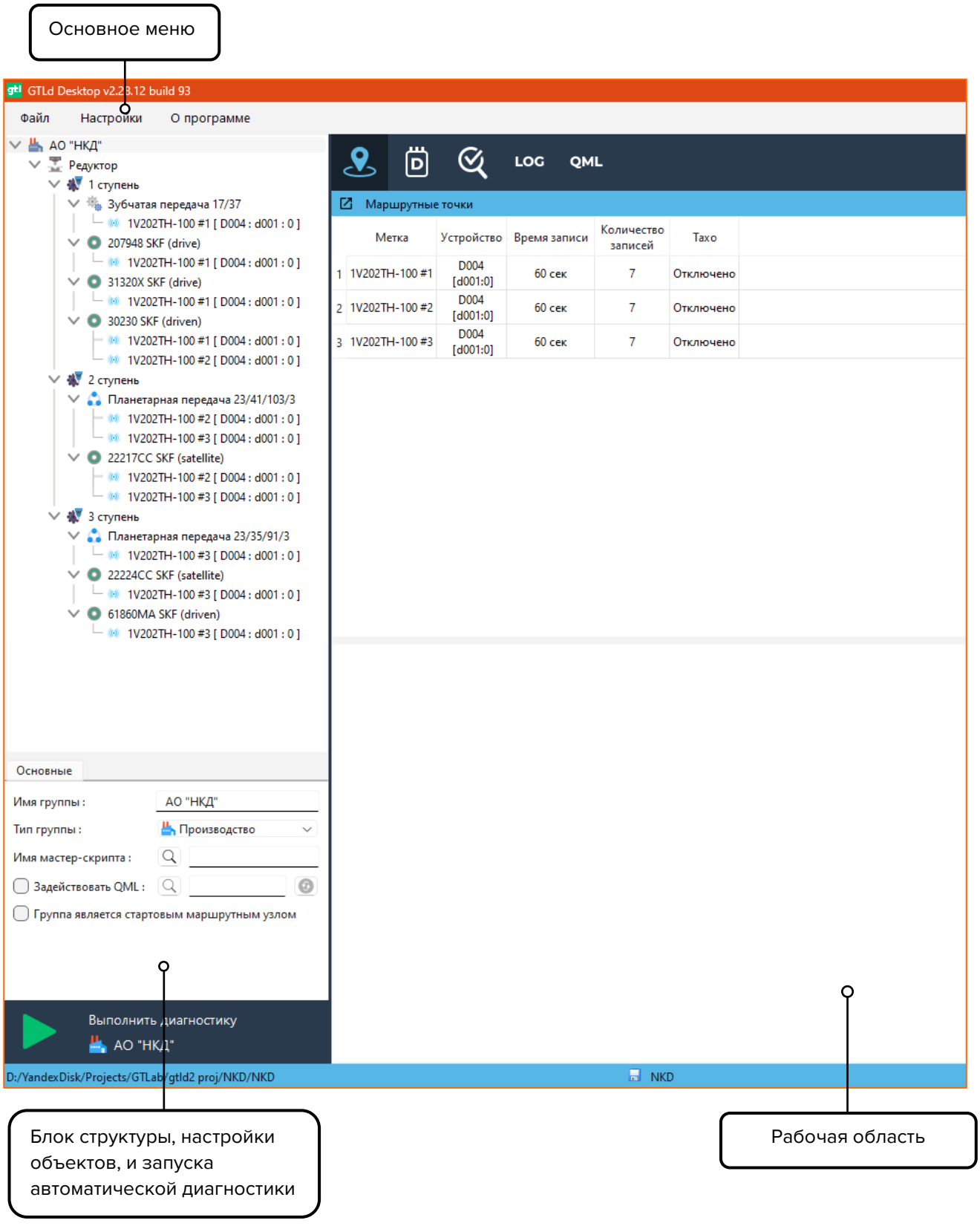
После ввода всех данных и нажатия кнопки «Далее» откроется окно «Выбор расположения проекта».



3.4 Основное окно программы

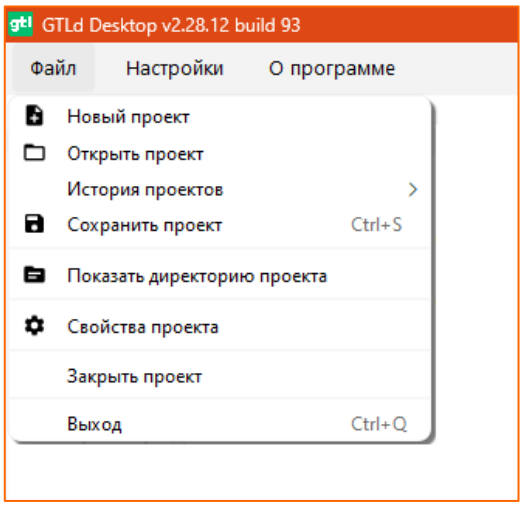
Основное окно программы состоит из нескольких блоков:

- основное меню;
- блок структуры, настройки объектов, и запуска автоматической диагностики;
- рабочая область.

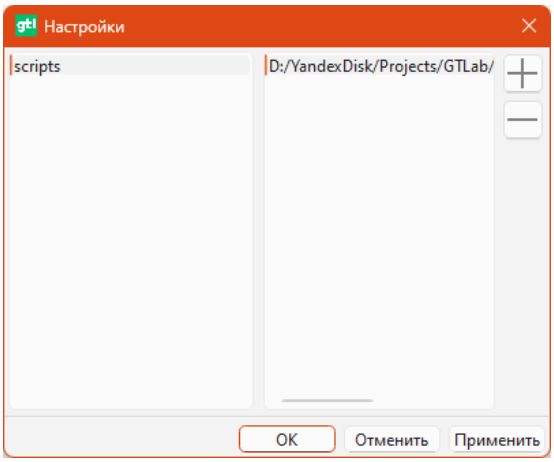


3.5 Основное меню

Основное меню предназначено для управления проектами: создание нового, открытие уже имеющегося, сохранение текущего, настройка и т.д.



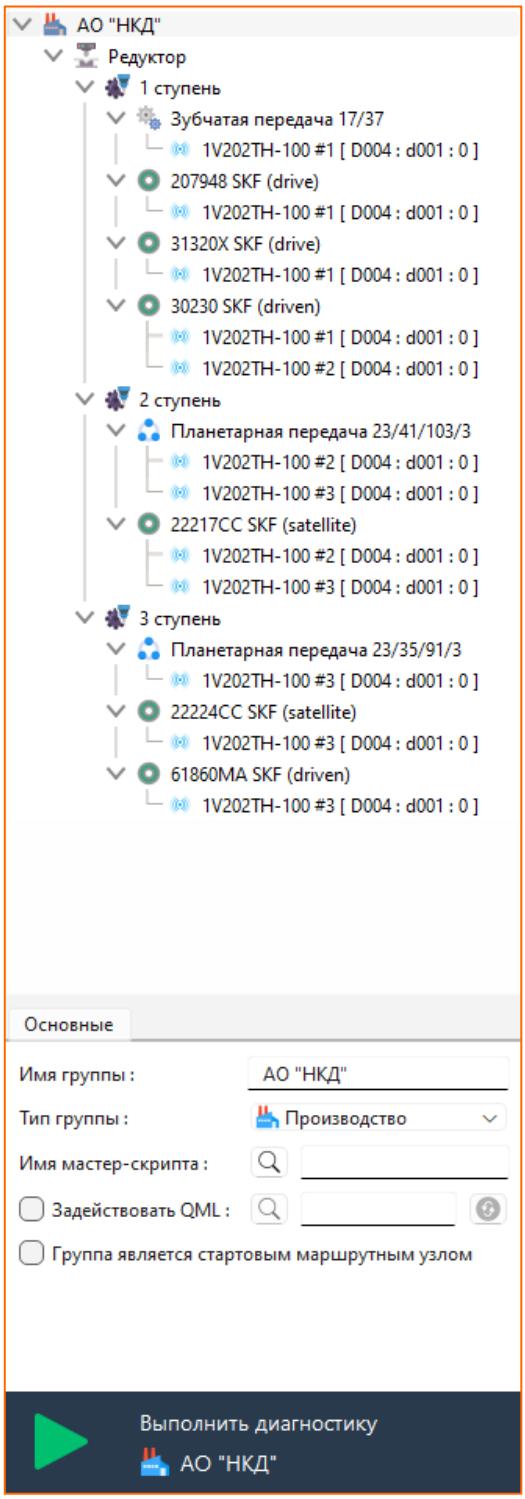
Пункт «Настройки» основного меню предназначен для добавления директорий, в которых хранятся файлы библиотек пользовательских функций, используемых в диагностических методиках.



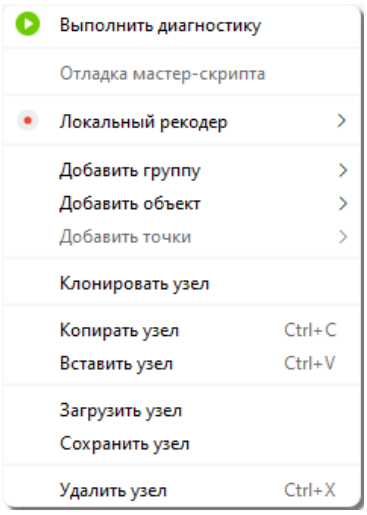
Пункт «О программе» основного меню выводит информационное окно с версией программного обеспечения.

3.6 Блок структуры и настройки объектов диагностики

Данный блок предназначен для формирования структуры объектов диагностирования, их настройки и управления запуском автоматической диагностики.

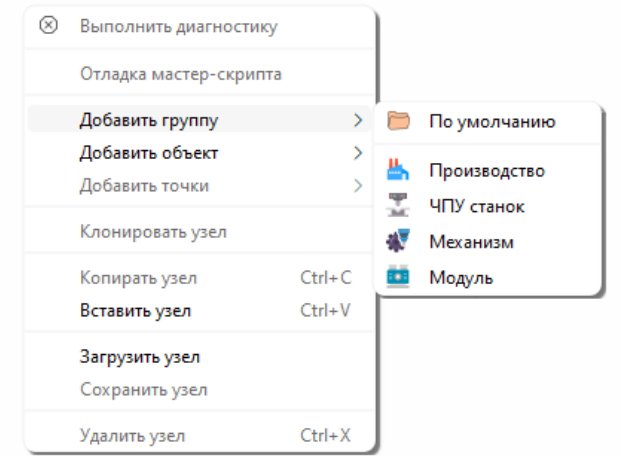


Создание древовидной структуры и управление ее элементами реализовано через контекстное меню при нажатии ПКМ на свободной области блока или элементе дерева. Доступность пунктов меню зависит от места его вызова (свободная область или структурный элемент дерева). Помимо стандартных команд для работы с узлами дерева, предусмотрено сохранение и загрузка структуры в формате .json, благодаря чему существует возможность создания набора стандартных структур объектов диагностирования и использование их в различных проектах.

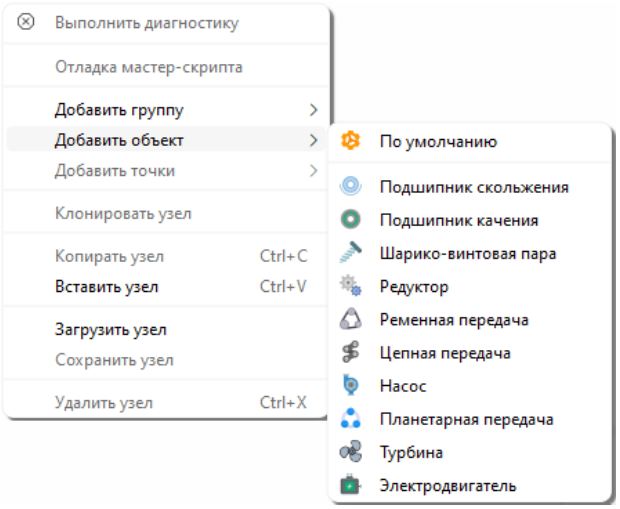


В программном обеспечении предусмотрено создание трехступенчатой структуры: «Группа» -> «Объект» -> «Точка», при этом группа может содержать множество подгрупп. Данный подход обеспечивает создание структуры любой сложности.

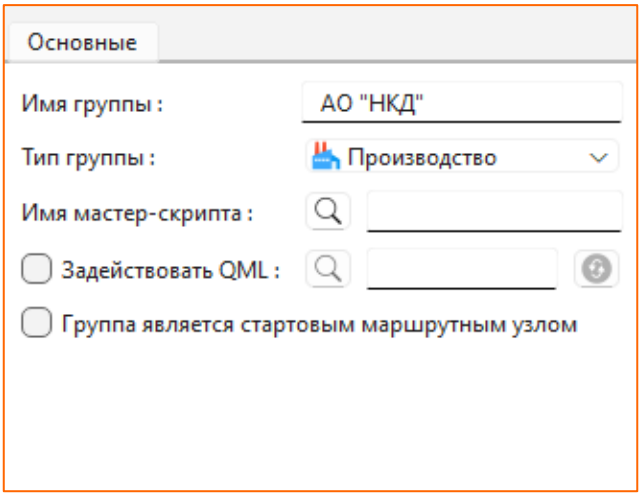
Для лучшей визуализации предусмотрено создание нескольких типов основных групп объектов с различными иконками.



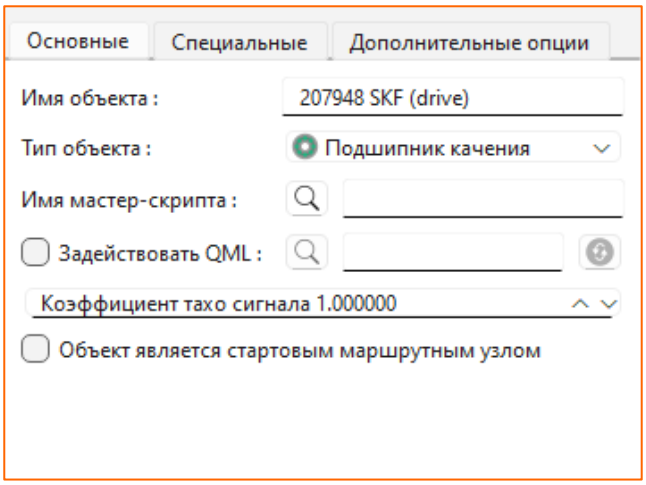
Основной диагностической единицей в программном обеспечении GTLd является «Объект» диагностики. На данный момент реализовано диагностирование широкого набора объектов.



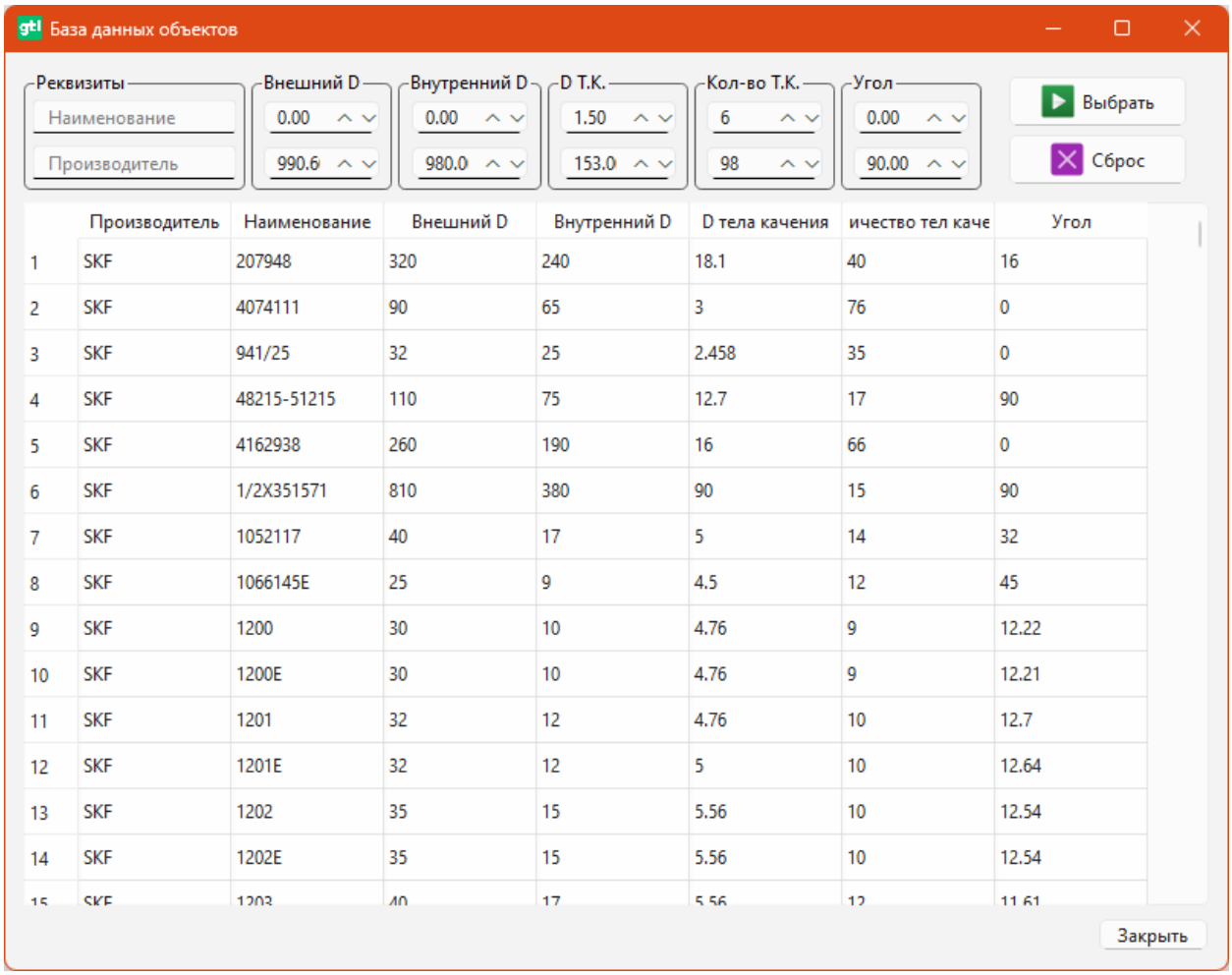
В нижней части структуры объекта диагностики расположен блок данных с настройками, содержание которого зависит от выбранного элемента дерева. Во вкладке «Основные» можно изменить имя, выбрать тип, указать файл мастер-скрипта, задействовать визуализацию на основе QML и указать стартовый узел маршрута измерений для внешних устройств. Кнопка предназначена для вызова предустановленного списка файлов скриптов.



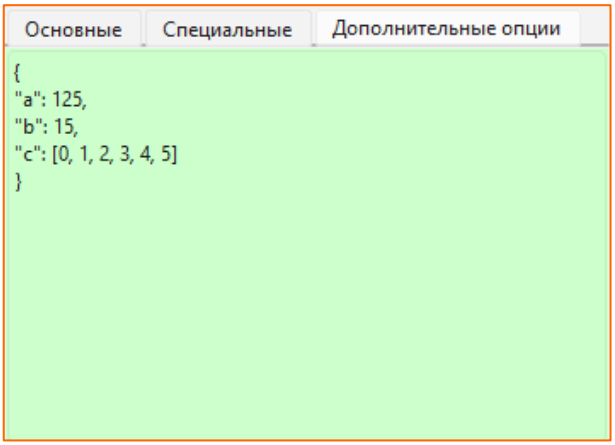
Для узлов типа «Объект» во вкладке «Основные» имеется дополнительное поле с настройкой коэффициента тахо сигнала, который обозначает коэффициент преобразования частоты вращения относительно вала, на котором установлен датчик оборотов. При данном подходе нет ограничения в каком месте объекта диагностирования установлен датчик оборотов и тахо сигнал можно получать как с ведущего вала многоуровневой системы привода, так и с любого их ведомых валов, рассчитав соответствующий коэффициент преобразования один раз на этапе формирования структуры механизма.



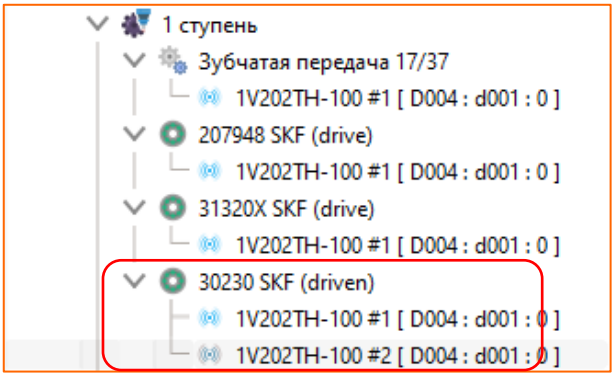
Вкладки «Специальные» и «Дополнительные опции» доступны только для узлов типа «Объект», содержание которой зависит от типа выбранного объекта диагностики. На рисунке приведены специальные настройки для объекта типа «Подшипник качения», где необходимо указать наименование, производителя и его основные геометрические параметры. Реализована возможность выбрать необходимый подшипник качения из готовой базы данных, нажав соответствующую кнопку [База данных] и воспользовавшись инструментом поиска по параметрам.



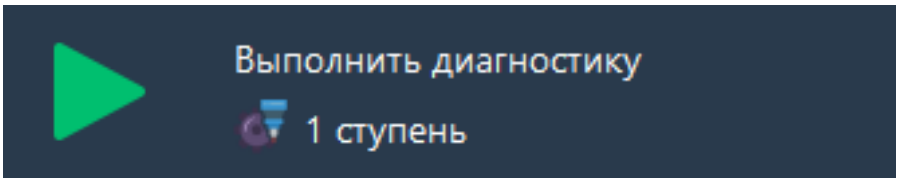
Вкладка «Дополнительные опции» предназначена для передачи пользовательских параметров в диагностический скрипт в виде объекта JSON. Необходима для отладки методик диагностики и балансировочного калькулятора. При формировании происходит автоматическая проверка синтаксиса: если ошибок не обнаружено, то поле ввода данных закрашивается в зеленый цвет, в противном случае - в красный.



Пункт контекстного меню «Добавить точки» доступен только при вызове его на узле дерева типа «Объект». «Точка» (маршрутная точка) является конечным элементом структуры, физически представляет собой точку (место) установки датчика вибрации и содержит в себе непосредственно данные вибрационных сигналов. К одному «Объекту» может быть привязано несколько «Точек». Подробнее о создании и настройке маршрутных точек написано в разделе 3.7.1 данного руководства.

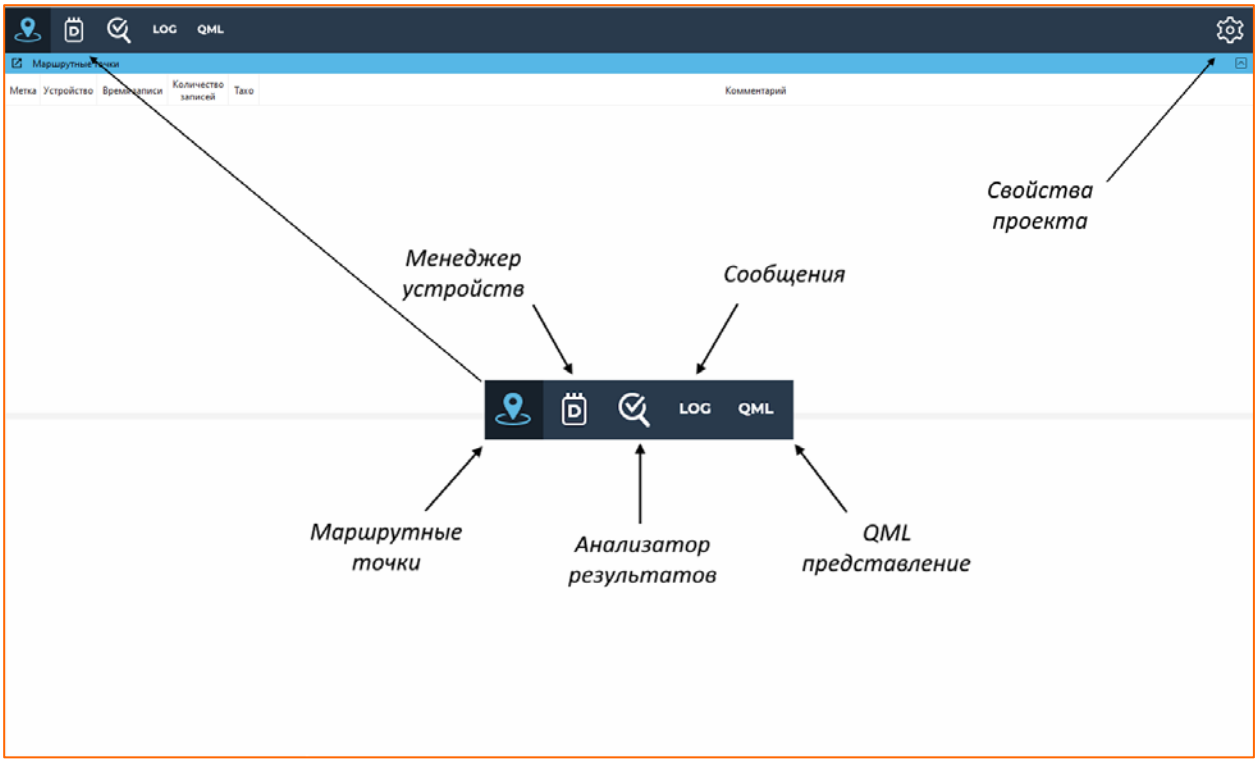


Ниже блока данных с настройками расположена кнопка запуска автоматической диагностики с указанием узла, для которого она будет запущена, при этом процесс диагностирования запускается для всех дочерних узлов (объектов) структуры.

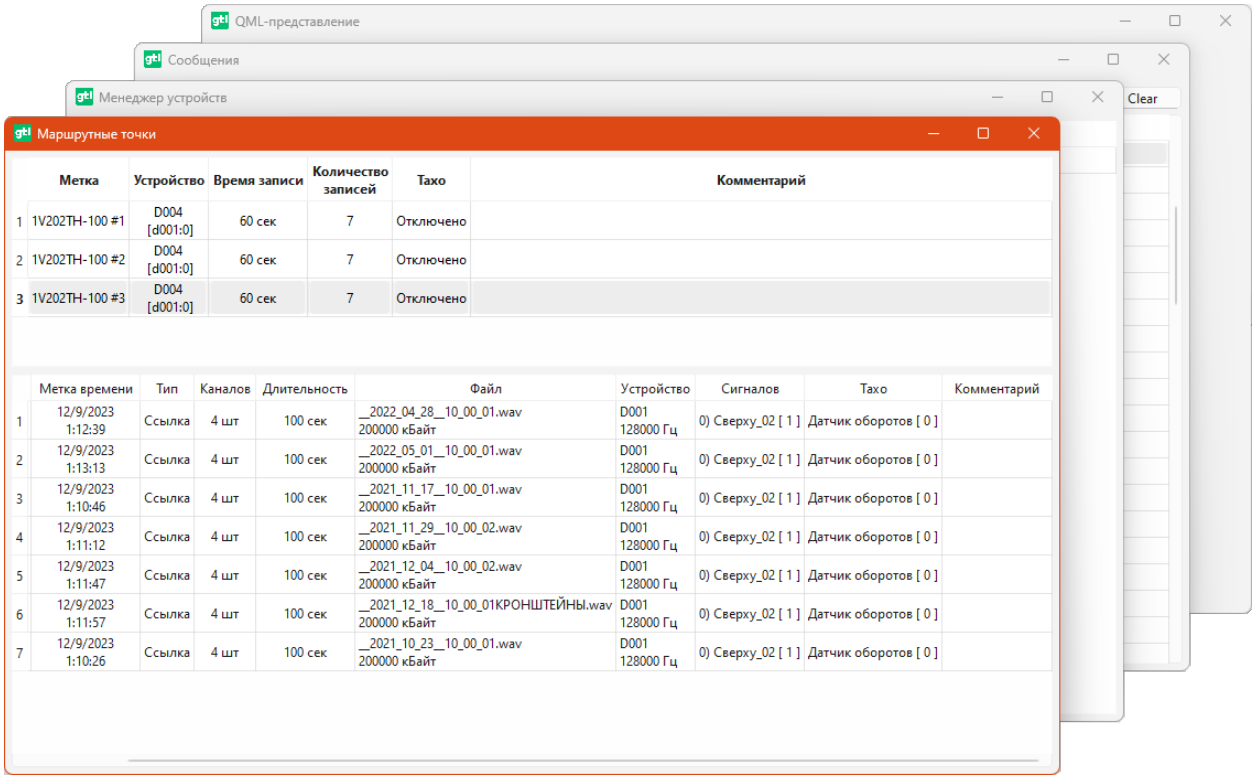


3.7 Рабочая область программы

Рабочая область предназначена для создания маршрутных точек, привязки записей вибрационных сигналов к ним, управления внешними устройствами, работы с результатами диагностики, ведения лога работы программы, управления визуализацией на основе QML и функционально разделена. Переключение функционала осуществляется соответствующими кнопками.



Архитектурой пользовательского интерфейса предусмотрено отделение функционала рабочей области в отдельные функциональные окна, что позволяет гибко настроить рабочее место специалиста на широкоформатном мониторе или на нескольких мониторах одновременно, распределяя весь функционал в каком угодно порядке. Отделение функциональных окон происходит по нажатию кнопки  , расположенной под панелью с кнопками переключения функционала. В некоторых случаях предусмотрено скрытие верхнего блока рабочей области при нажатии кнопки  .

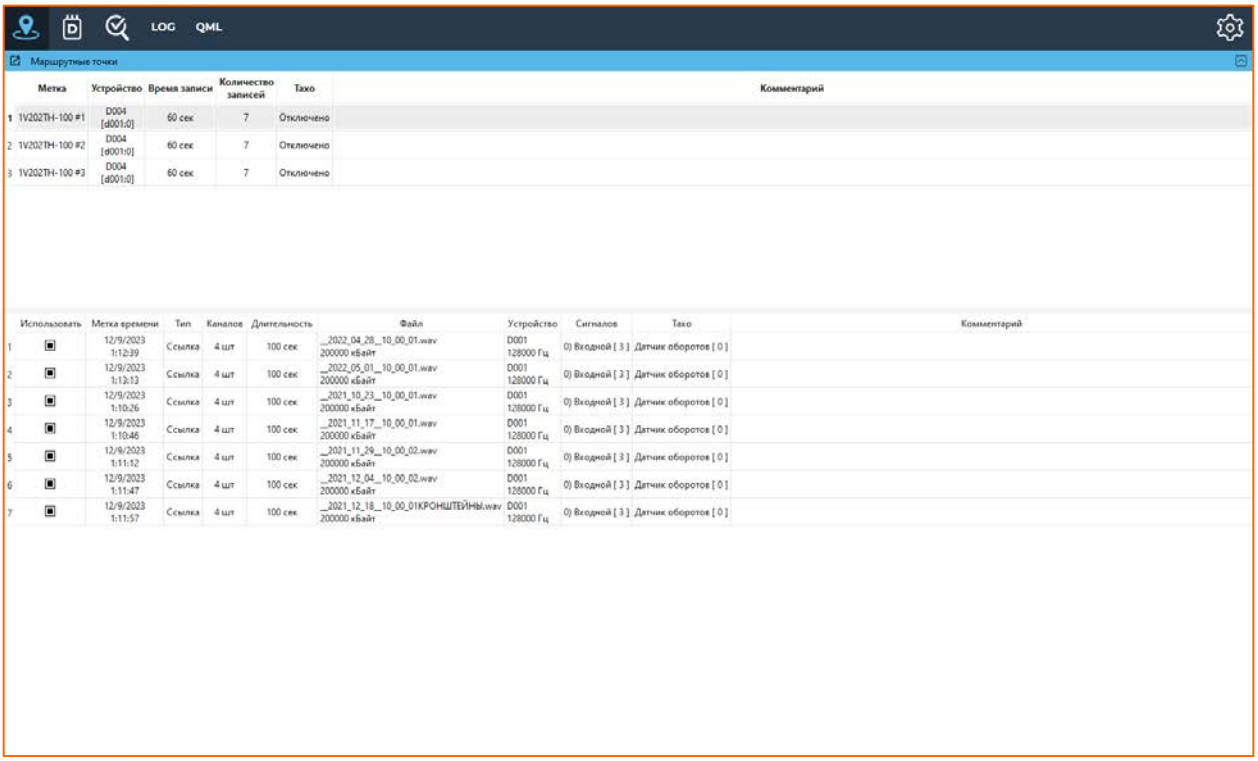


	Метка	Устройство	Время записи	Количество записей	Тахо	Комментарий
1	1V202TH-100 #1	D004 [d001:0]	60 сек	7	Отключено	
2	1V202TH-100 #2	D004 [d001:0]	60 сек	7	Отключено	
3	1V202TH-100 #3	D004 [d001:0]	60 сек	7	Отключено	

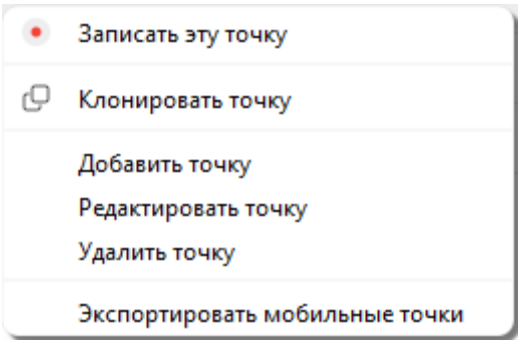
	Метка времени	Тип	Каналов	Длительность	Файл	Устройство	Сигналов	Тахо	Комментарий
1	12/9/2023 1:12:39	Ссылка	4 шт	100 сек	__2022_04_28__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
2	12/9/2023 1:13:13	Ссылка	4 шт	100 сек	__2022_05_01__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
3	12/9/2023 1:10:46	Ссылка	4 шт	100 сек	__2021_11_17__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
4	12/9/2023 1:11:12	Ссылка	4 шт	100 сек	__2021_11_29__10_00_02.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
5	12/9/2023 1:11:47	Ссылка	4 шт	100 сек	__2021_12_04__10_00_02.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
6	12/9/2023 1:11:57	Ссылка	4 шт	100 сек	__2021_12_18__10_00_01КРОНШТЕЙНЫ.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	
7	12/9/2023 1:10:26	Ссылка	4 шт	100 сек	__2021_10_23__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Сверху_02 [1]	Датчик оборотов [0]	

3.7.1 Маршрутные точки

Данный функционал предназначен, как было описано ранее, для работы с маршрутными точками, их создания, настройки и привязки вибрационных сигналов (записей) к ним. В этом случае рабочая область разделена горизонтально на два отдельных блока для маршрутных точек и вибрационных сигналов (записей).



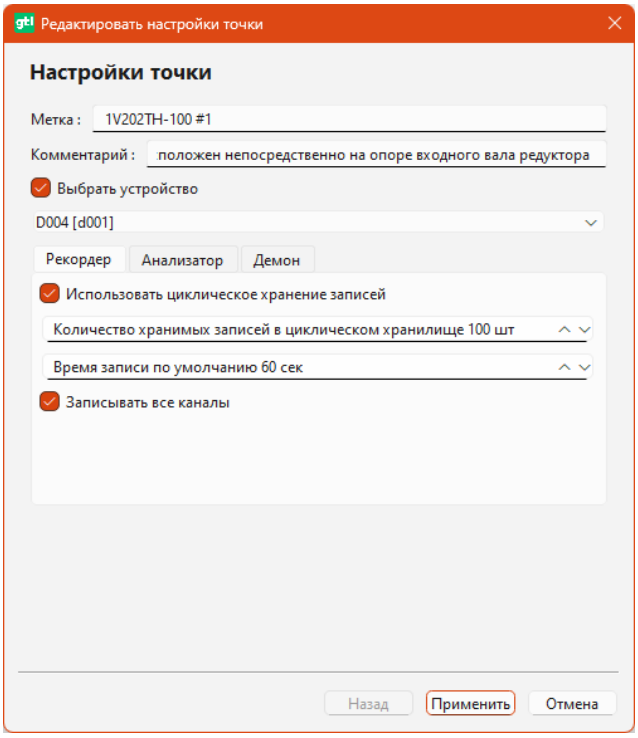
Работа с маршрутными точками (верхний блок рабочей области) реализована посредством контекстного меню, вызываемого при нажатии ПКМ.



В контекстном меню доступны следующие команды:

- «Записать эту точку» — запуск локального рекордера для записи сигнала с последующим сохранением и привязкой к выбранной маршрутной точке;
- «Клонировать точку» — создание копии маршрутной точки с сохранением всех настроек (без привязки записей);
- «Добавить точку» — добавление новой маршрутной точки в маршрут;
- «Редактировать точку» — изменение параметров выбранной маршрутной точки;
- «Удалить точку» — удаление выбранной точки из маршрута;
- «Экспортировать маршрутные точки» — сохранение параметров всех маршрутных точек во внешний файл в формате .csv.

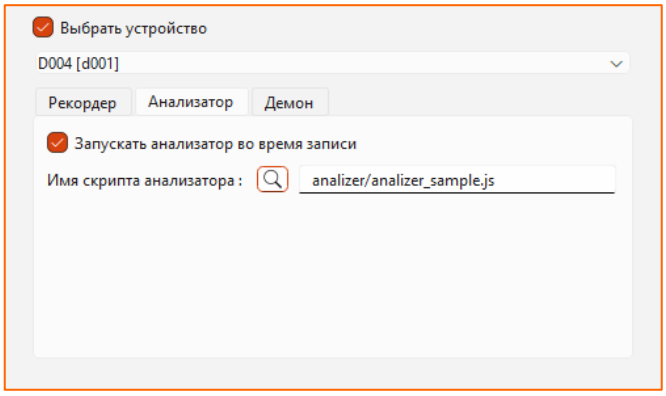
Более подробно рассмотрим процесс создания маршрутной точки, привязку к ней вибрационных сигналов (записей) и добавление маршрутных точек в созданные ранее «Объекты». Диалоговое окно добавления (редактирования) маршрутной точки:



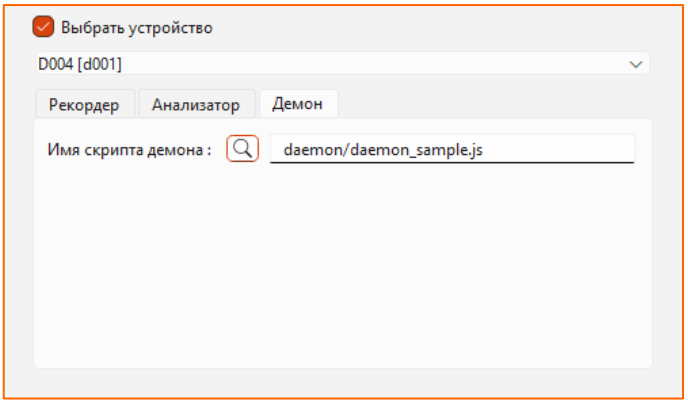
Для добавления маршрутной точки достаточно указать ее метку (имя) и добавить комментарий при необходимости. Дополнительно можно указать программе с какого внешнего устройства необходимо получать запись вибрационного сигнала (на примере указан модуль сбора данных d001 производства компании GTLAB) и настроить некоторые характерные параметры рекордера, анализатора и демона в зависимости от сценария использования системы диагностики и набора внешних устройств. Список доступных устройств формируется при помощи функционала менеджера устройств, работа с которым описана в разделе 3.3.2 данного руководства.

В настройках рекордера возможно настроить циклическое хранение записей сигналов, установив соответствующий чекбокс и указав их максимальное количество. Данный режим позволяет сохранять только указанное количество последних записей на диске (внешнем хранилище или планшете), что значительным образом может сэкономить дефицитное пространство (более старые записи будут удаляться). Дополнительно можно настроить длительность записи по умолчанию, указать конкретные каналы для записи вибрационного сигнала или выбрать все.

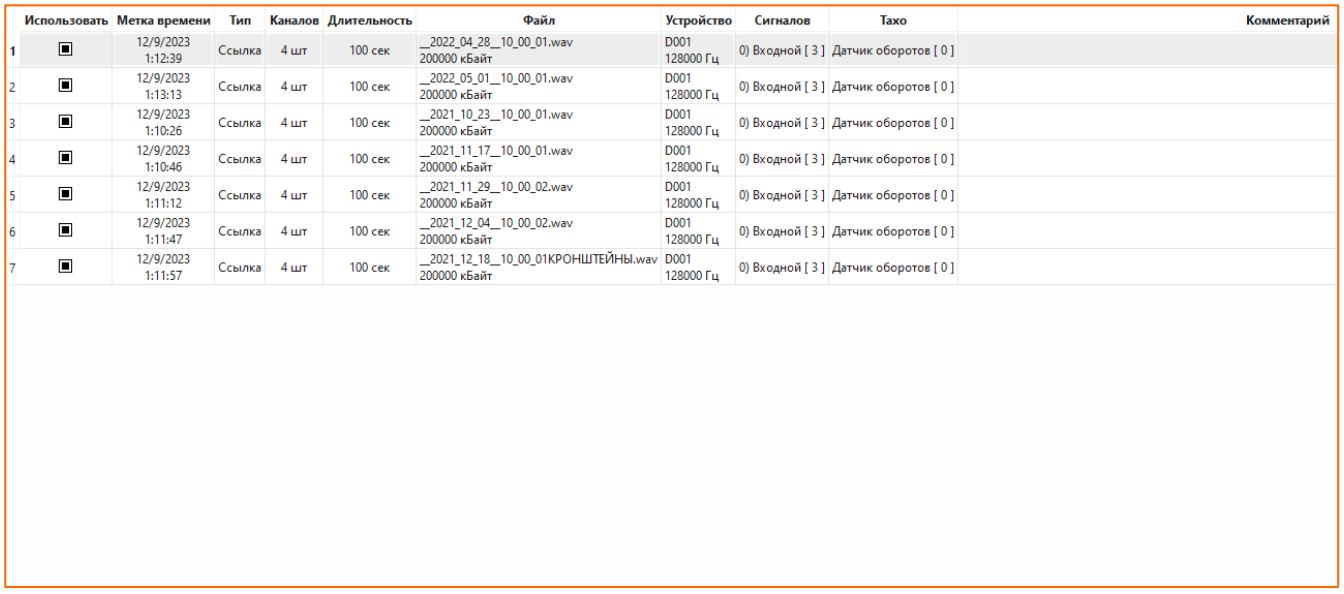
При работе в режиме виброанализатора D104 в настройках анализатора необходимо указать скрипт, по которому виброанализатор будет обрабатывать сигнал в реальном времени. Выбор предустановленных скриптов доступен из списка, вызываемого при нажатии на кнопку  .



При создании стационарной системы диагностики необходимо в настройках демона указать скрипт, по которому демон (отдельная консольная утилита) будет обрабатывать вибрационный сигнал и отдавать результаты выше по системе.

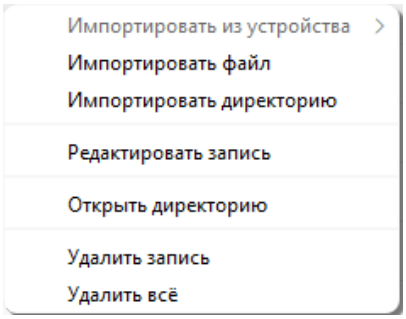


При выборе маршрутной точки в нижнем блоке рабочей области отображаются все привязанные к ней записи вибрационных сигналов.



	Использовать	Метка времени	Тип	Каналов	Длительность	Файл	Устройство	Сигналов	Тахо	Комментарий
1	▣	12/9/2023 1:12:39	Ссылка	4 шт	100 сек	__2022_04_28__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
2	▣	12/9/2023 1:13:13	Ссылка	4 шт	100 сек	__2022_05_01__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
3	▣	12/9/2023 1:10:26	Ссылка	4 шт	100 сек	__2021_10_23__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
4	▣	12/9/2023 1:10:46	Ссылка	4 шт	100 сек	__2021_11_17__10_00_01.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
5	▣	12/9/2023 1:11:12	Ссылка	4 шт	100 сек	__2021_11_29__10_00_02.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
6	▣	12/9/2023 1:11:47	Ссылка	4 шт	100 сек	__2021_12_04__10_00_02.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	
7	▣	12/9/2023 1:11:57	Ссылка	4 шт	100 сек	__2021_12_18__10_00_01КРОНШТЕЙНЫ.wav 200000 кБайт	D001 128000 Гц	0) Входной [3]	Датчик оборотов [0]	

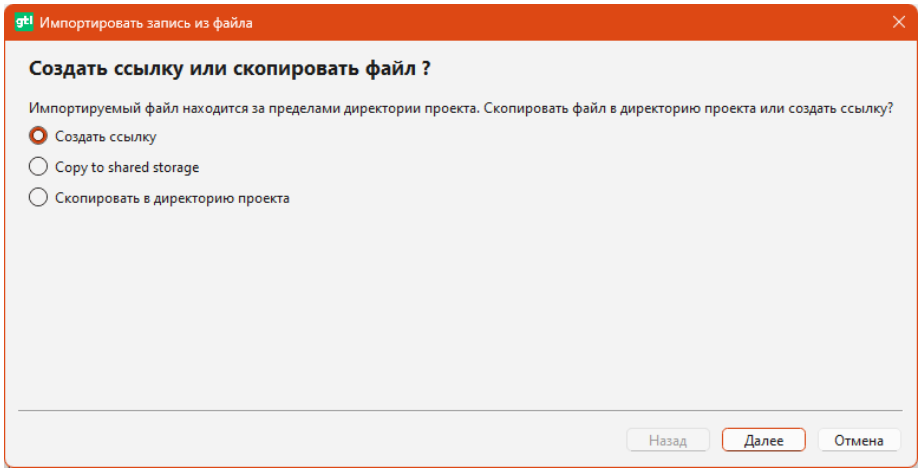
Работа с записями также реализована посредством контекстного меню, вызываемого при нажатии ПКМ.



В контекстном меню доступны следующие команды:

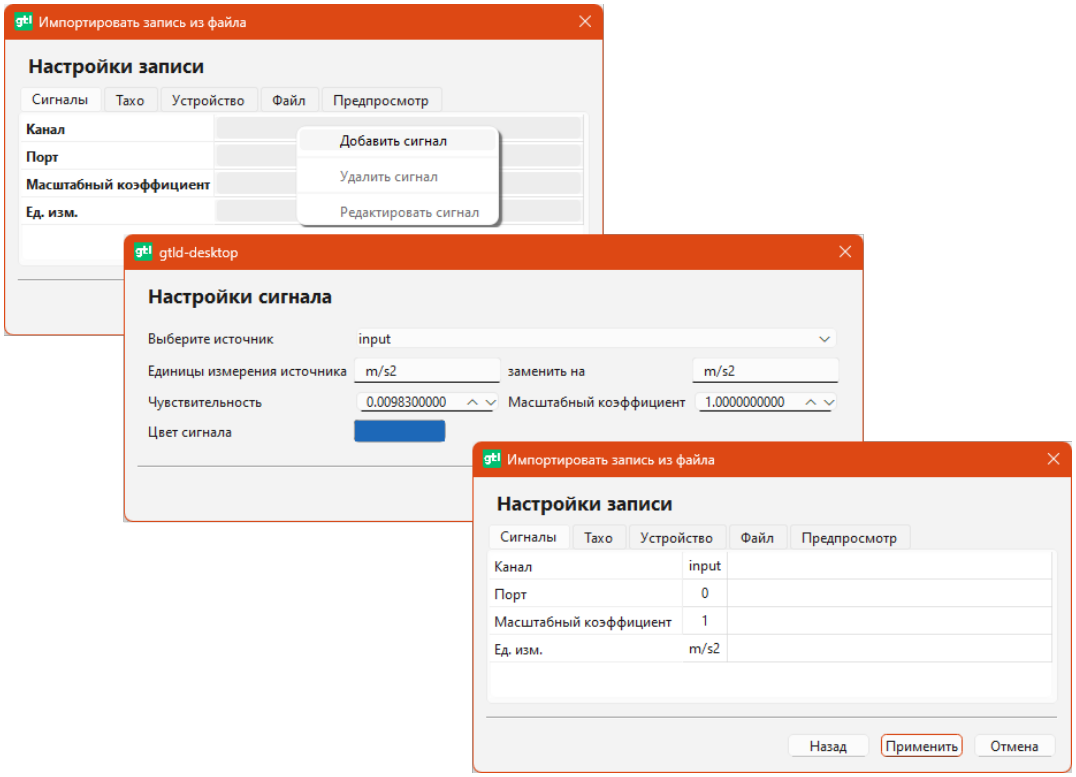
- «Импортировать из устройства» — получение записи вибрационного сигнала непосредственно из внешнего устройства;
- «Импортировать файл» — импорт файла сохраненной ранее записи с диска (внешнего хранилища);
- «Импортировать директорию» — импорт всех файлов из указанной директории;
- «Открыть директорию» — открытие директории в проводнике, в которой сохранена выбранная запись;
- «Удалить запись» — удаление выбранной записи из списка;
- «Удалить все» — очистка списка привязанных к маршрутной точке записей.

Более подробно рассмотрим процесс добавления (привязки) одной записи в маршрутную точку. Для этого необходимо вызвать контекстное меню, щелкнув ПКМ на свободном месте блока записей и выбрать команду «Импортировать файл». После выбора файла в проводнике запускается диалоговое окно, где вы можете указать необходимый способ добавления: ссылка, копирование в общую директорию или копирование файла непосредственно в проект.

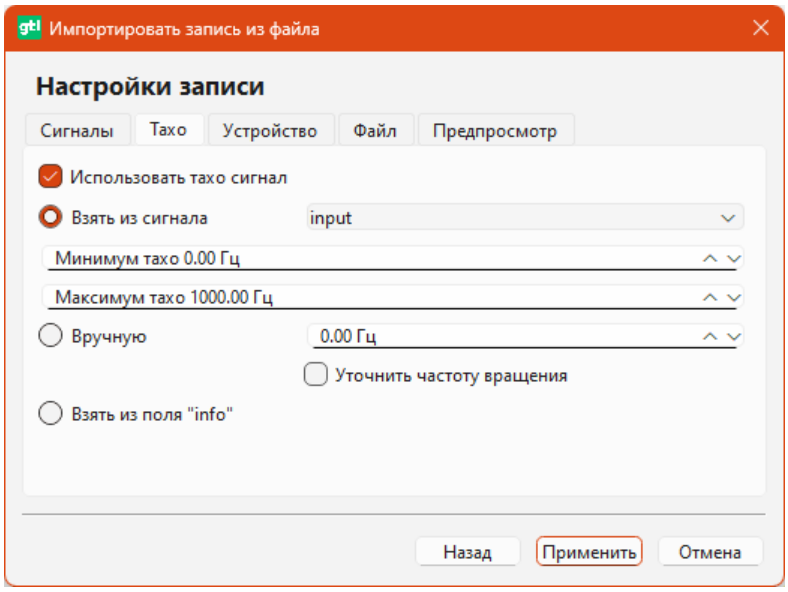


Следующим шагом (после нажатия на кнопку [Далее]) необходимо настроить параметры записи в несколько этапов, переключая вкладки диалогового окна настроек:

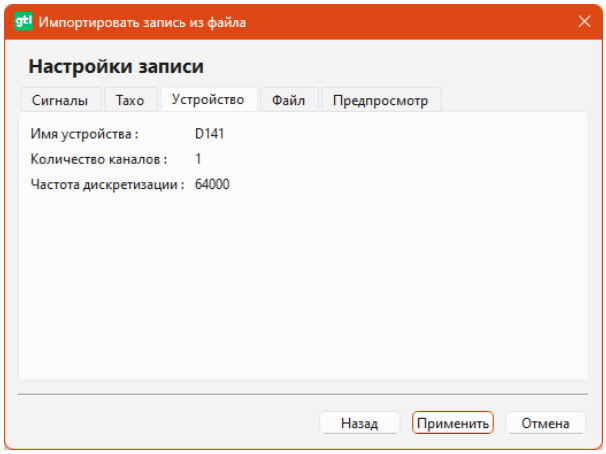
- «Сигналы» — добавление вибрационных сигналов в запись по каналам. Реализовано через контекстное меню при нажатии ПКМ. В окне настройки сигнала необходимо выбрать источник (канал при многоканальной записи), проверить (при необходимости откорректировать) единицы измерения, чувствительность, масштабный коэффициент и выбрать цвет отображения сигнала;



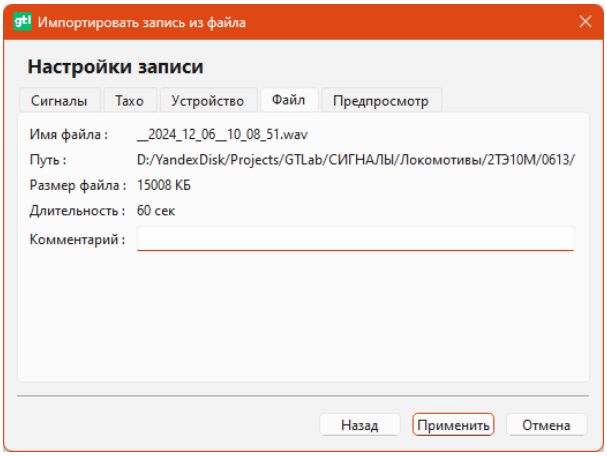
- «Тахо» — настройка источника данных по частоте вращения. В этом разделе вы можете указать каким образом получать информацию по частоте вращения: из указанного сигнала (канала), определить диапазон возможного разброса частоты вращения для её расчета из сигнала, установить значение вручную (реализована возможность уточнения заданной частоты по сигналу) или получать данные из поля «info» внешнего устройства (например виброметра);



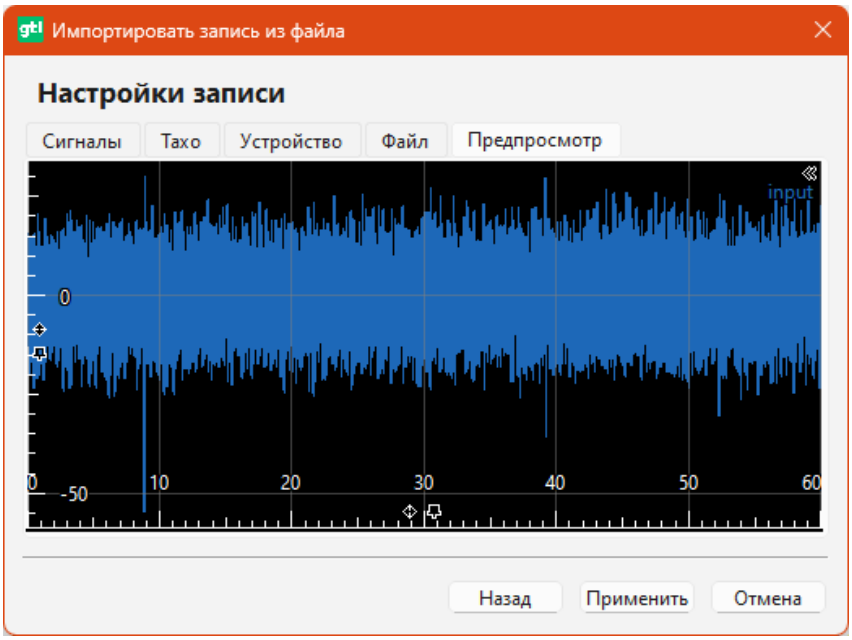
- «Устройство» — справочная информация об устройстве, на которое был записан вибрационный сигнал, а также о его частоте дискретизации;



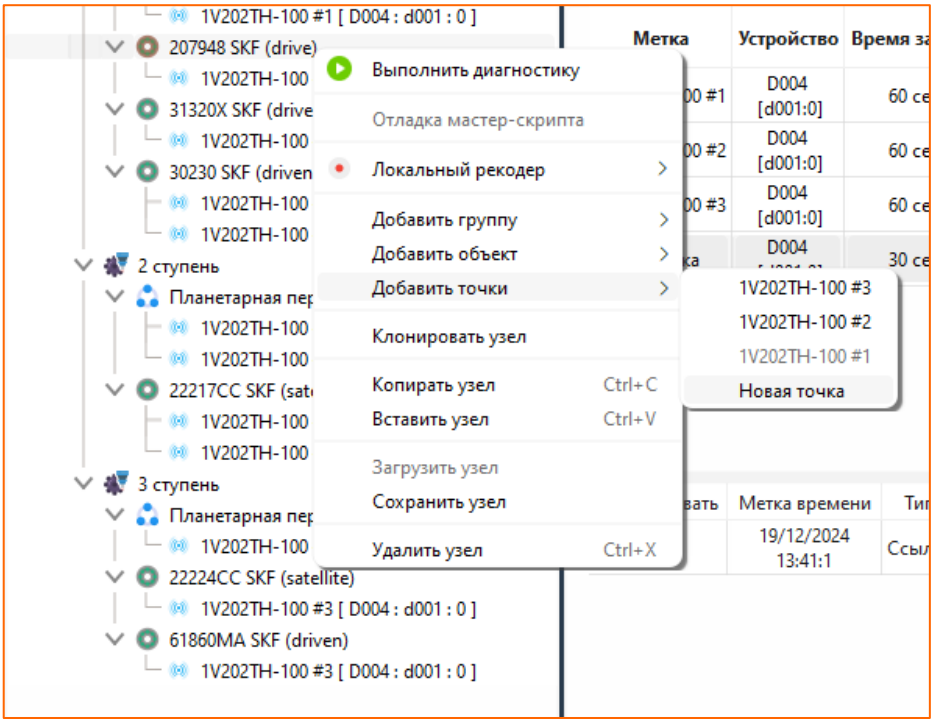
- «Файл» — справочная информация о файле записи;



- «Предпросмотр» — предварительный просмотр временной развертки вибрационного сигнала;

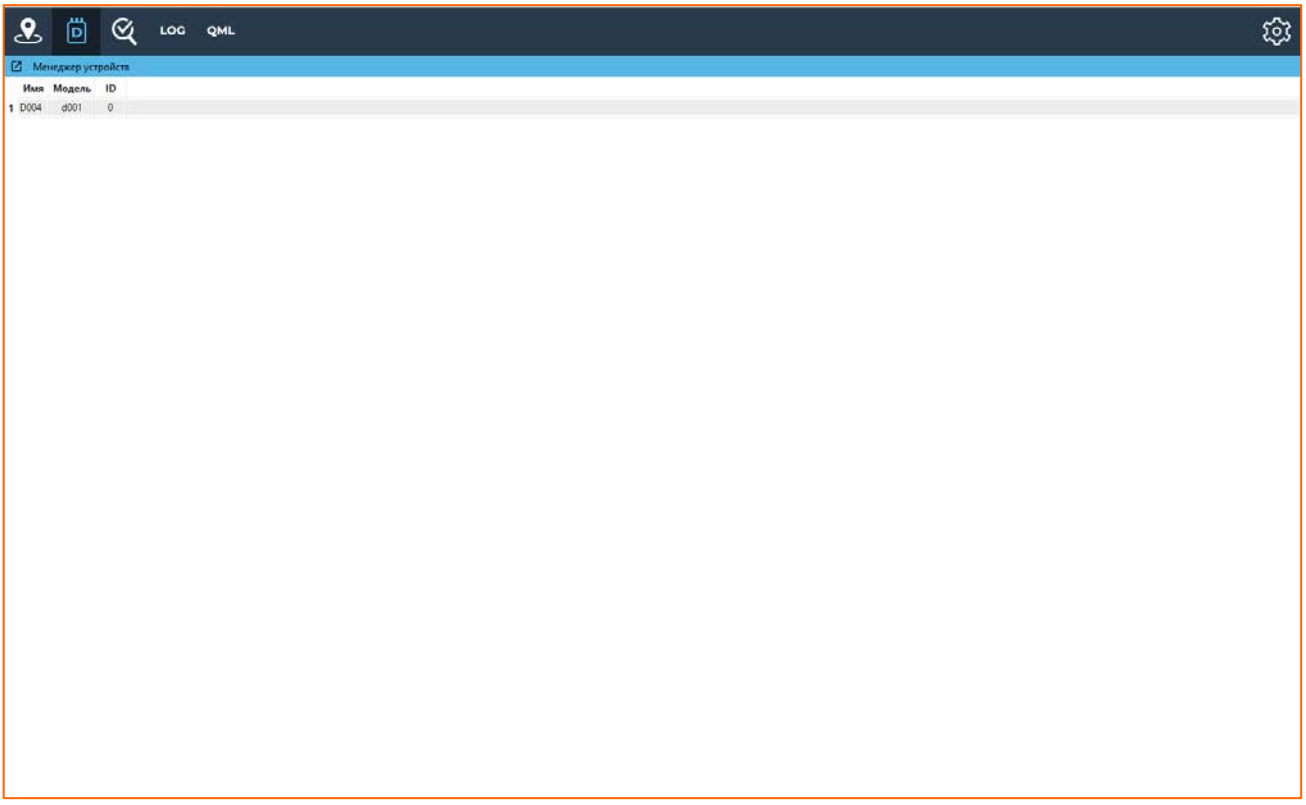


После создания маршрутной точки ее можно привязать к нужному «Объекту» через контекстное меню при нажатии ПКМ на выбранном объекте, либо перетаскиванием при зажатой ЛКМ.

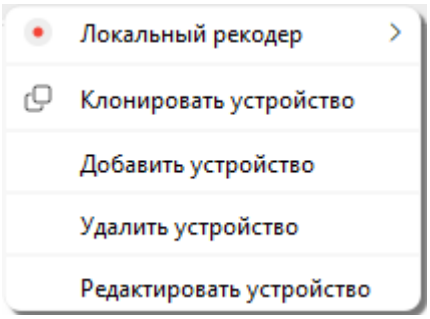


3.7.2 Менеджер устройств

Данный функционал предназначен для работы с внешними устройствами (модулями сбора данных), их подключением и настройкой.



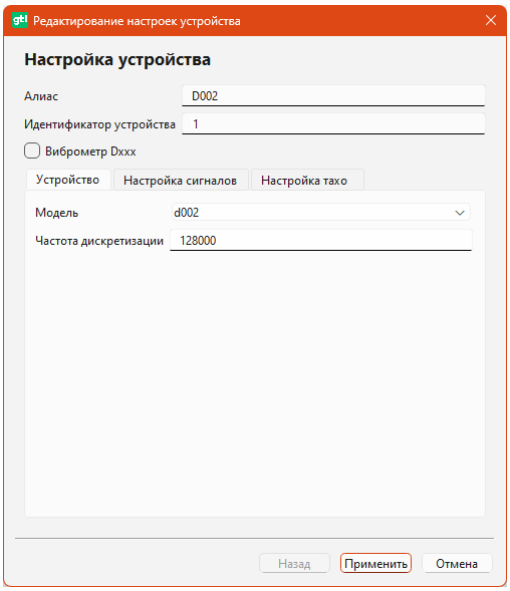
Работа с устройствами также реализована посредством контекстного меню, вызываемого при нажатии ПКМ.



В контекстном меню доступны следующие команды:

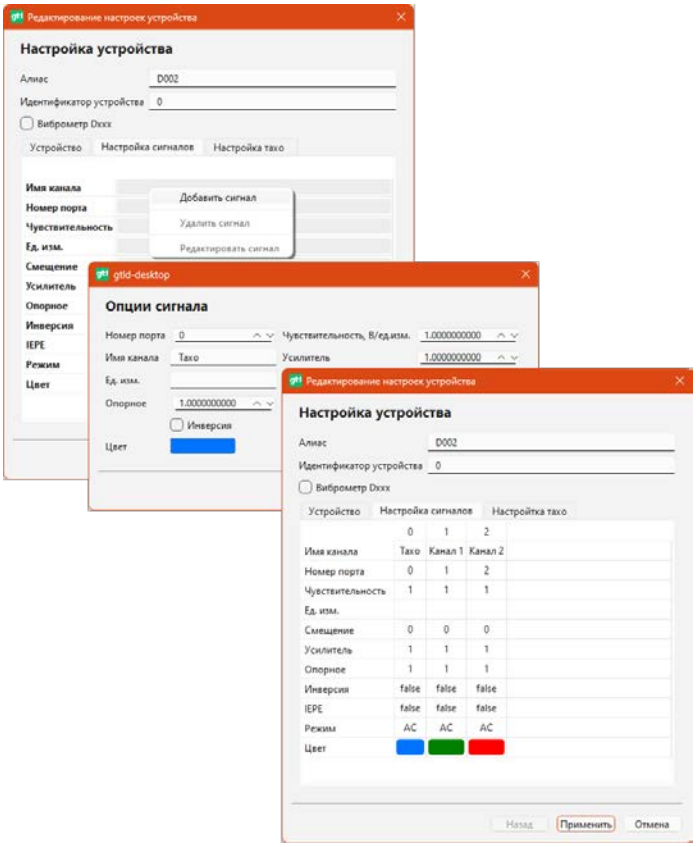
- «Локальный рекордер» — запуск локального рекордера для записи сигнала с выбором маршрутной точки для её привязки;
- «Клонировать устройство» — создание копии устройства со всеми настройками;
- «Добавить устройство» — добавление нового внешнего устройства;
- «Удалить устройство» — удаление выбранного устройства;
- «Редактировать устройство» — изменение настроек выбранного устройства;

Более подробно рассмотрим процесс добавления нового устройства. Для этого необходимо вызвать контекстное меню, щелкнув ПКМ на свободном месте блока записей и выбрать команду «Добавить устройство». Диалоговое окно добавления (редактирования) устройства:

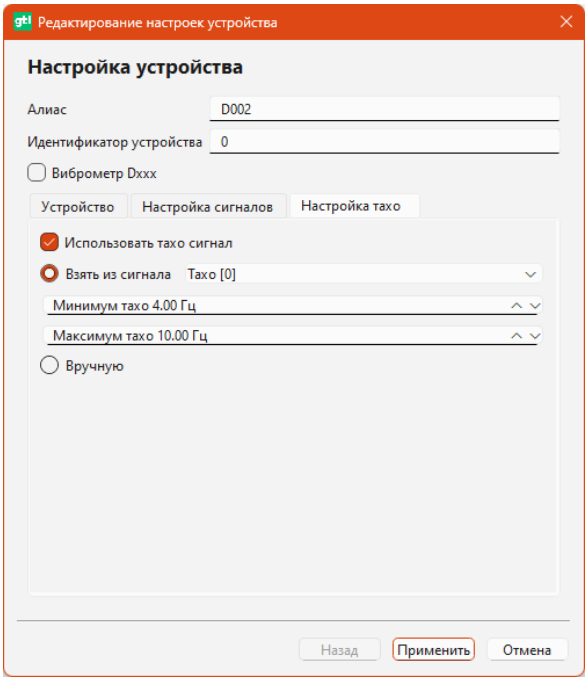


Для добавления устройства необходимо указать алиас (краткое наименование) и указать идентификатор. При работе с виброметрами необходимо отметить соответствующий чекбокс «Виброметр Dxxx». Если в проекте используются внешние модули сбора данных, то необходимо произвести настройку данных устройств в несколько этапов, переключая вкладки диалогового окна настроек:

- «Устройство» — выбор соответствующей модели и задание необходимой частоты дискретизации сигнала;
- «Настройка сигналов» — добавление вибрационных сигналов в запись по каналам. Реализовано через контекстное меню при нажатии ПКМ. В окне настройки сигнала необходимо указать номер порта устройства и настроить характерные параметры;



- «Настройка тахо» — настройка источника данных по частоте вращения. В этом разделе вы можете указать каким образом получать информацию по частоте вращения: из ранее добавленного канала, определить диапазон возможного разброса частоты вращения для её расчета из сигнала или установить значение вручную.



3.7.3 Анализатор результатов

Данный функционал предназначен для работы с результатами автоматической диагностики, их анализом и настройкой. В данном случае рабочая область разделена горизонтально на три отдельных блока для выбора результатов из базы данных, отображения списка полученных

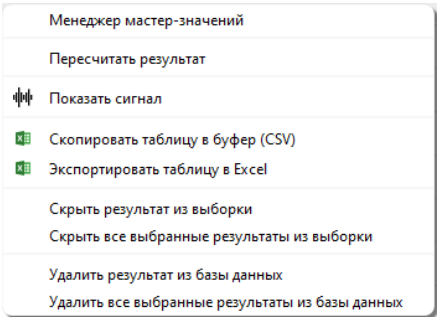
результатов из базы данных в виде таблицы и непосредственного отображения подробной информации по результатам автоматической диагностики.

Блок выбора результатов из базы данных реализован по принципу фильтра. Отбор результатов можно осуществлять по тегу и дате проведения диагностики. Тег представляет собой идентификатор, по которому результаты диагностики группируются в базе данных. Тег указывается при каждом запуске автоматической диагностики и позволяет разделять результаты по любому принципу (изменение параметров при отладке методики диагностики, разграничение результатов по режимам работы оборудования и т.д.). Для выбора результатов из базы данных необходимо настроить фильтры тега и даты и нажать кнопку [Выбрать]. Список полученных результатов отображается ниже в виде таблицы. Кнопка [Сброс] предназначена для очистки данной таблицы.

	Дата записи	Дата вычисления	Тэг	Комментарий	Имя объекта	Метка точки	Имя файла	Длина результата	rms_A2_10000
1	16:52:19 02.02.2025	9:46:50 06.02.2025	mask		maskMethod	1V202TH-100 #4	ТЭМ18 1966 КМБ2 Букса_Левая - Дефект сепаратора 4.19.wav	2635	0.211214
2	16:51:04 02.02.2025	9:46:45 06.02.2025	mask		maskMethod	1V202TH-100 #4	ТЭМ18 1966 КМБ1 Букса_Правая - Дефект НК 4.88.wav	2649	0.240957
3	16:49:05 02.02.2025	9:46:41 06.02.2025	mask		maskMethod	1V202TH-100 #4	ТЭМ18 1966 КМБ1 Букса_Левая - Дефект НК 4.79.wav	2603	0.229231
4	16:33:39 02.02.2025	9:46:22 06.02.2025	mask		maskMethod	1V202TH-100 #4	2ТЭ10 0613 А4 Букса_Правая - Дефект НК 5.28.wav	3452	0.50129
5	16:33:28 02.02.2025	9:46:26 06.02.2025	mask		maskMethod	1V202TH-100 #4	2ТЭ10 0613 А6 Букса_Правая - Дефект НК 4.62.wav	2648	0.390691
6	16:33:28 02.02.2025	9:46:31 06.02.2025	mask		maskMethod	1V202TH-100 #4	2ТЭ10 0613 Б3 Букса_Левая - Дефект НК 4.9.wav	3049	0.318842
7	16:33:28 02.02.2025	9:46:36 06.02.2025	mask		maskMethod	1V202TH-100 #4	2ТЭ10 0613 Б6 Букса_Левая - Дефект НК 5.45.wav	2672	0.649069
8	16:20:46 02.02.2025	9:46:17 06.02.2025	mask		maskMethod	1V202TH-100 #4	2М62 030 А5 Букса_Левая - Дефект НК 4.8.wav	2867	0.280009

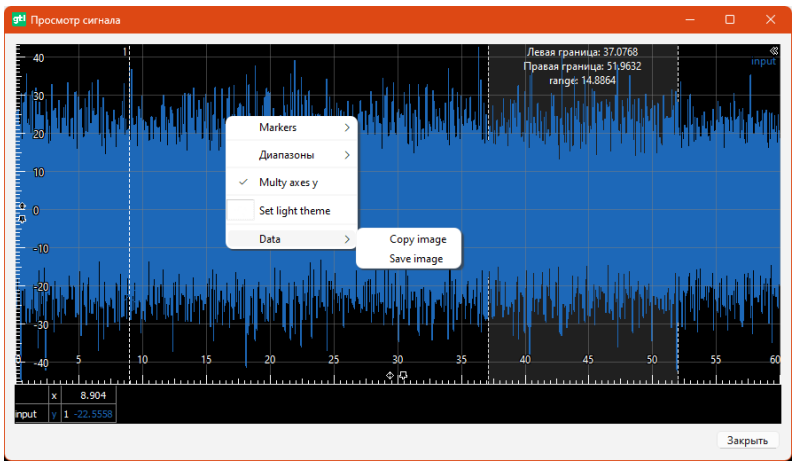
Помимо отображения списка результатов в таблице предусмотрен функционал для отслеживания любых параметров (мастер-значений), выводимых в блок JSON обозревателя (подробнее описание функционала ниже) с отображением границ состояния.

Работа с выбранными результатами реализована посредством контекстного меню, вызываемого при нажатии ПКМ.

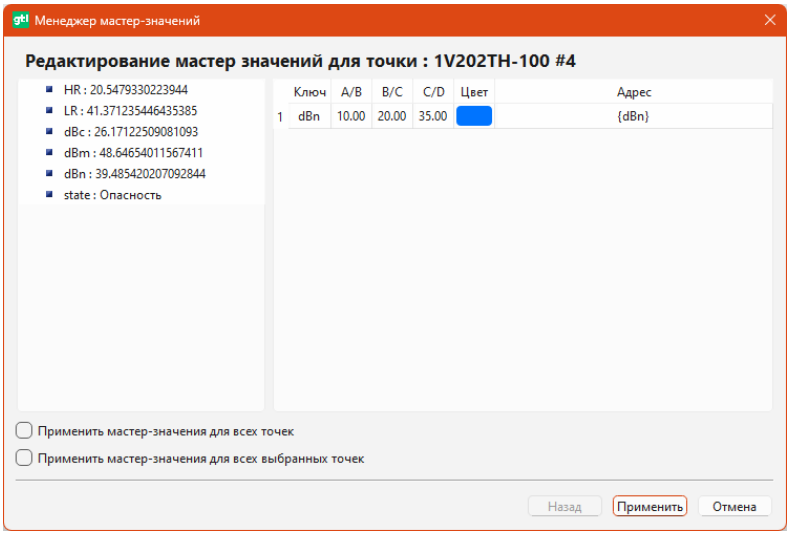


В контекстном меню доступны следующие команды:

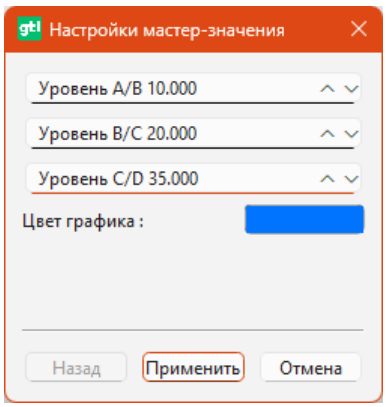
- «Менеджер мастер-значений» — запуск инструмента для выбора отслеживаемых параметров, которые доступны в JSON обозревателе, для реализации функционала мониторинга. Детальная работа с менеджером описана ниже;
- «Пересчитать результат» — повторный запуск автоматической диагностики и пересчетом результатов;
- «Показать сигнал» — запуск инструмента просмотра вибрационного сигнала с возможностью, установки маркеров, добавления диапазона сигнала и сохранения его в формате изображения посредством команд контекстного меню, вызываемого при нажатии ПКМ;
- «Скопировать таблицу в буфер (CSV)» — сохранение таблицы результатов в буфер обмена в формате CSV;
- «Экспортировать таблицу в Excel» — сохранение таблицы результатов в файл формата .xls;
- «Скрыть результат из выборки» — удаление выбранного результата из таблицы результатов;
- «Скрыть все выбранные результаты из выборки»: удаление всех результатов из таблицы результатов (команда аналогична нажатию кнопки [Сброс] в блоке выбора результатов из базы данных);
- «Удалить результат из базы данных» — безвозвратное удаление выбранного результата из базы данных;
- «Удалить все выбранные результаты из базы данных» — безвозвратное удаление всех результатов в таблице из базы данных;



Рассмотрим подробнее функционал менеджера мастер-значений. Окно менеджера состоит из двух блоков. В левом блоке отображаются все доступные для отслеживания (мониторинга) параметры, а в правом блоке список выбранных параметром с установленными границами зон состояния.



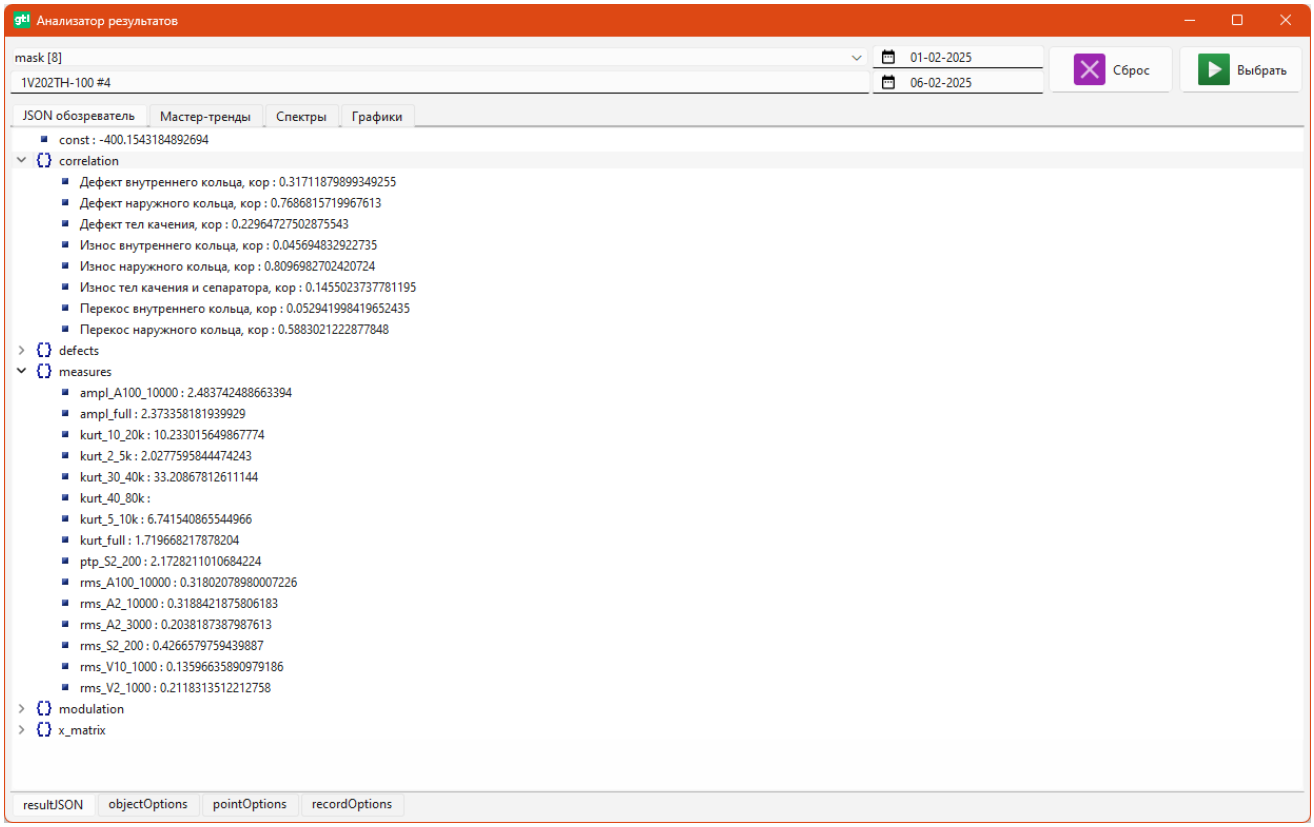
Выбор необходимого параметра происходит по команде «Выбрать мастер-значение» контекстного меню, вызываемого при нажатии ПКМ. Редактирование границ зон состояния (A/B, B/C и C/D) и выбор цвета отображения линии тренда реализовано в отдельном окне, которое вызывается двойным нажатием ЛКМ на выбранном параметре в правом блоке менеджера.



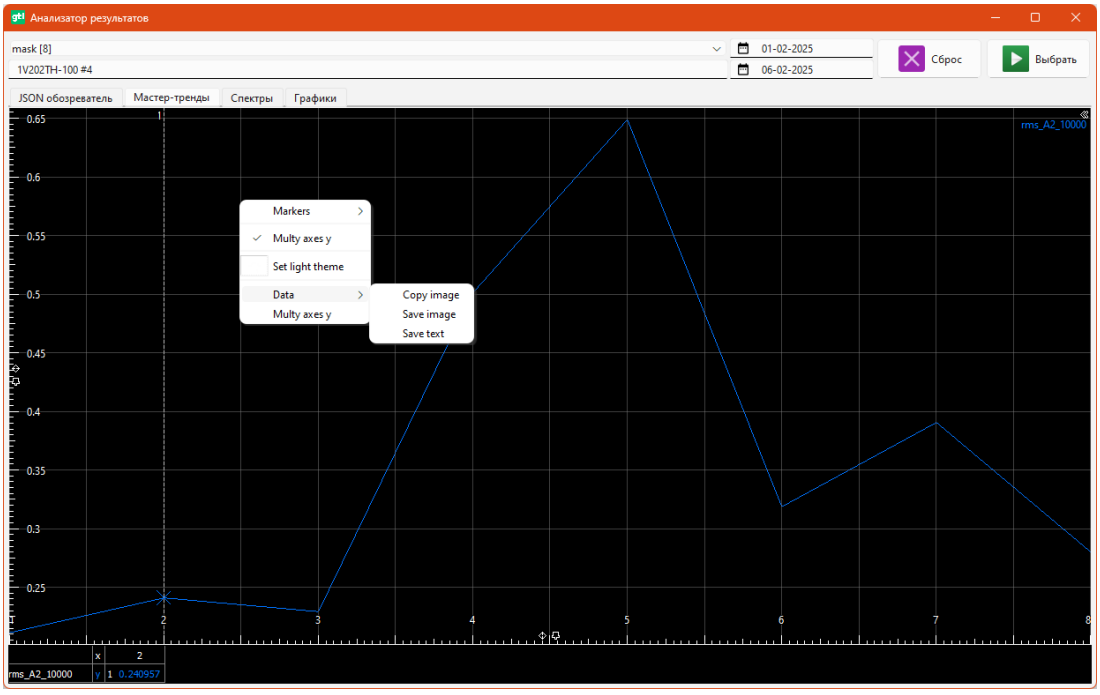
Настройки отслеживания параметров (мастер-значений) возможно сохранить для выбранных маршрутных точек объекта или для всех точек объекта, установив соответствующий маркер в окне менеджера мастер-значений.

Для того, чтобы посмотреть подробности по результату автоматической диагностики необходимо выделить нужный результат в таблице нажатием ЛКМ. Блок отображения подробной информации по результатам автоматической диагностики состоит из нескольких вкладок:

- «JSON обозреватель» — отображение результатов автоматической диагностики по выбранной методике в формате JSON, а также другой служебной информации в соответствующих вкладках, расположенных в его нижней части;

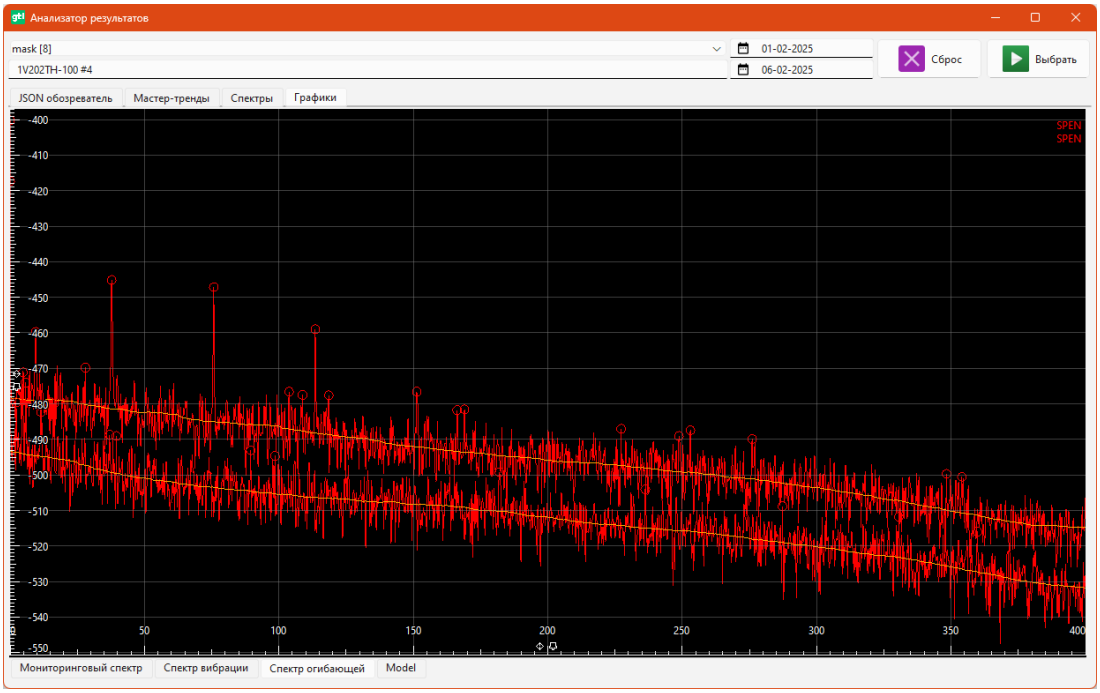


- «Мастер-тренды» — построение линий трендов отслеживаемых параметров (мастер-значений) на координатной плоскости с возможностью, установки маркеров и сохранения результата в формате изображения или в файл .csv посредством команд контекстного меню, вызываемого при нажатии ПКМ;

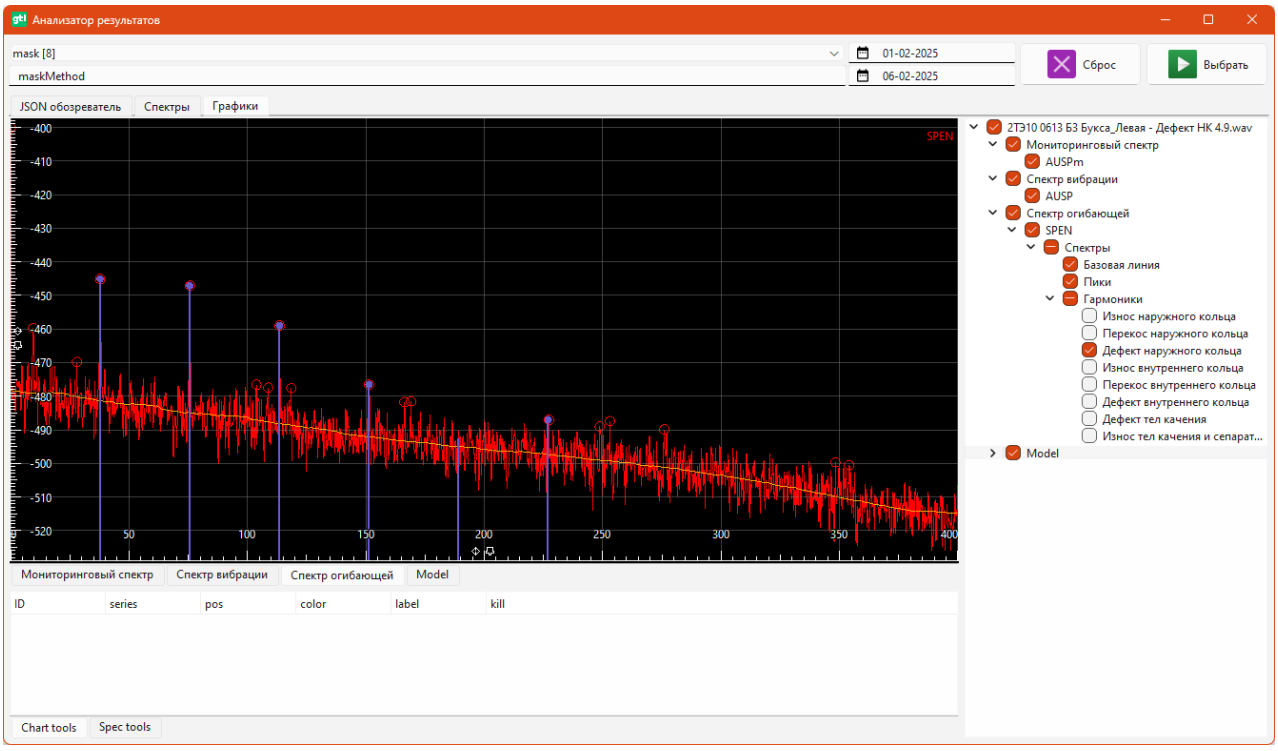


- «Графики» — графическое отображение результатов автоматической диагностики по выбранной методике. Для того, чтобы отобразить графические результаты, необходимо перетянуть выбранный результат из таблицы на свободное поле вкладки «Графики» путем нажатия и удержания ЛКМ. Реализована возможность сравнения между собой нескольких

результатов путем их поочередного перетягивания на поле данной вкладки. В нижней части вкладки расположены дополнительные кнопки для перехода на координатные плоскости всех графиков, построенных по выбранной методике;

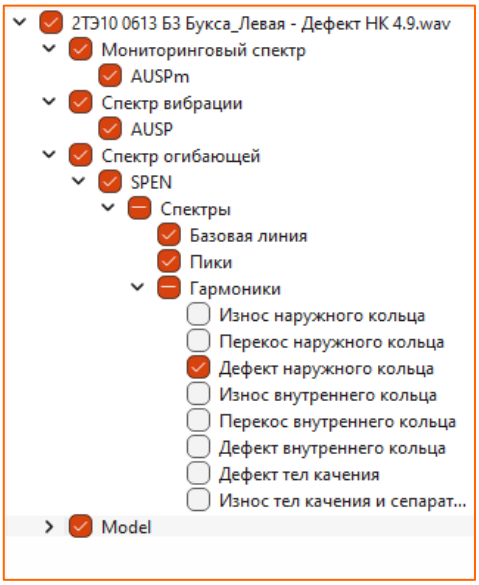


Графики являются важным элементом, отражающим результаты автоматической диагностики, для аналитической работы с которыми предусмотрены инструменты, выведенные в отдельные блоки: правый, отвечающий за отображение структуры выбранного результата диагностики и нижний, предназначенный для работы с маркерами на графиках.



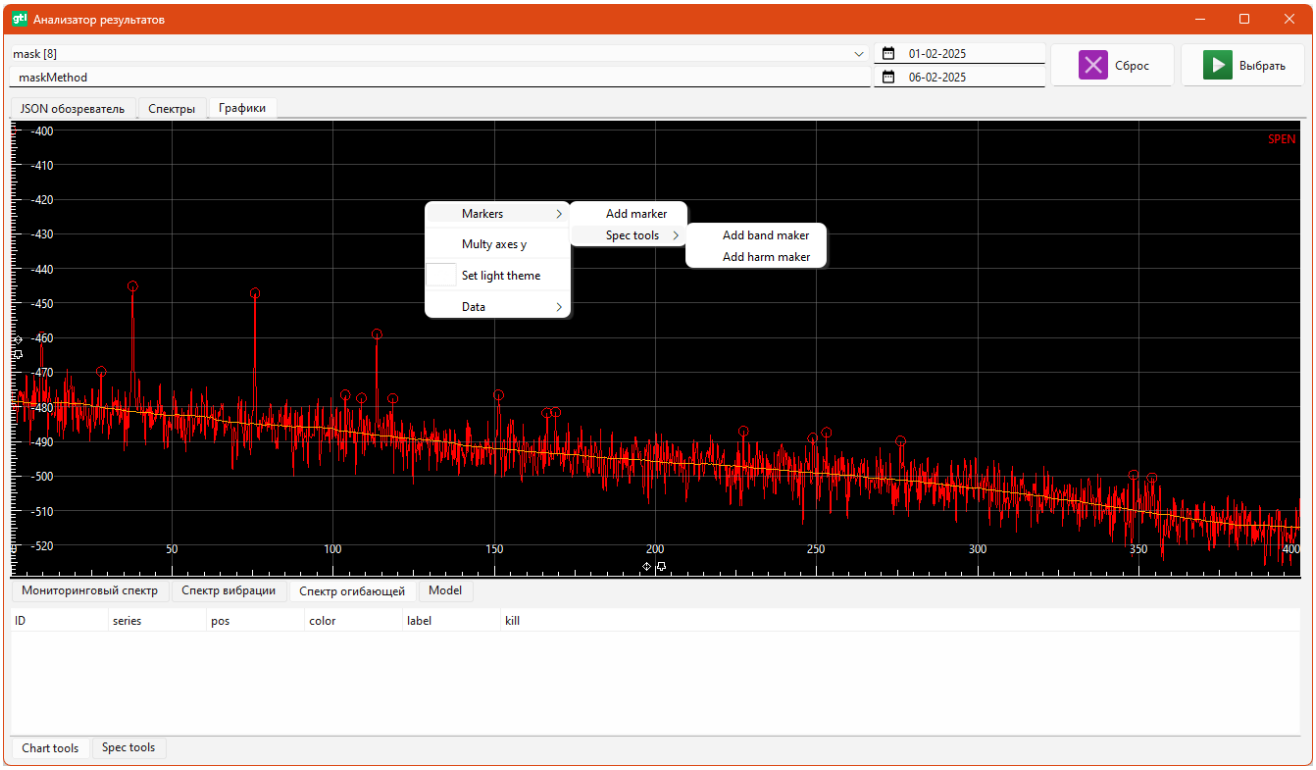
Правый блок предназначен для отображения структуры выбранного из таблицы результата в виде дерева, корневой узел которого соответствует названию выбранного результата

(наименованию записи вибрационного сигнала) – «2ТЭ10 0613 Б3 Букса_Левая – Дефект НК», а его нижние ветви соответствуют каждому из построенных по выбранной методике графику и добавленным на них компонентам (маркеры обнаруженных гармоник, скользящая средняя линия спектра, спектральные линии и т.д.). Управление видимостью всех элементов осуществляется маркерами (чекбоксами).



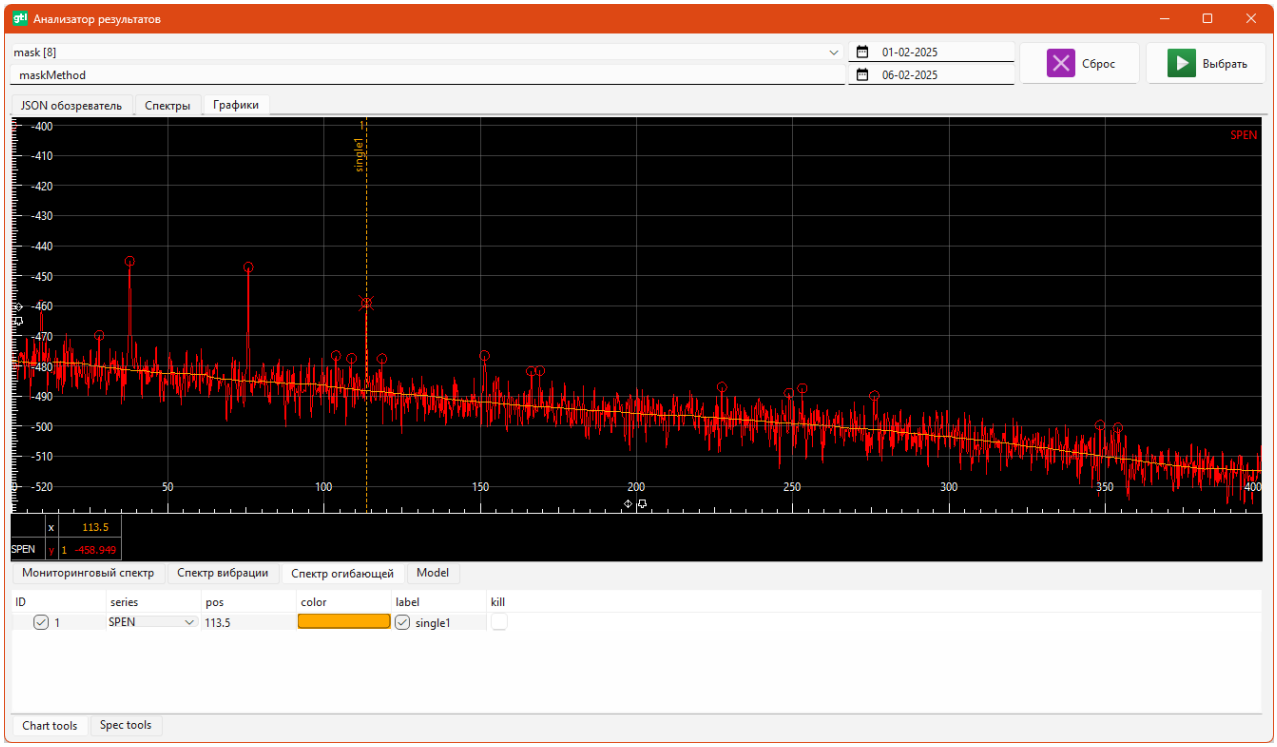
Структура дерева результата зависит от выбранной методики диагностики и формируется исключительно с помощью интерфейса JS API в диагностическом скрипте.

Нижний блок предназначен для управления маркерами, добавление которых на график реализовано через команды контекстного меню, вызываемого при нажатии ПКМ.



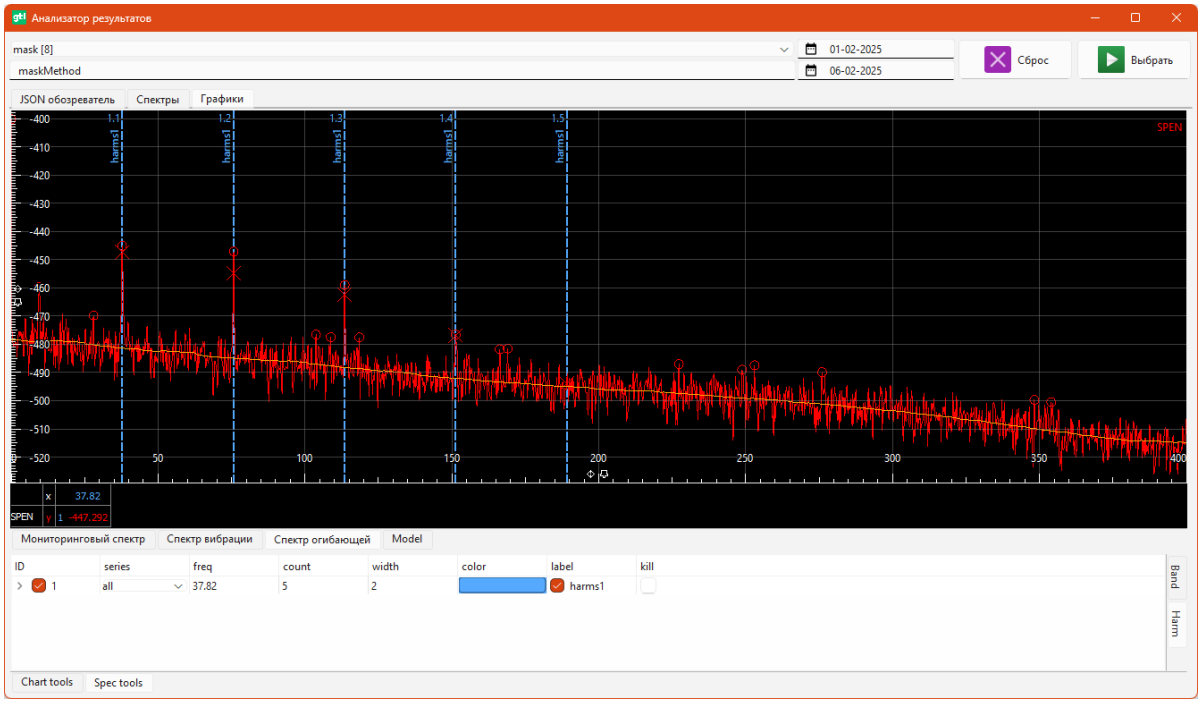
Реализовано три основных типа маркеров:

- «Одиночный» — отдельный независимый маркер, который предназначен для анализа параметров отдельных составляющих графика (спектра). Добавляется командой контекстного меню Markers -> Add marker, параметры которого можно изменять таблице, расположенной во вкладке Chart tools. В таблице настроек маркера имеются следующие параметры:
  - «ID» – идентификатор. Присваивается в порядке его добавления на график. Реализована возможность управления видимостью маркера при помощи соответствующего чекбокса;
  - «series» – привязка маркера к конкретному графику (или ко всем) в случае отображения нескольких графиков на одной координатной плоскости (выбирается из списка);
  - «pos» – позиция маркера по оси X. Изменение позиции также реализовано при помощи ЛКМ непосредственно на графике;
  - «color» – выбор цвета маркера;
  - «label» – подпись маркера. Реализована возможность управления видимостью подписи при помощи соответствующего чекбокса;
  - «kill» – кнопка удаления маркера.

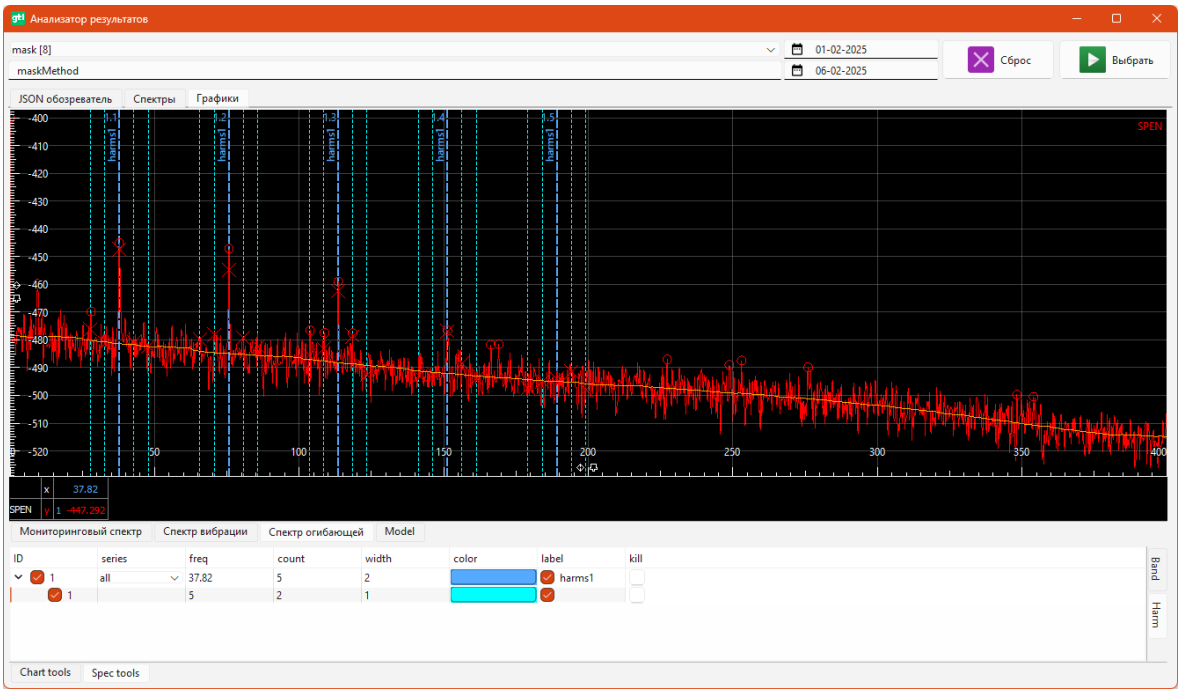


- «Гармонический» — маркер, способный отображать ряд суперпозиций, кратных его значению. Предназначен для анализа параметров периодических составляющих на графике (гармонических рядов на спектрах вибрации). Добавляется командой контекстного меню Markers -> Spec tools -> Add harm marker, параметры которого можно изменять таблице, расположенной во вкладке Spec tools, раздел harm. В таблице настроек маркера имеются следующие параметры:
  - «ID» – идентификатор. Присваивается в порядке его добавления на график. Реализована возможность управления видимостью маркера при помощи соответствующего чекбокса;
  - «series» – привязка маркера к конкретному графику (или ко всем) в случае отображения нескольких графиков на одной координатной плоскости (выбирается из списка);
  - «freq» – позиция маркера по оси X (в спектре вибрации его частота). Изменение позиции также реализовано при помощи ЛКМ непосредственно на графике, при этом происходит автоматический пересчет его суперпозиций (гармоник на спектре);
  - «count» – количество составляющих ряда (гармоник спектра). Изменение позиции любой составляющей ряда также реализовано при помощи ЛКМ непосредственно на графике, при этом происходит автоматический пересчет всех составляющих ряда;

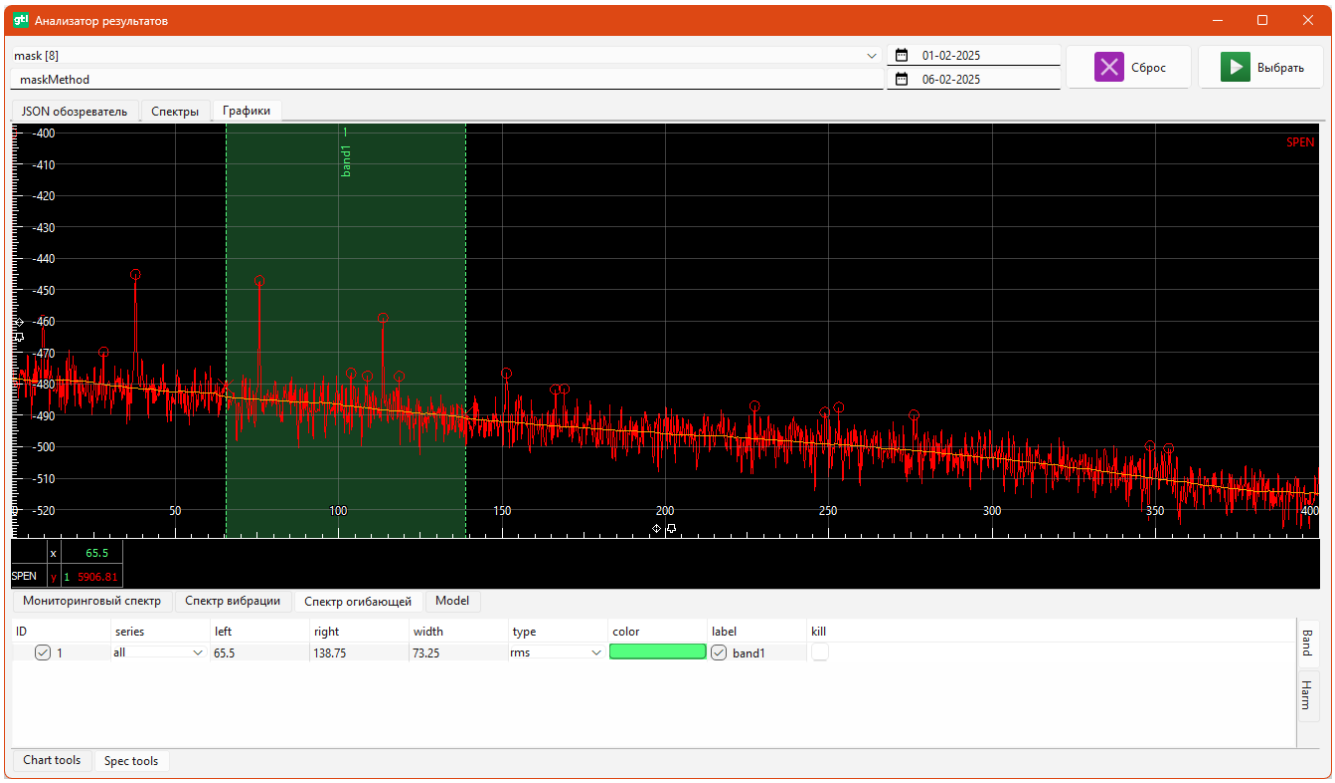
- «width» – толщина линии маркера (ряда);
- «color» – выбор цвета маркера (ряда);
- «label» – подпись маркера. Реализована возможность управления видимостью подписи при помощи соответствующего чекбокса;
- «kill» – кнопка удаления маркера.



Специально для анализа спектров вибрации реализовано добавление «модулирующего маркера» или боковых составляющих к гармоническому маркеру (в случае амплитудной модуляции сигнала). Данный дополнительный маркер добавляется командой Add modulating harmonics контекстного меню, вызываемого при нажатии ПКМ на необходимой строке таблицы списка гармонических маркеров. Настройки и работа с «модулирующим маркером» происходит аналогично гармоническому маркеру.

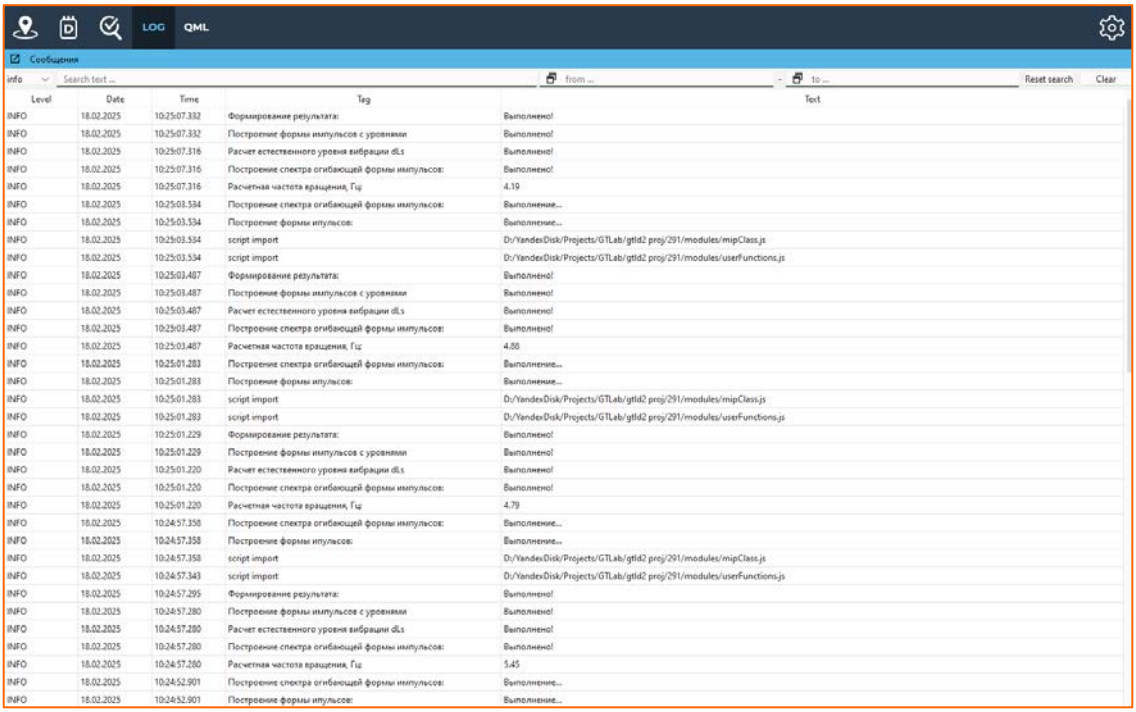


- «Полосовой» — маркер, способный выделять определенную область (диапазон) на графике. Предназначен для анализа параметров внутри выделенного диапазона. Добавляется командой контекстного меню Markers -> Spec tools -> Add band marker. Сначала устанавливается его левая граница нажатием ЛКМ, а затем правая повторным нажатием ЛКМ. Параметры добавленного маркера можно изменять таблице, расположенной во вкладке Spec tools, раздел band. В таблице настроек маркера имеются следующие параметры:
  - «ID» – идентификатор. Присваивается в порядке его добавления на график. Реализована возможность управления видимостью маркера при помощи соответствующего чекбокса;
  - «series» – привязка маркера с конкретному графику (или ко всем) в случае отображения нескольких графиков на одной координатной плоскости (выбирается из списка);
  - «left» – левая граница маркера (в спектре вибрации его частота). Изменение позиции также реализовано при помощи ЛКМ непосредственно на графике;
  - «right» – правая граница маркера (в спектре вибрации его частота). Изменение позиции также реализовано при помощи ЛКМ непосредственно на графике;
  - «width» – ширина полосы маркера;
  - «type» – расчетный параметр в указанной полосе: rms – среднеквадратическое значение, peak – максимальное значение, power density – удельная мощность;
  - «color» – выбор цвета маркера (ряда);
  - «label» – подпись маркера. Реализована возможность управления видимостью подписи при помощи соответствующего чекбокса;
  - «kill» – кнопка удаления маркера.

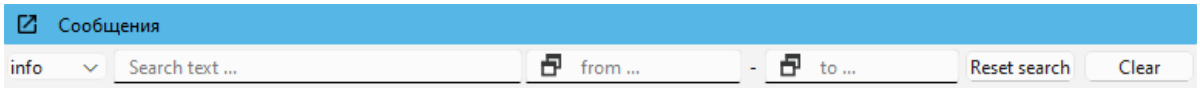


3.7.4 Сообщения

Данный функционал предназначен для вывода служебной информации при запуске диагностики. В окно сообщений выводится информация по используемым модулям и библиотекам, процессу выполнения диагностического скрипта, результаты промежуточных расчетов, ошибки и любая другая информация, необходимая для отладки диагностической методики.

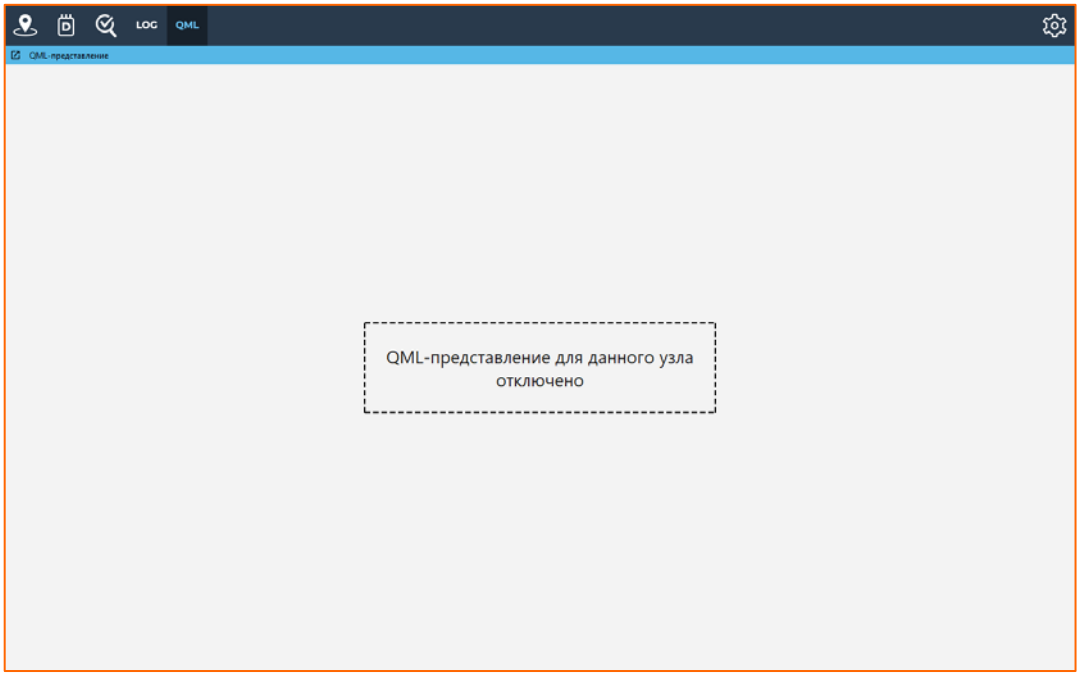


Дополнительно реализована возможность фильтрации (поиска) сообщений по категории, тексту и дате при помощи отдельных инструментов, а также предусмотрена очистка таблицы сообщений при нажатии на кнопку [Clear].



3.7.5 QML-представление

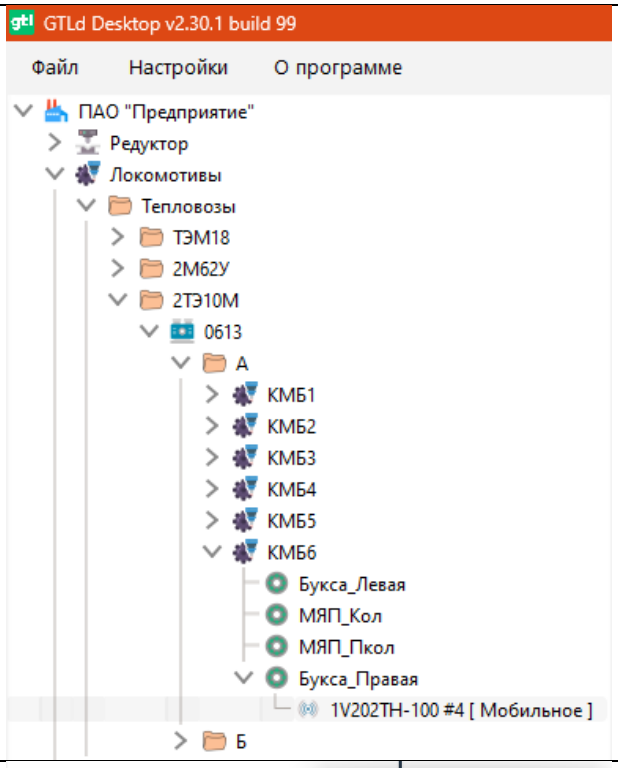
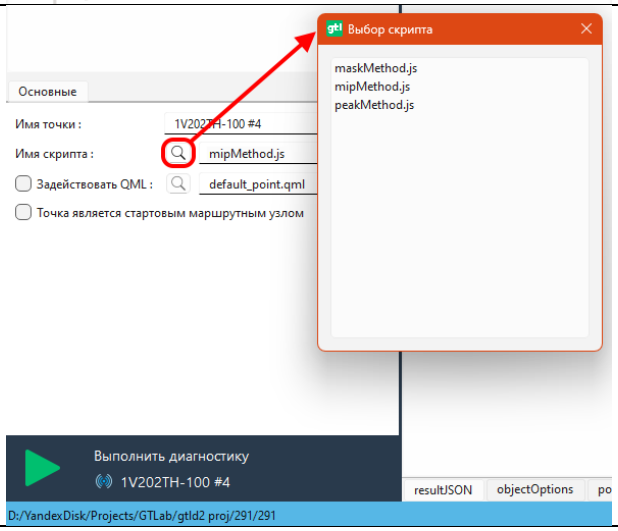
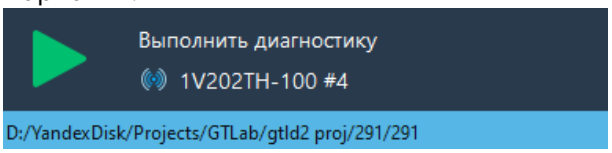
Данный функционал предназначен для вывода визуализации результатов диагностики или каких-либо вычислений в виде таблиц, блоков или графиков. Чтобы включить визуализацию, необходимо в основных свойствах объекта установить маркер напротив пункта «Задействовать QML» и выбрать из списка необходимый вариант визуализации.

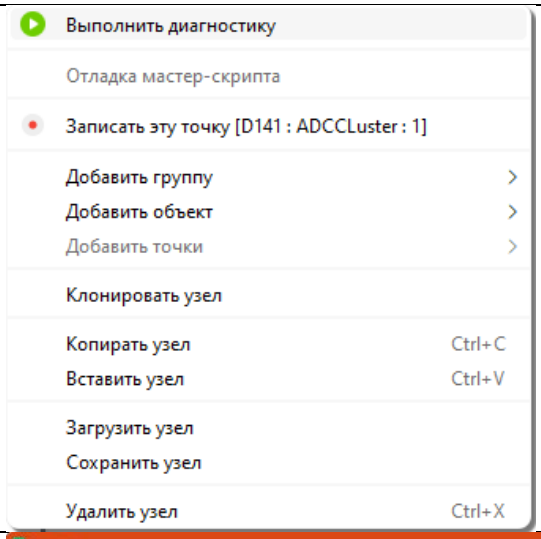
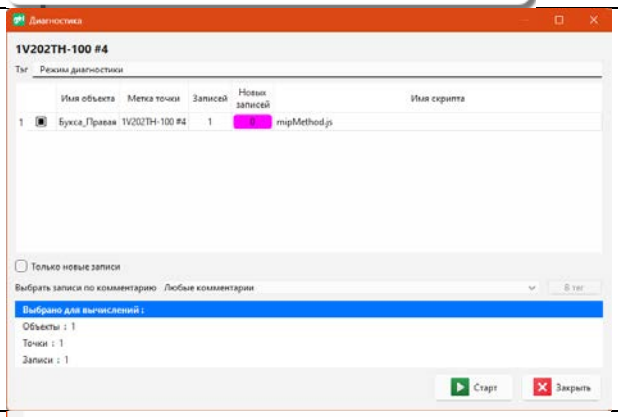
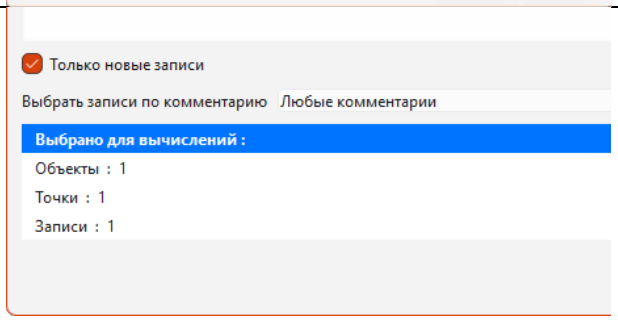
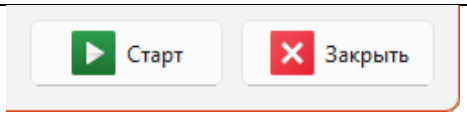


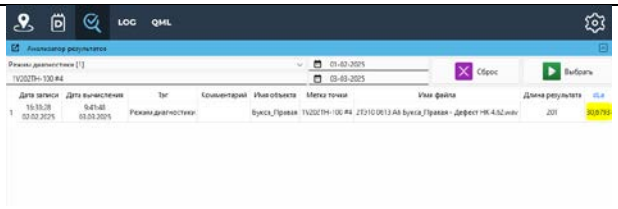
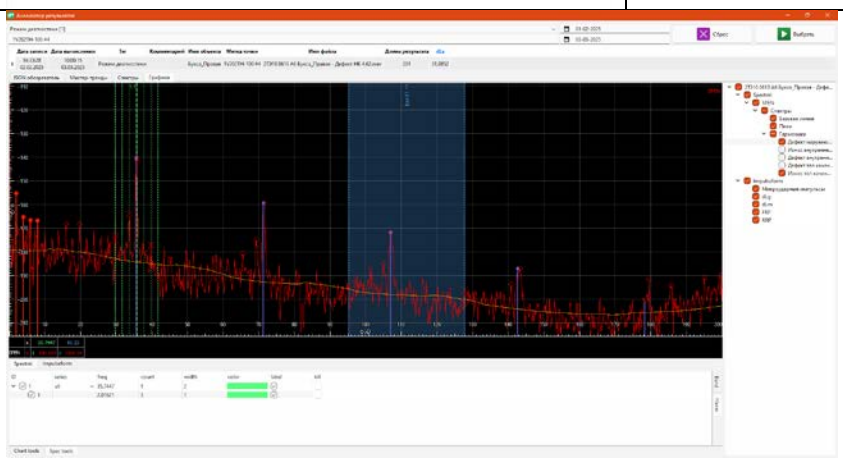
4. Запуск процесса диагностики

Запуск процесса автоматической диагностики пошагово описан в таблице 4.1. Рассмотрен пример запуска диагностики для маршрутной точки.

Таблица 4.1 – Запуск процесса автоматической диагностики

N° п/п	Описание	Изображение	Примечание
1	Выбрать узел дерева объекта, который необходимо продиагностировать.		Если узел содержит в себе дочерние элементы (группы, узлы или маршрутные точки), то процесс диагностики запускается по всем этим элементам.
2	Выбрать из списка один из доступным диагностических скриптов (методик диагностики).		В окне отображается список файлов, находящихся в директории scripts текущего проекта.
3	Нажать кнопку [Выполнить диагностику] или воспользоваться командой контекстного меню, вызываемого нажатием ПКМ по выбранному элементу дерева объекта.	Вариант 1:  Вариант 2:	

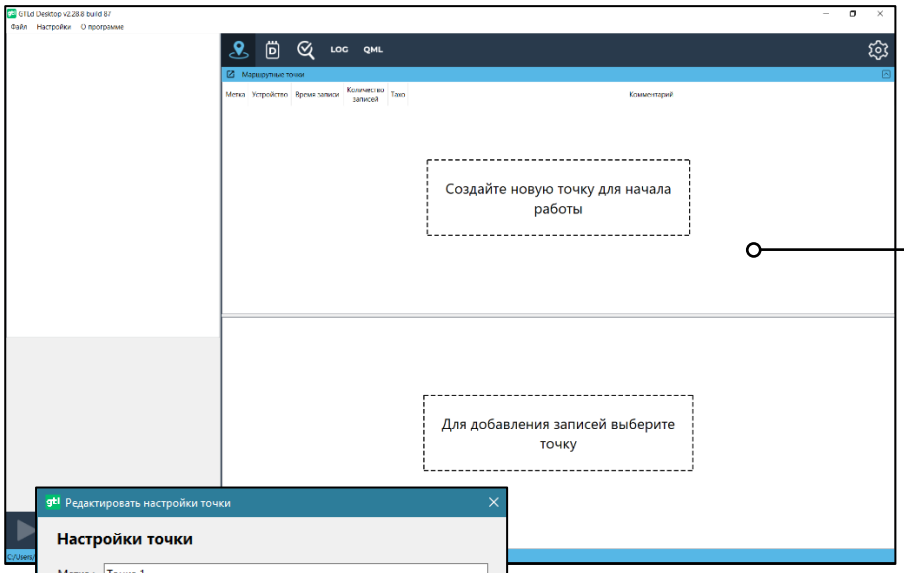
			
4	Указать тэг (идентификатор), под которым будут храниться результаты диагностики.		Тэг необходим для разделения режимов работы агрегата (при необходимости), а также используется на этапе отладки методик диагностики.
5	Указать программе использовать все записи для диагностики или только новые установив соответствующий чек бокс «Только новые записи».		При использовании режима «Только новые записи» программа будет игнорировать все записи, по которым уже имеется результат диагностики в базе данных.
6	Нажать кнопку [Старт] и дождаться завершения процесса вычислений.		Длительность процесса вычислений зависит от сложности диагностической методики и применяемых в ней инструментов математического анализа.

7	В блоке поиска выбрать из списка результат под необходимым тэгом и нажать кнопку [Выбрать].		В соответствии с настройками мастер-значений в таблице результатов произведется подсветка значений по системе «светофор».
8	Для получения более подробной информации по результатам автоматической диагностики выбрать в таблице интересующий результат.		
9	При необходимости воспользоваться инструментами глубокого анализа результатов автоматической диагностики согласно выбранной методики.		
10	Выдать заключение о техническом состоянии продиагностированного объекта.		

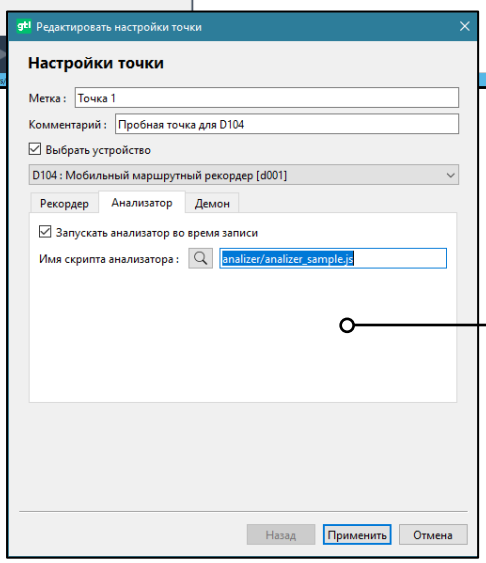
5. Планшетный режим

Начало работы в планшетном режиме дублирует пункты руководства 3.1-3.3.

После ввода данных и нажатия кнопки «Применить» откроется главное окно «Настройки проекта».

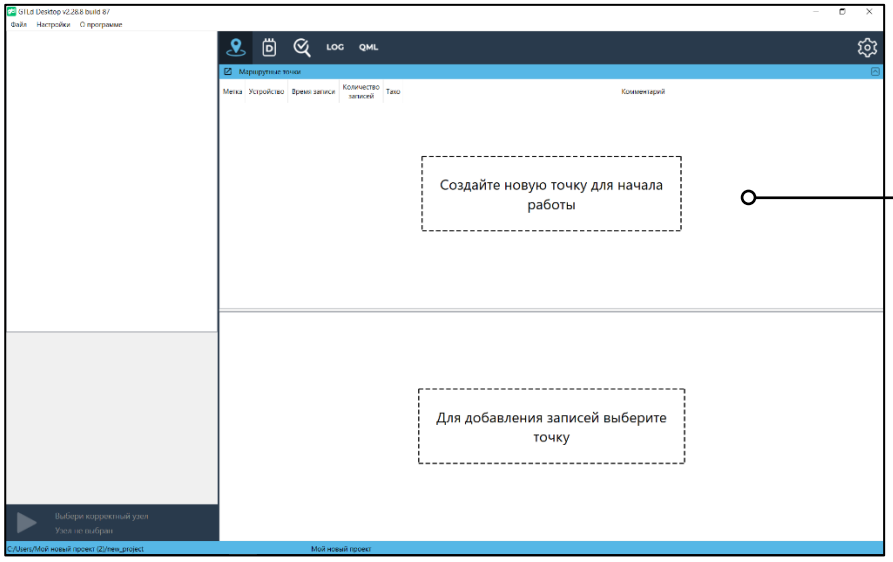


Окно настройки проекта при создании D104 проекта



Для работы с маршрутами выполните следующие шаги:

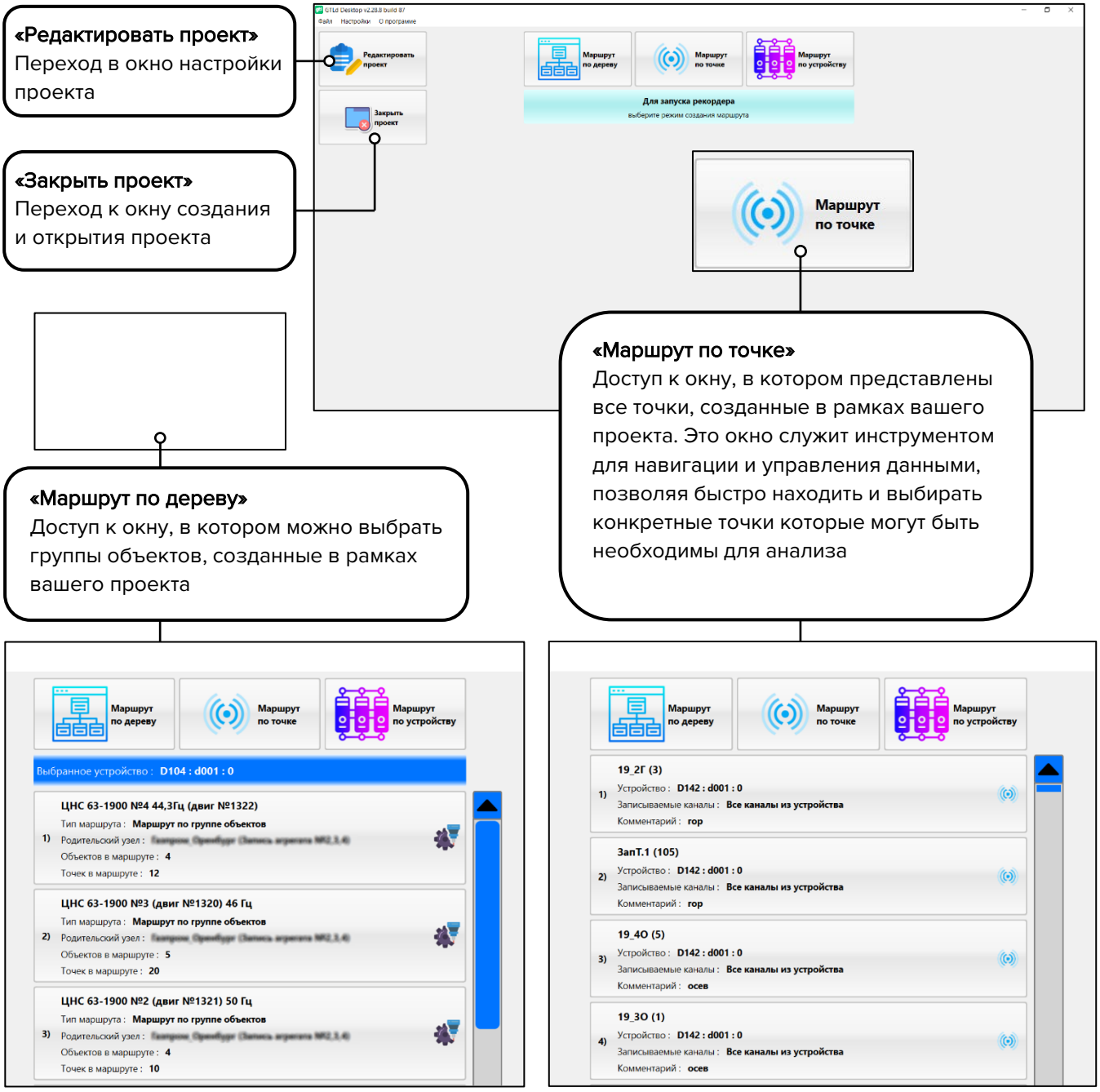
- в настройках проекта откройте окно древа проекта и нажмите «Добавить группу»;
- после выбора группы добавьте объект в выбранной группе во вкладке «Маршрутные точки»;
- нажмите «Добавить точку» и в окне «Метка» укажите название точки. В качестве устройства выберите D104;
- в окне «Анализатор» установите галочку напротив «Запускать анализатор во время записи».



Окно настройки проекта при создании пустого проекта

5.1 Выбор маршрута

После создания или выбора заранее созданного проекта пользователь переходит в окно «Выбор маршрута».

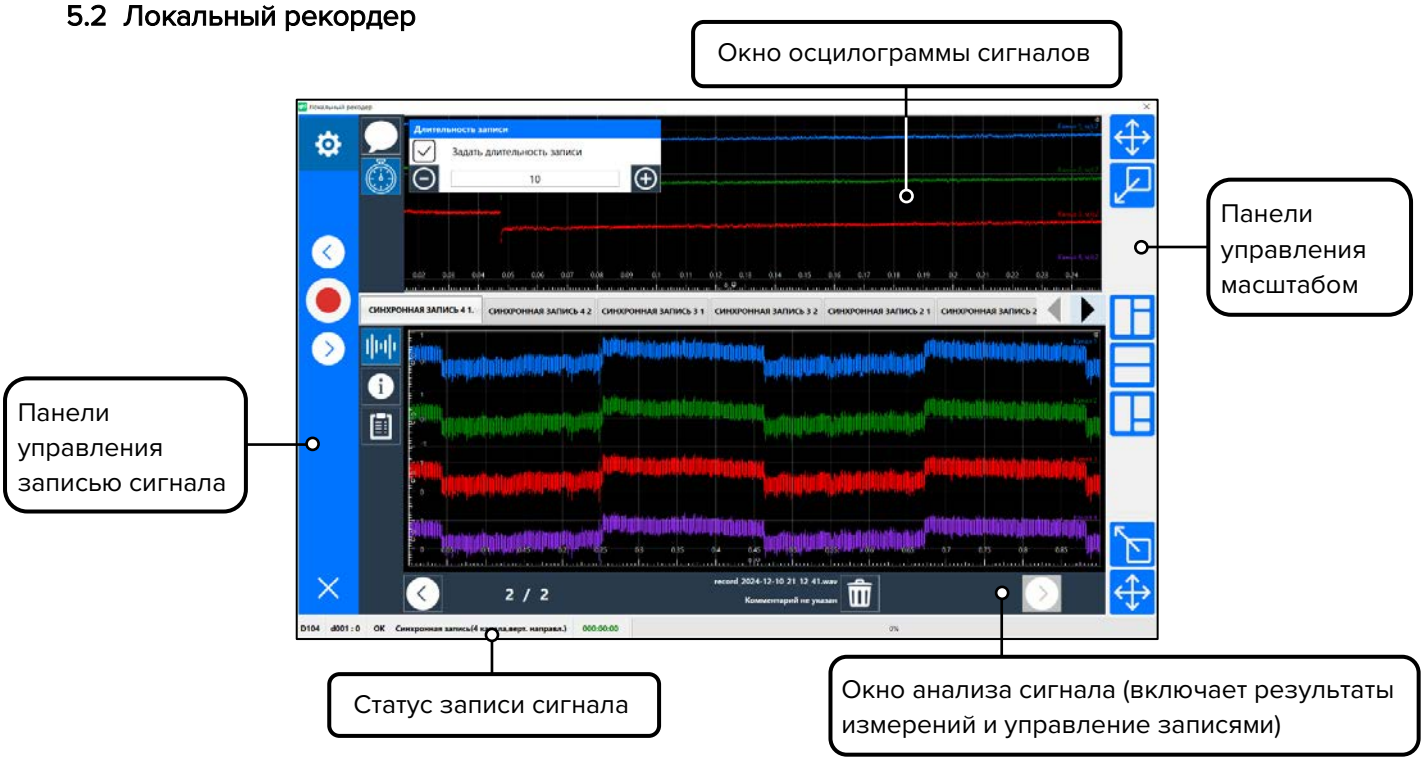


«Маршрут по устройству» – окно, в котором можно выбрать устройства, добавленные в соответствующие точки проекта. Это окно предоставляет интерфейс для управления и навигации по всем устройствам, связанным с конкретными точками, что позволяет находить необходимые элементы для дальнейшей работы.

По умолчанию виброанализатор работает с одним устройством, однако он также поддерживает возможность подключения к Wi-Fi сети, в которой могут находиться другие АЦП. В этом случае виброанализатор способен выполнять запись и обработку сигналов с этих устройств.

После выбора необходимого маршрута откроется окно локального рекордера.

5.2 Локальный рекордер

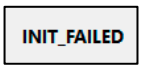


5.3 Панель управления записью сигнала

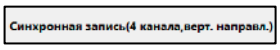
Функциональное назначение кнопок:

Кнопка инициализации АЦП- инициализирует АЦП.
В строке состояния отображается следующая информация:

Имя устройства



Статус инициализации:
DLE - инициализация не начата
OK - инициализация завершена успешно
INIT_FAILED - сбой в работе АЦП
INIT_IN_PROGRESS - инициализация в процессе



Имя измеряемой точки

Статус записи сигнала

Кнопка настройки параметров записи сигнала - открывает меню для настройки параметров записи.

Кнопка комментарий к записи - предназначена для добавления дополнительной информации к записям сигналов.

Она позволяет вводить важные заметки или уточнения при записи необходимых точек измерения. Это может быть особенно полезно для дальнейшей сортировки и анализа данных при выгрузке результатов в Excel – таблицу. Например, в строке комментария можно указать действительную частоту вращения агрегата, что значительно упрощает диагностику и анализ работы в дальнейшем. Наличие таких комментариев помогает лучше понимать контекст записей, облегчает идентификацию проблем и улучшает качество анализа данных в будущем.

Кнопка «Установки времени записи» – предоставляет возможность задать время начала записи сигнала без учета ранее установленного начального времени заданного в проекте.

Кнопки навигации по точкам маршрута – позволяет выбрать нужные точки для записи.

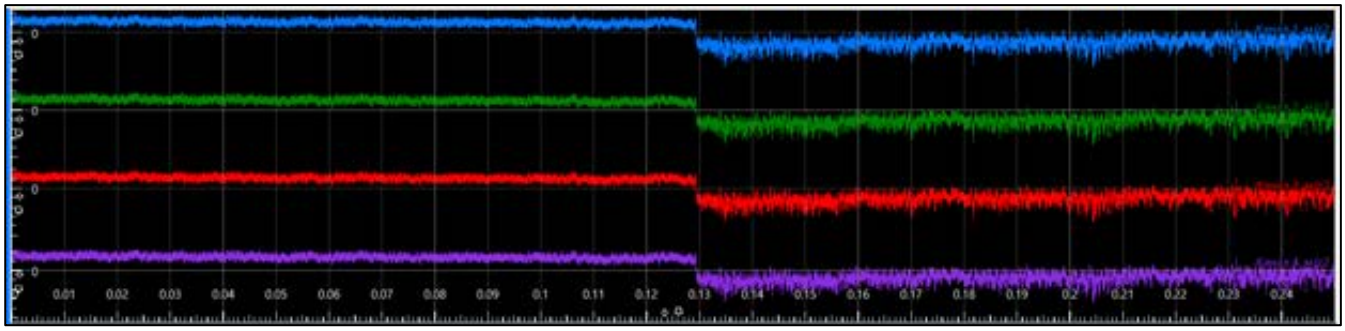
Кнопка начала записи сигнала – инициирует процесс записи и расчетов контролируемых параметров и спектров.

Кнопка выхода из анализатора – завершает работу с рекордером анализатора.

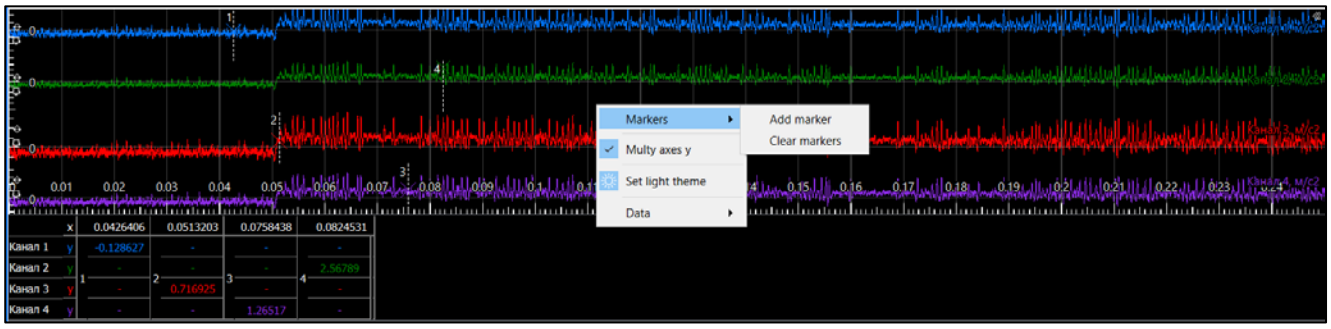
5.4 Окно осциллограммы сигналов

При инициализации рекордера в окне осциллограммы сигналов, на экране отображается график, представляющий электрические сигналы, поступающие с различных датчиков. Каждый канал графика подписан справа, что облегчает идентификацию каждого сигнала.

В данном окне также предусмотрена возможность масштабирования сигнала: пользователь может зажать экран пальцем и потянуть его, чтобы увеличить или уменьшить масштаб отображаемого сигнала. Кроме того, двойное нажатие на сигнал автоматически выполнит его масштабирование.



При долгом нажатии на экран открывается окно «Настройки интерфейса».



В этом окне пользователи могут:

- установить маркер на сигнал;
- включить или отключить мультиось* сигналов;
- выбрать светлую или темную тему оформления;
- сделать скриншот текущего состояния окна сигналов.

*Мультиось – это функция, позволяющая отображать данные с нескольких каналов на отдельных осях. При выключенном режиме все четыре канала с АЦП представлены на одной общей оси. Если включить мультиось, окно разбивается на четыре отдельных канала, где каждый сигнал имеет свою собственную ось X и Y.

5.5 Кнопки управления видом окна локального рекордера

Кнопки управления видом локального рекордера предоставляет ряд функции, позволяющий эффективно управлять отображением сигналов и результатами измерений.

Масштабирование в окне осциллограммы сигналов – данная кнопка автоматически подстраивает масштаб отображения сигналов, обеспечивая оптимальное предоставление данных в зависимости от их амплитуды и частоты. Это позволяет пользователя быстро адаптироваться к изменениям в сигнале и анализировать его характеристики.

Масштабирование окна осциллограммы сигналов - с помощью этой кнопки можно вручную изменять масштаб окна осциллограммы сигналов, что дает возможность детально рассмотреть определенные участки данных или наоборот, получить общий обзор. Особенно полезно при анализе сложных сигналов в режиме реального времени.

Скрыть окно осциллограммы сигналов и расширить окно анализа сигнала (окна результатов и управления записями, последний результат измерений)- нажатие этой кнопки позволяет скрыть окно осциллограммы сигналов, тем самым увеличивая пространство для анализа результатов и правления записями. Это удобно, когда необходимо сосредоточиться на анализе данных или последнем результате измерений.

Скрыть панель управления записью сигнала- данная функция позволяет скрыть панель управления, освобождая дополнительное пространство на экране. Это может быть полезно в ситуациях, когда управление записью не требуется.

Расширить окна осциллограммы сигналов- это кнопка позволяет увеличить размер окна осциллограммы сигналов, что улучшает видимость данных и упрощает их анализ.

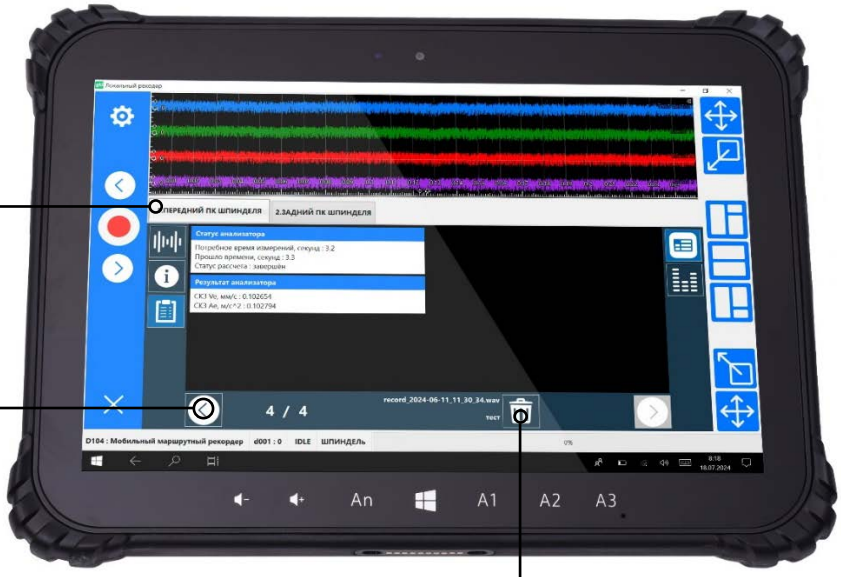
Масштабирование окна результатов и управления записью, последний результат измерений- с помощью этой кнопки пользователи могут изменять масштаб окна, отображающего результата измерения. Это позволяет более удобно просматривать и анализировать последние результаты.

Автомасштабирование в окне результатов и управления записью, последний результат измерений- данная функция автоматически настраивает масштаб в окне результатов, что дает возможность детально рассмотреть определенные участки данных или наоборот, получить общий обзор. Особенно полезно при анализе сложных записанных сигналов.

Таким образом, панель управления масштабом окна рекордера предоставляет гибкие инструменты для управления отображения данных.

5.6 Окно анализа сигнала (окна результатов и управления записями, последний результат измерений)

В окне анализа записанного сигнала представлены, кнопки, каждая из которых выполняет определенные функции, позволяя эффективно управлять анализом записанных сигналов.



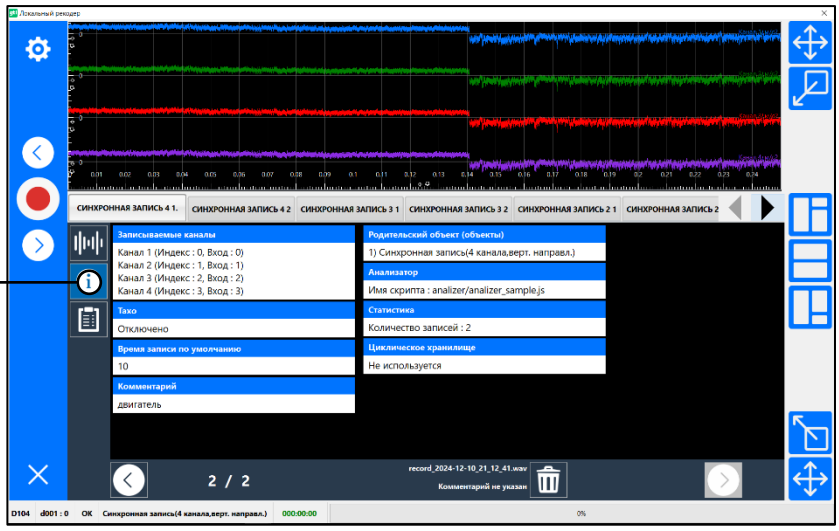
Выбор контрольной точки маршрута
Анализ и интерпретация данных

Оперативный просмотр истории записей
Позволяет легко перемещаться между различными записями сигналов, обеспечивая удобный доступ к необходимым записям. При удерживании кнопки окно переходит к самой первой или самой последней записи

Удаление записи сигнала
Возможность удаления выбранной записи сигнала точки. Позволяет управлять хранимыми данными и поддерживать порядок в записях

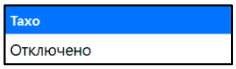
Окно сведений о настройках точки

В окне сведений о настройках точки представлена полная информация, необходимая для оценки параметров записи. В этом окне можно найти следующие данные:

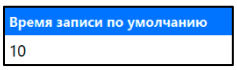


Информация о записи сигналов
Доступ к информации о характеристиках сигналов, зарегистрированных в текущей записи

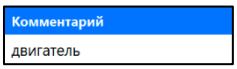
Записываемые каналы – настройка, где указываются каналы, которые будут записаны в данной точке. Например, [Канал 1]; [Канал 2]; [Канал 3]; [Канал 4]. Это позволяет пользователю быстро понять, сколько каналов было зафиксировано при записи сигнала. (Чтобы настроить данный параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки».Выберите нужную точку, затем в открывшемся окне «Настройка точки» перейдите на вкладку «Рекордер». Здесь вы сможете выбрать нужное количество каналов для записи).



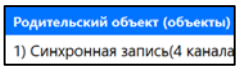
Тахо – параметр показывает, предусмотрена ли обработка тахо сигнала в настройках точки [Включено /Отключено]. Наличие тахометра может существенно повлиять на интерпретацию данных, особенно в случаях, когда требуется точная информация о скорости вращения или частоте измеряемого оборудования. (Чтобы настроить данный параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Менеджер устройств». Выберите устройство D104, затем в открывшемся окне «Настройка устройства» перейдите на вкладку «Настройка тахо». Если тахо-сигнал необходим, отметьте опцию «Использовать тахо-сигнал»).



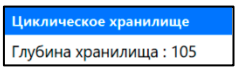
Время записи по умолчанию – отображается точно время записи сигнала. (Чтобы настроить данный параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки». Выберите нужную точку, затем в открывшемся окне «Настройка точки» в поле «Время записи по умолчанию» укажите необходимое время записи).



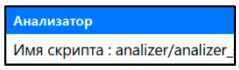
Комментарий – текстовые заметки или комментарии, которые могут содержать дополнительную информацию о записи, ее условиях или особенностях. Также может быть полезно для понимания контекста данных. (Чтобы настроить данный параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки». Выберите нужную точку, затем в открывшемся окне «Настройка точки» в поле «Комментарий» укажите необходимую информацию).



Родительский объект (объекты) – название объекта, с которым связана запись сигнала. (Для настройки данного параметра выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки». Выберите нужную точку и переместите её в соответствующую группу в дереве проекта).



Циклическое хранилища записей – позволяет предотвратить переполнение памяти устройства при выполнении большого количества записей, ограничив максимальное количество записей в одной точке, при включении данной опции старые записи будут автоматически перезаписаны. (Чтобы настроить этот параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки». Выберите необходимую точку, затем в открывшемся окне «Настройка точки» перейдите на вкладку «Рекордер». Установите галочку напротив «Использовать циклическое хранение записей» и задайте максимальное количество записей, которое будет храниться в данной точке).



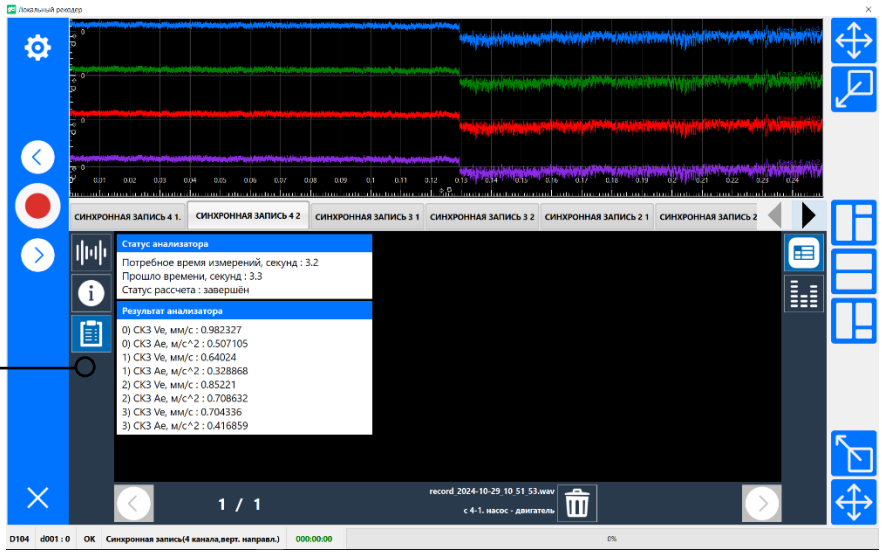
Анализатор (имя скрипта для анализа) – имя скрипта, настроенного в точке для анализа данных, помогает пользователю понять, какие методы и алгоритмы были применены к данным.

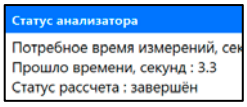
(Чтобы настроить этот параметр, выполните следующие шаги: откройте окно настройки проекта и перейдите во вкладку «Маршрутные точки». Выберите нужную точку, а затем в открывшемся окне «Настройка точки» перейдите на вкладку «Анализатор». В разделе «Имя скрипта анализатора» выберите необходимый скрипт, который будет использоваться для обработки записей в вашей точке).

Окно результатов анализатора

В окне результатов анализатора представлена важная информация о текущем статусе анализа сигнала.

Результаты анализатора
Окно с результатами анализа, позволяет просмотреть и оценить результаты обработки сигналов

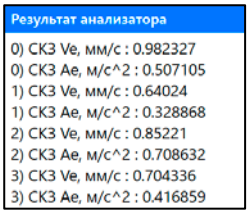




Статус анализатора.
Потребное время измерений (в секундах). Параметр показывает, сколько сигнального времени требуется для выполнения всех расчетов, заложенных в логику скрипта. Он позволяет оценить длительность процесса записи.

Прошедшее время (в секундах). Отображение фактического времени, прошедшее с момента начала измерений. Это значение позволяет отслеживать прогресс анализа.

Статус расчета. Указывается завершение расчета (завершен или не завершён). Критически важная информация, так как она информирует пользователя о готовности результатов и дальнейших действий.



Результат анализатора.
СКЗ (среднеквадратичное значение) виброскорости и виброускорения. Эти значения рассчитываются для каждого канала и позволяют оценить уровень вибраций, воздействующих на объект.

Окно предварительного анализа

Стандартный скрипт (analizer/analizer_sample.js), поставляемый с анализатором D104, предоставляет инструменты для анализа вибрационных сигналов. Он позволяет проводить различные расчеты и создавать спектры. Вот основные возможности, которые предлагает этот скрипт:

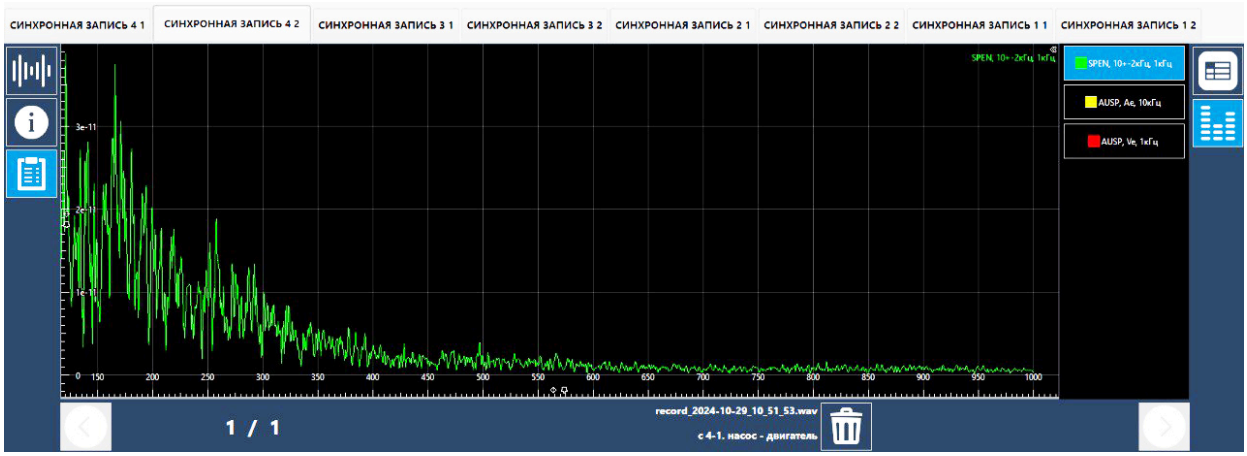
1. Расчет среднеквадратичного значения (СКЗ)

Скрипт рассчитывает среднеквадратичное значение (СКЗ) для виброускорения в диапазоне частот от 1 Гц до 10 кГц и виброскорости в диапазоне частот от 2 Гц до 1 кГц. Это параметр, который помогает оценить уровень вибрации и выявить возможные аномалии в работе оборудования.

2. Создание спектров

Скрипт поддерживает создание различных спектров, которые помогают в анализе сигналов:

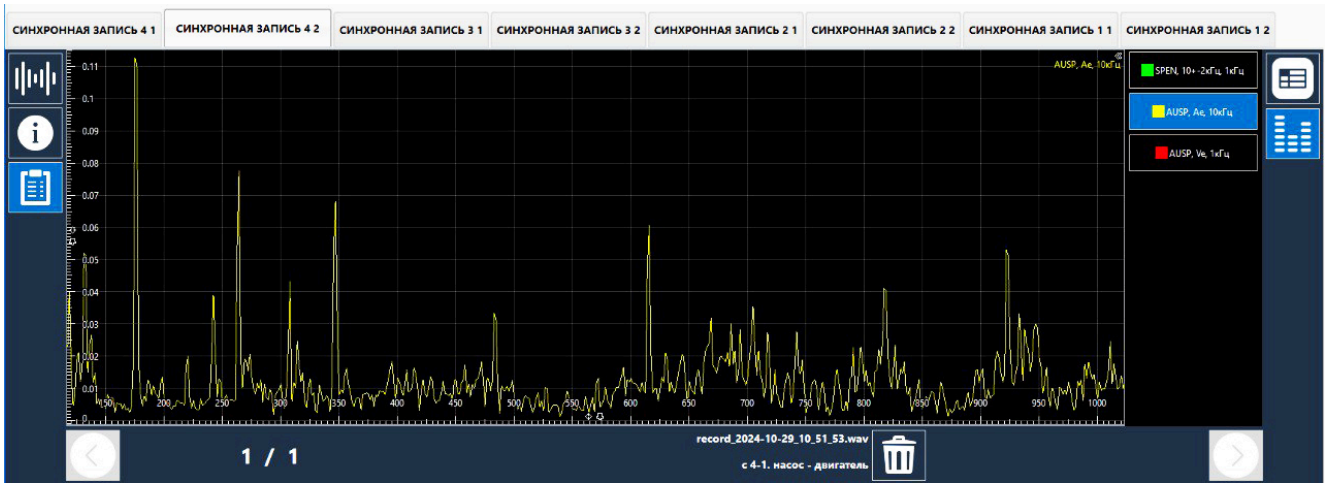
Спектр огибающей (SPEN)



Диапазон частот – обрабатывает частоты в диапазоне 10 ... 2 кГц.

Назначение – спектр огибающей помогает выявить модальные частоты и резонансные характеристики системы, а также позволяет анализировать амплитуду колебаний в высокочастотном диапазоне.

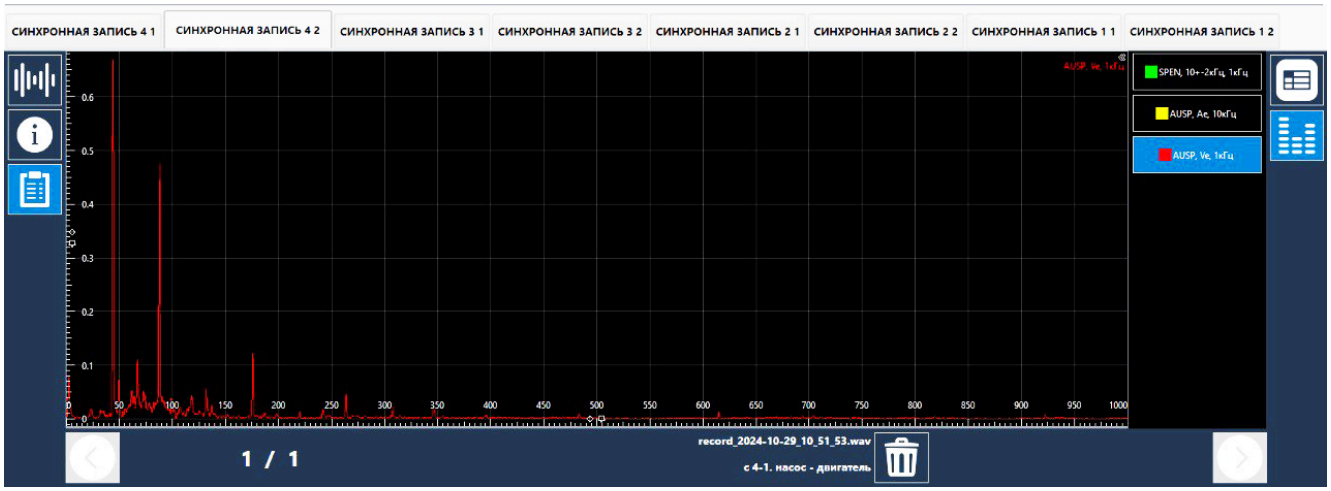
Автоспектр для ускорения (AUSP, Ae)



Диапазон частот: обрабатывает частоты 10 кГц.

Назначение: этот автоспектр позволяет анализировать вибрации, которые могут указывать на динамические изменения в работе оборудования, такие как дисбаланс, несоосность или механические повреждения.

Автоспектр для скорости (AUSP, Ve)



Диапазон частот: обрабатывает частоты до 1 кГц.

Назначение: автоспектр для скорости используется для анализа более низкочастотных вибраций, что может быть полезно для диагностики состояния подшипников, передачи и других механических компонентов.

3. Применение спектров

Спектры позволяют анализировать различные аспекты сигналов, такие как:

Вибрации: помогают выявить проблемы с балансировкой, несоосностью и другими механическими аномалиями.

Динамические характеристики объектов: позволяют оценить состояние оборудования и предсказать возможные отказы, что важно для планового обслуживания и предотвращения аварий.

6. Разработка алгоритмов и методик

Источник - https://docs.gtlab.pro/index.php/%D0%A1%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%87%D0%BD%D0%B8%D0%BA_GTLD:JS-API

6.1 Основные понятия.

Диагностические скрипты Gtld.script:
Скрипт или сценарий — это программа, которая выполняет конкретную задачу. Диагностический скрипт предназначен для анализа поступающих сигналов вибрации в конкретной точке объекта диагностики. Посредством данной программы реализуется конкретная методика диагностики. Результаты работы диагностического скрипта могут передаваться выше для дальнейшей групповой обработки и постановки общего диагноза состояния объекта диагностики.

Мастер-скрипты Gtld.masterscript:
Мастер скрипт или мастер сценарий — это программа, предназначенная для обработки результатов диагностических скриптов различных точек контроля. Результатом работы Мастер скрипта является общий диагноз состояния объекта диагностики с указанием проблем в конкретных его точках.

Мониторинговые скрипты Gtld.monitoringscript:
В разработке

6.2 Свойства и методы объекта gtld.node.

Объект gtld.node предоставляет API для доступа к свойствам и методам узлов дерева диагностируемого объекта.

6.2.1 Свойство gtld.node.name - QString gtld.node.name.

Содержит имя узла дерева.

Параметры:

Не предусмотрены.

Пример вызова:

```
let __name = gtld.node.name;
```

Результат:

В результате получаем строковую переменную, содержащую имя объекта, контейнера или точки.

6.2.2 Свойство gtld.node.uuid - QUuid gtld.node.uuid.

Содержит идентификатор узла дерева. Тип результата QUuid. Но большее практическое значение имеет идентификатор, преобразованный в строковую переменную с помощью метода `toString()`

Параметры:

Не предусмотрены.

Пример вызова:

```
let __uuid = gtld.node.uuid;
```

или

```
let __uuidString = gtld.node.uuid.toString();
```

Результат:

В результате получаем либо объект типа QUuid, если таковой требуется, либо строковую переменную, содержащую идентификатор узла дерева.

6.2.3 Свойство gtld.node.options - {} gtld.node.options.

Возвращает JSON-объект, содержащий опции узла.

Параметры:

Не предусмотрены.

Пример вызова:

```
let __options = gtld.node.options;
```

Результат:

В результате возвращается JSON-объект вида:

```
{ "tachoRatio", _tachoRatio },
{ "objectType", (int)_objectType },
{ "scriptName", _scriptName},
{ "masterScriptName", _masterScriptName},
{ "qmlScriptName", _qmlScriptName},
{ "name", data(Qt::DisplayRole).toString() },
// rolling bearing
{ "rbModelName", _rbModelName },
{ "rbVendor", _rbVendor },
{ "rbOuterD", _rbOuterD },
{ "rbInnerD", _rbInnerD },
{ "rbRollerD", _rbRollerD },
{ "rbRollerCount", _rbRollerCount },
{ "rbAngle", _rbAngle },
// Ball screw
{ "bsModelName", _bsModelName },
{ "bsVendor", _bsVendor },
{ "bsOuterD", _bsOuterD },
{ "bsInnerD", _bsInnerD },
{ "bsRollerD", _bsRollerD },
{ "bsRollerCount", _bsRollerCount },
{ "bsAngle", _bsAngle },
// Gear Transmitssion
```

```
{ "gtZ1", _gtZ1 },
{ "gtZ2", _gtZ2 },
// Belt Drive
{ "bdD1", _bdD1 },
{ "bdD2", _bdD2 },
{ "bdL", _bdL },
// Chain belt drive
{ "cbdZ1", _cbdZ1 },
{ "cbdZ2", _cbdZ2 },
{ "cbdZ3", _cbdZ3 },
// Pump
{ "pmBlades", _pmBlades },
// Planetary gear
{ "pgZ1", _pgZ1 },
{ "pgZ2", _pgZ2 },
{ "pgZ3", _pgZ3 },
{ "pgN", _pgN },
// Turbine
{ "trBlades", _trBlades },
```

Значение полей может устареть, актуальная информация в методе: main_model_item :: getOptions()

6.2.4 Свойство gtld.node.parent - main_model_item * gtld.node.parent.

Содержит ссылку на родителя выбранного узла. У возвращённого объекта будут доступны те же методы и свойства, что есть у объекта gtld.node.

Параметры:

Не предусмотрены.

Пример вызова:

```
let __parent = gtld.node.parent;  // берём родительский элемент
let __parentName = parent.name; // берём имя родительского элемента
```

Результат:

Ссылка на родительский объект.

6.2.5 Свойство gtld.node.children - [main_model_item *] gtld.node.children.

Возвращает массив ссылок на дочерние узлы.

Параметры:

Не предусмотрены.

Пример 1:

```
let __children = gtld.node.children;
```

Пример 2:

```
// для примера создадим функцию,
// возвращающую ссылку на дочерний элемент по его имени

getNodeByName = (searchNodeName) => {
  let __children = gtld.node.children;
  let __child = undefined;
  __children.forEach((node) => {
    if (node.name == searchNodeName) {
      __child = node;
    }
  });
  return __node;
}
```

С помощью функции getNodeByName получить uuid дочернего узла по его имени можно так:

```
gtl.log.info("uuid", getNodeByName("point").uuid.toString());
```

где "point" - имя дочернего узла.

Результат:

Результирующим значением является массив ссылок на дочерние объекты main_model_item, у него доступны такие же методы и свойства, как у объекта gtld.node.

6.2.6 Свойство gtld.node.lastResult - {} gtld.node.lastResult (для контейнеров и объектов), [{}] gtld.node.lastResult (для точек).

Данное свойство содержит параметры запуска процедуры диагностики: опции объекта, точки и рекорда; даты запуска диагностики и даты создания рекорда; идентификаторы объекта, точки и рекорда; таблицу мастер-значений; а также последний результат работы скрипта. Полный состав данных данного свойства варьируется в зависимости от типа узла. В зависимости от типа узла в данном свойстве сохраняется, либо результат работы **мастер-скрипта** (для групп и объектов), либо **диагностического скрипта** (для точек). Данные содержатся в виде JSON-объекта. Содержимое данного свойства обновляется при каждом запуске процедуры диагностики.

Параметры:

Не предусмотрены.

Примеры использования:

Выбрать весь объект результата:

```
let __lastResult = gtld.node.lastResult;
```

Выбрать только результаты работы скрипта (для объектов и групп):

```
let __resultJSON = gtld.node.lastResult.resultJSON;
```

для точек не забываем, что данное свойство является массивом объектов, поэтому требуется использование индекса, индекс 0 — это результат по последнему из обсчитанных рекордов:

```
let __resultJSON = gtld.node.lastResult[0].resultJSON;
```

Выбрать только параметры объекта:

```
let __objectOptions = gtld.node.lastResult.objectOptions;
```

Выбрать только параметры рекорда (только для точек):

```
let __recordOptions = gtld.node.lastResult[0].recordOptions;
```

Выбрать только параметры точки (только для точек):

```
let __pointOptions = gtld.node.lastResult[0].pointOptions;
```

Выбрать только мастер-значения (пример для объекта, для точек не забываем индекс):

```
let __masterValues = gtld.node.lastResult.masterValues;
```

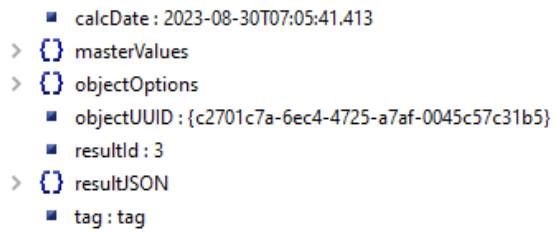
Вот таким образом можно обратиться к мастер значению с ключём "skz":

```
let __skz = gtld.node.lastResult.masterValues['skz'];
```

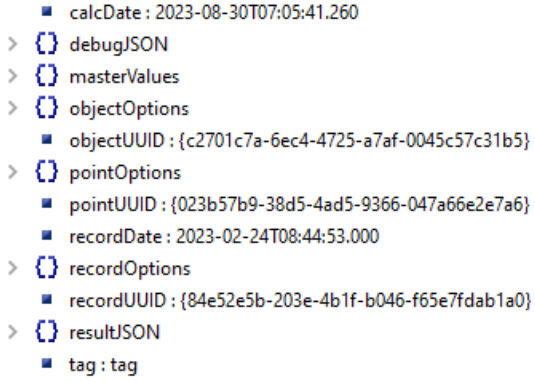
Результат:

Структура данных, хранимых в свойстве **lastResult** представлена на рисунках ниже. Структура данных зависит от типа узла дерева.

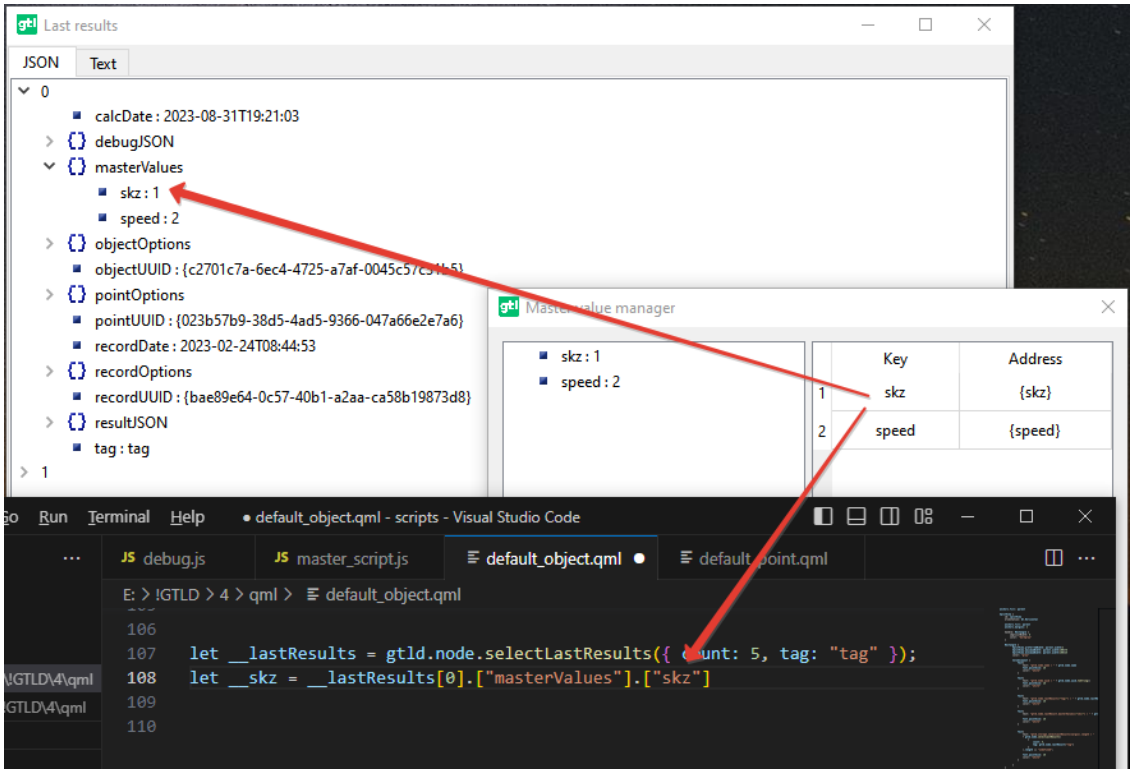
Для объектов и групп:



Для точек:



Для доступа к историческим данным используем объект **masterValues**. Ключами объекта **masterValues** являются ключи мастер-значений, сконфигурированные в менеджере мастер-значений.



6.2.7 Свойство gtld.node.selectLastResults - [{}] gtld.node.lastResults({}).

Данное метод осуществляет выборку массива исторических значений диагностики заданного узла. Формат данных, выбранных объектов, соответствует формату данных для свойства **gtld.node.lastResult;**

Параметры:

В качестве аргумента метода передаётся объект следующего вида:

```
{
  count: __count,
  tag: __tag,
}
```

1. **__count** - глубина выборки (количество записей, которые требуется выбрать);
2. **__tag** - тэг эксперимента.

Примеры использования:

Выбрать 5 последних результатов диагностики эксперимента с тэгом "tag":

```
let __lastResults = gtld.node.selectLastResults(
                {
                    count: 5,
                    tag: "tag",
                }); // получаем массив результатов, максимальный размер массива - 5
```

Результат:

Структура данных каждого элемента массива результатов, представлена на рисунках ниже. Структура данных зависит от типа узла дерева.

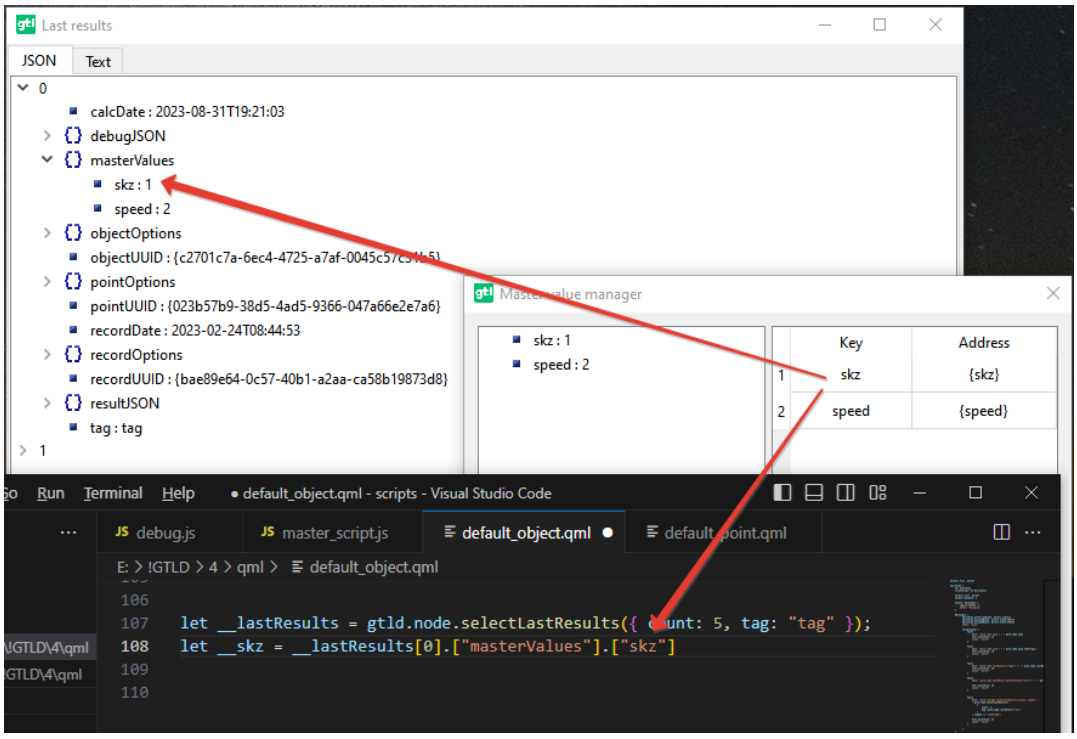
Для объектов и групп:

- calcDate : 2023-08-30T07:05:41.413
- masterValues
- objectOptions
- objectUUID : {c2701c7a-6ec4-4725-a7af-0045c57c31b5}
- resultId : 3
- resultJSON
- tag : tag

Для точек:

- calcDate : 2023-08-30T07:05:41.260
- debugJSON
- masterValues
- objectOptions
- objectUUID : {c2701c7a-6ec4-4725-a7af-0045c57c31b5}
- pointOptions
- pointUUID : {023b57b9-38d5-4ad5-9366-047a66e2e7a6}
- recordDate : 2023-02-24T08:44:53.000
- recordOptions
- recordUUID : {84e52e5b-203e-4b1f-b046-f65e7fdab1a0}
- resultJSON
- tag : tag

Для доступа к историческим данным используем объект masterValues. Ключами объекта masterValues являются ключи мастер-значений, сконфигурированные в менеджере мастер-значений.



6.3 Свойства и методы объекта gtld.storage

Объект gtld.storage предназначен для работы с историческими данными.

6.3.1 Метод gtld.storage.selectLastPointResults - [] gtld.storage.selectLastPointResults({})

(Вместо данного метода рекомендуется использовать метод gtld.node.selectLastResults())

Выборка крайних результатов работы диагностического скрипта точки. Каждый результат соответствует одному рекорду (а не одному запуску диагностики). Каждый результат помечен меткой (тегом), задаваемым пользователем. При выборке результатов требуется указать тег. Результат выбираются в порядке последний пришел – первым выбран. Таким образом получается, что на 0-м месте всегда лежит результат, соответствующий последнему по дате создания рекорду, т.е. самый актуальный результат.

Параметры:

В качестве аргумента метода передаётся объект следующего вида:

```
{
  count: __count,
  objectUUID: __objectUUID,
  pointUUID: __pointUUID,
  tag: __tag,
}
```

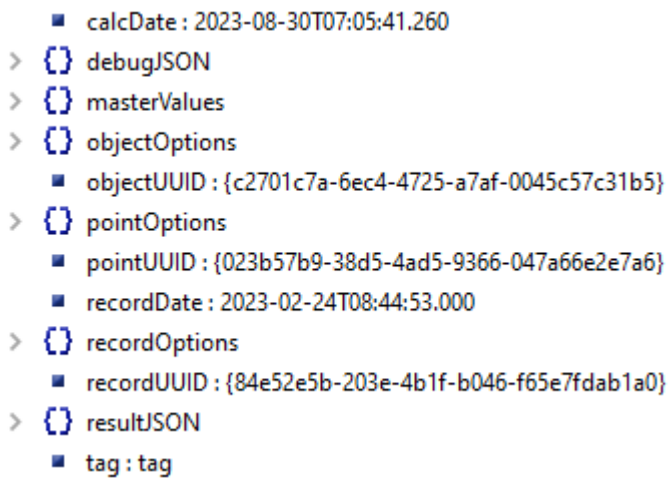
1. **__count** - глубина выборки (количество записей, которые требуется выбрать);
2. **__pointUUID** - идентификатор узла, для текущего узла можно использовать запись **gtld.node.uuid.toString();**
3. **__objectUUID** - идентификатор отцовского объекта, результат обработки узла дерева типа "точка" имеет значение только в привязке к отцовскому объекту, для получения UUID отцовского элемента текущего узла можно использовать запись вида: **gtld.node.parent.uuid.toString();**
4. **__tag** - тэг эксперимента.

Пример вызова:

```
let __results = gtld.storage.selectLastPointsResults({
  count: __count,
  objectUUID: __objectUUID,
  pointUUID: __pointUUID,
  tag: __tag,
});
```

Результат:

В результате возвращается массив JSON-объектов вида:



Каждый объект соответствует одной записи сигнала, объекты в массиве отсортированы по дате создания записи (последний - вперёд), самая младшая запись имеет индекс **0**. Т.е. под индексом **0** всегда находится результат диагностики, соответствующий актуальной (последней) записи сигнала.

6.3.2 Метод gtld.storage.selectLastObjectResults - [{}] gtld.storage.selectLastObjectResults({})

(Вместо данного метода рекомендуется использовать метод gtld.node.selectLastResults()).

Выборка последних результатов работы мастер-скрипта объекта или контейнера.

Параметры:

В качестве аргумента метода передаётся объект следующего вида:

```
{
  count: __count,
  objectUUID: __objectUUID,
  pointUUID: __pointUUID,
  tag: __tag,
}
```

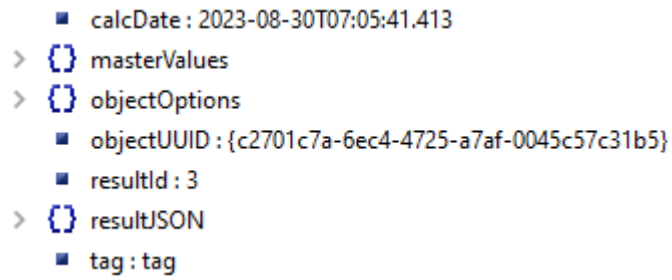
1. **__count** - глубина выборки (количество записей, которые требуется выбрать);
2. **__objectUUID** - идентификатор объекта, для текущего узла можно использовать запись вида: **gtld.node.uuid.toString();**
3. **__tag** - тэг эксперимента.

Пример вызова:

```
let __results = gtld.storage.selectLastObjectResults({
  count: __count,
  objectUUID: __objectUUID,
  tag: __tag,
});
```

Результат:

В результате возвращается массив JSON-объектов вида:



Каждый объект соответствует одному запуску процедуры диагностики, объекты в массиве отсортированы по дате выполнения процедуры диагностики (в отличии от выборки по точкам), самая младшая запись имеет индекс **0**. Т.е. под индексом **0** всегда находится актуальный (последний) результат диагностики.

6.4 Часто используемые функции JavaScript.

6.4.1 Использование массивов - Js arrays.

Массив - это особый тип объекта, предназначенный для работы с упорядоченным набором элементов.

Объявление массива

Для создания пустого массива данных используется вариант записи:

`let arr = [];`

Часто применяемые методы массива

`[i]` - доступ к элементу массива по индексу:

```
let arr = ['первый', 'второй'];
arr[0]; //получим доступ к значению 'первый'
```

`indexOf( значение )` - возвращает первый индекс, по которому элемент может быть найден в массиве. Если элемент отсутствует в массиве, то возвращается -1.

```
let arr = ['первый', 'второй', 'третий'];
arr.indexOf('второй'); // получим 1
```

`push` - предназначен для добавления нового элемента в конец массива:

```
let arr = ['первый', 'второй'];
arr.push('третий'); //добавим элемент 'третий' в конец массива
```

`pop` - удаляет последний элемент из массива:

```
let arr = ['первый', 'второй', 'третий'];
arr.pop(); //удаляем элемент 'третий' из массива
```

length - отражает длину массива или, если точнее, его последний цифровой индекс плюс один. Длина корректируется автоматически методами массива. Если мы уменьшаем length вручную, массив укорачивается.

```
let arr = ['первый', 'второй', 'третий'];
arr.length; // получим 3
```

Пример использования

Наглядным примером могут служить массивы амплитуд составляющих спектров вибрации:

```
ausp.data;
spen.data.
```

6.4.2 Использование объектов - Js objects.

Объекты - ассоциативные массивы данных и используются для хранения коллекций различных значений и более сложных сущностей по принципу "Ключ: Значение".

Объявление объекта

Для создания пустого объекта данных используется вариант записи:

```
let obj = {};
```

Часто применяемые методы объекта

Доступ к элементу объекта по ключу:

```
let obj = {
  name: "Aleksey",
  age: 40
};
obj.name; //получим доступ к значению "Aleksey"
obj["name"]; //альтернативный способ доступа к значению "Aleksey"
```

delete - удаление свойства:

```
let obj = {
  name: "Aleksey",
  age: 40
};
delete.age; //удаляем свойство "age" из объекта
```

in - проверяет существует ли свойство в объекте:

```
let obj = {
  name: "Aleksey",
  age: 40
};
```

```
"age" in obj; //вернет значение true
```

В качестве ключа может выступать значение переменной:

```
let obj = {};
let key1 = "name";
let key2 = 1021;

obj[key1] = "Aleksey"; //записываем пару name: "Aleksey";
obj[key2] = 1210; //записываем пару 1021: 1210;
```

Пример использования

Наглядным примером может служить формирование результата работы диагностического скрипта:

```
let result = {
  Result: true,
  AMPL: ampl_spen.value,
  RMS: rms_spen.value,
  PF: ampl_spen.value / rms_spen.value,
  Defects: Defect,
  Types: Defect_type,
  Square: AQ
};
```

6.4.3 Условное ветвление if(...) – Js if.

Инструкция `if(...)` вычисляет условие в скобках и, если результат `true`, то выполняет блок кода.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
if ( условие ) { операции при выполнении условия } else { операции в обратном случае };
```

Оператор `else` можно не использовать.

Пример использования

Наглядным примером может служить вычисление какого-либо параметра при определенном условии:

```
let min = 5;
if (freq > min) {
    let wim = 2 * Math.PI * freq;
    gtl.log.info("Угловая частота = ", wim);
}
else { gtl.log.info("Частота вращения меньше минимально рекомендуемой!") };
```

В данном примере если расчетная переменная `freq` окажется больше `min`, тогда произойдет расчет переменной `wim` и ее значение выведется в консоль. В противном случае выведется соответствующий текст предупреждения.

6.4.4 Конструкция ветвления switch - Js switch.

Конструкция `switch` заменяет собой сразу несколько `if(...)`. Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
switch (var) {
   case 'value1':
      break;
   case 'value2':
      break;
   case 'value3':
      break;
   default:
};
```

- Переменная `var` проверяется на строгое равенство первому значению `value1`, затем второму `value2` и так далее.
- Если соответствие установлено – `switch` начинает выполняться от соответствующей директивы `case` и далее, до ближайшего `break` (или до конца `switch`).
- Если ни один `case` не совпал – выполняется (если есть) вариант `default`.

Пример использования

Наглядным примером может служить определение уникального набора действий в зависимости от выбранного значения параметра `state` (например настройки из выпадающего списка):

```
switch (state) {
   case 0: //частота вращения определена из сигнала
      if (imp.FREQ() <= imp.FREQNESS()) {
         gtl.log.info("Частота вращения меньше рекомендуемой", "Минимально рекомендуемая
частота: " + imp.FREQNESS());
         gtl.diagnostic.stop(); //принудительная остановка диагностики
      };
      break;

   case 1: //частота вращения фиксированная
      spen.frequency = imp.standart_width();
      spen.lines = imp.standart_lines();
      break;

   case 2: //частота вращения из внешнего источника
      gtl.diagnostic.interval = acq_time;
      break;
};
```

В данном примере для каждого значения переменной `state` определен свой порядок действий (расчетов).

6.4.5 Цикл for - Js for.

Циклы предназначены для многократного повторения одного участка кода.

Синтаксис

Стандартная конструкция цикла `for` выглядит таким образом:

```
for (начало; условие; шаг) { ... тело цикла ... };
```

Пример использования

Наглядным примером может служить вывод в консоль всех обнаруженных составляющих на определенной частоте:

```
for (let i = 0; i <= 4; i++) {
    if (spen_BPFO.harms[i].is_present == true)
        gtl.log.info("Частота " + (i + 1) + "Fн", spen_BPFO.harms[i].amplitude);
};
```

В данном примере запускается цикл (пять повторений от 0 до 4 с шагом 1), в котором происходит проверка условия обнаружения `i-й` гармонической составляющей. При выполнении условия, выводится значение её амплитуды в консоль.

6.5 Функции инициализации диагностики GTLd:

6.5.1 Общая информация - Gtld common info;

Функции данного раздела предназначены для подготовки данных перед запуском основной функции диагностики, содержащей в себе логику постановки автоматического диагноза.

Источник сигнала вибрации

Для обращения к источнику сигнала используется общая запись:

```
gtl.analog_inputs[i]
```

i - индекс канала вибрации

Если требуется представить результаты измерения спектров вибрации в относительных единицах (дБ), то необходимо установить соответствующее опорное значение:

```
gtl.analog_inputs[0].reference = 1e-6;
```

1 м/с2 = 120 дБ виброускорения (0 дБ = 10-6 м/с2)
1 мм/с = 120 дБ виброскорости (0 дБ = 10-6 мм/с)
1 мкм = 120 дБ вибросмещения (0 дБ = 10-6 мкм)

6.5.2 Фильтрация сигнала - Gtl.add filter irr.

Функция предназначена для фильтрации сигнала. Необходима для выделения в сигнале частотных диапазонов при проведении конкретных видов анализа вибрации.

Объявление функции

```
var filter = gtl.add_filter_irr( номер измерительного канала );
```

Свойства (методы)

`filter.kind = gtl.filter_irr.butterworth;` - тип окна;

```
bessel;
butterworth;
chebyshev1,
chebyshev2,
elliptic,
legendre,
rbj,
undef.
```

`filter.type = gtl.filter_irr.lowpass;` - тип фильтра;

```
lowpass;
highpass;
bandpass.
```

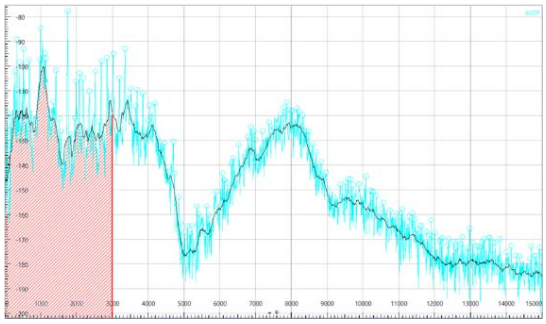
`filter.order = 4;` - порядок фильтра;

`filter.frequency = 3000;` - граничная частота фильтра (для полосового фильтра - центральная частота), Гц;

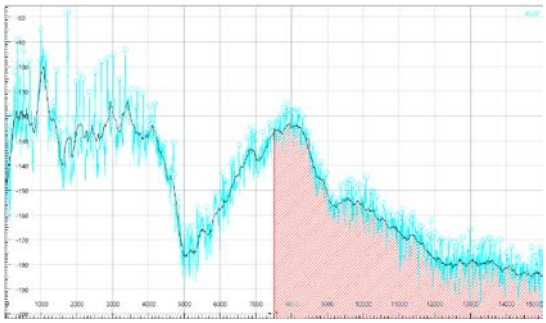
`filter.width = 1250;` - ширина полосы фильтра (для полосового фильтра), Гц;

Пример использования

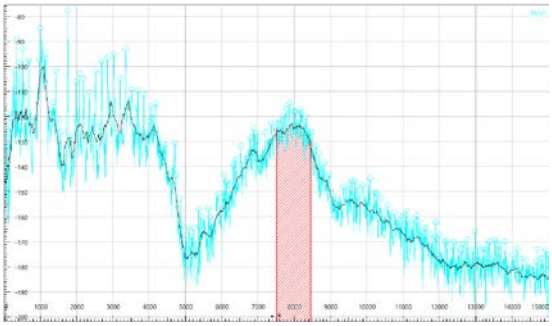
```
//фильтрация сигнала перед определением частоты вращения из сигнала
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 10; //граничная частота фильтра

//определение частоты вращения
var freq = gtl.add_value_freq(filter); //объявление переменной частоты вращения
freq.time = 1; //время выборки
freq.avg_cnt = 6; //количество усреднений

gtl.diagnostic.interval = freq.time * freq.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Частота вращения", freq.value);
}
```



lowpass



highpass



Bandpass

6.5.3 Определение частоты вращения - Gtl.add value freq.

Функция предназначена для определения частоты вращения из сигнала вибрации.

Объявление функции

```
var freq = gtl.add_value_freq( фильтр );
```

Свойства (методы)

`freq.time = 1;` - время выборки, сек;

`freq.avg_cnt = 6;` - количество отсчетов для усреднения;

`freq.dc = -0.05;` - уровень, при переходе через который считаются периоды;

Пример использования

```
//фильтрация сигнала перед определением частоты вращения из сигнала
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 10; //граничная частота фильтра
```

```
//определение частоты вращения
var freq = gtl.add_value_freq(filter); //объявление переменной частоты вращения
freq.time = 1; //время выборки
freq.avg_cnt = 6; //количество усреднений

gtl.diagnostic.interval = freq.time * freq.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Частота вращения", freq.value);
};
```

6.5.4 Интегрирование сигнала - Gtl.add intg.

Функция предназначена для интегрирования сигнала вибрации. Необходима для получения значений параметров виброскорости и виброперемещения из исходного сигнала виброускорения. Интегрирование можно производить как до фильтрации сигнала, так и после, а также несколько раз.

Объявление функции

`var integ = gtl.add_intg( номер измерительного канала или фильтр );`

Свойства (методы)

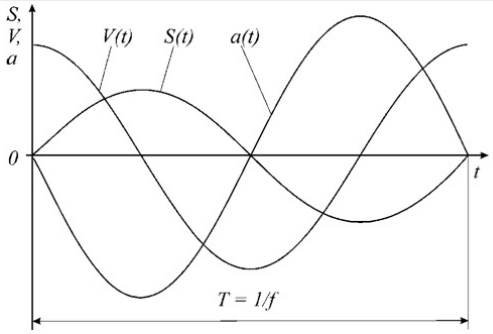
`integ.taps = 1;` - степень интегрирования (1-2 в зависимости от исходного сигнала);

`integ.scale = 1000;` - коэффициент масштабирования (перевод величин);

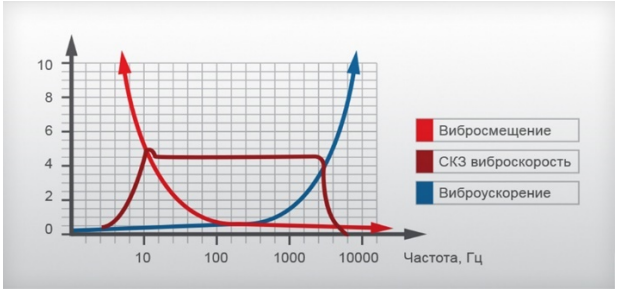
Пример использования

```
//фильтрация сигнала в диапазоне 10-1000 Гц
var filter = gtl.add_filter_iir(gtl.analog_inputs[0]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
filter.order = 10; //порядок фильтра
filter.frequency = 505; //центральная частота полосового фильтра
filter.width = 990; //ширина полосы фильтра

//интегрирование исходного сигнала виброускорения (преобразуем в виброскорость)
var integ = gtl.add_intg(filter); //интегрирование
integ.taps = 1; //степень интегрирования (скорость из ускорения - 1-нарное интегрирование)
integ.scale = 1000; //переводим метры в миллиметры

//определение среднего квадратического значения виброскорости в диапазоне 10-1000 Гц
var rms = gtl.add_value_rms(integ); //объявление переменной СКЗ
rms.time = 1; //время выборки
rms.avg_cnt = 4; //количество усреднений

gtl.diagnostic.interval = rms.time * rms.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("СКЗ виброскорости в диапазоне 10-1000 Гц", rms.value);
```

```
};
```



Виброперемещение, виброскорость и виброускорение (фазы)



Виброперемещение, виброскорость и виброускорение (частотные характеристики)

6.5.5 Получение скользящего среднего значения - Gtl.create moving avg.

Функция предназначена для получения скользящего среднего значения для последующей обработки различными методами.

Объявление функции

```
let __avg = gtl.create_moving_avg(
  {
    src : gtl.analog_inputs[0],
    name : "avg",
    time : 0.1
  }
);
```

Свойства (методы)

`__avg.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__avg.name = "avg";` - присвоение имени скользящего среднего значения;

`__avg.time = 0.1;` - время выборки данных;

Пример использования

```
//построение графика скользящего среднего значения
let time = 1;
let __avg = gtl.create_moving_avg(
  {
    src : gtl.analog_inputs[0],
```

```
    name : "avg",
    time : time
  }
);

__avg.history = time;
gtl.diagnostic.interval = time;
let plot = gtl.plots.add("avg values");

function diagnose()
{
  plot.add(
    {
      color: 0x0000ff,
      name: __avg.name,
      x: 1 / __avg.rate,
      y: __avg.getHistoryArray()
    }
  );

  gtl.diagnostic.stop();
};
```

6.5.6 Получение скользящего значения смещения - Gtl.create moving offset.

Функция предназначена для получения скользящего значения смещения для последующей обработки различными методами.

Объявление функции

```
let __offset = gtl.create_moving_offset(
  {
    src : gtl.analog_inputs[0],
    name : "offset",
    time : 0.1
  }
);
```

Свойства (методы)

`__offset.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__offset.name = "offset";` - присвоение имени скользящего значения смещения;

`__offset.time = 0.1;` - время выборки данных;

Пример использования

```
//построение графика скользящего значения смещения
let time = 1;
let __offset = gtl.create_moving_offset(
  {
    src : gtl.analog_inputs[0],
```

```
    name : "offset",
    time : time
  }
);

__offset.history = time;
gtl.diagnostic.interval = time;
let plot = gtl.plots.add("offset values");

function diagnose()
{
  plot.add(
    {
      color: 0x0000ff,
      name: __offset.name,
      x: 1 / __offset.rate,
      y: __offset.getHistoryArray()
    }
  );

  gtl.diagnostic.stop();
};
```

6.5.7 Получение скользящего значения частоты вращения - Gtl.create moving freq.

Функция предназначена для получения скользящего значения частоты вращения для последующей обработки различными методами.

Объявление функции

```
let __freq = gtl.create_moving_freq(
  {
    src : gtl.analog_inputs[0],
    name : "freq",
    time : 0.1,
    dc: 0
  }
);
```

Свойства (методы)

`__freq.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__freq.name = "freq";` - присвоение имени скользящего значения частоты вращения;

`__freq.time = 0.1;` - время выборки данных;

`__freq.dc = 0;` - уровень, при переходе через который считаются периоды;

Пример использования

```
//построение графика скользящего значения смещения
let time = 1;
let __freq = gtl.create_moving_freq(
```

```
  {
    src : gtl.analog_inputs[0],
    name : "freq",
    time : time
  }
);

__freq.history = time;
gtl.diagnostic.interval = time;
let plot = gtl.plots.add("freq values");

function diagnose()
{
  plot.add(
    {
      color: 0x0000ff,
      name: __freq.name,
      x: 1 / __freq.rate,
      y: __freq.getHistoryArray()
    }
  );

  gtl.diagnostic.stop();
};
```

6.5.8 Получение скользящего значения максимальных амплитуд - Gtl.create moving max.

Функция предназначена для получения скользящего значения максимальных амплитуд для последующей обработки различными методами.

Объявление функции

```
let __max = gtl.create_moving_max(
  {
    src : gtl.analog_inputs[0],
    name : "max",
    time : 0.1
  }
);
```

Свойства (методы)

`__max.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__max.name = "max";` - присвоение имени скользящего значения максимальных амплитуд;

`__max.time = 0.1;` - время выборки данных;

Пример использования

```
//построение спектра
let __max = gtl.create_moving_max(
  {
```

```
    src : gtl.analog_inputs[0],
    name : "max",
    time : 0.1
  }
);

let plot1 = gtl.plots.add("plot1");
let ausp = gtl.create_ausp(
   {
     "src": __max,
     "frequency": 1000,
     "resolution": 1,
     "average": 3,
     "overlap": 0.5,
     "window": gtl.spec.rectangular,
     "view": gtl.spec.unit
   }
);

gtl.diagnostic.interval = gtl.acq_time;

function diagnose()
{
  plot1.add(
    {
      color: 0x0000ff,
      name: "ausp",
      x: ausp.resolution,
      y: ausp.data
    }
  );

  gtl.diagnostic.stop();
};
```

6.5.9 Получение скользящего значения минимальных амплитуд - Gtl.create moving min.

Функция предназначена для получения скользящего значения минимальных амплитуд для последующей обработки различными методами.

Объявление функции

```
let __min = gtl.create_moving_min(
  {
    src : gtl.analog_inputs[0],
    name : "min",
    time : 0.1
  }
);
```

Свойства (методы)

`__min.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__min.name = "min";` - присвоение имени скользящего значения минимальных амплитуд;

`__min.time = 0.1;` - время выборки данных;

Пример использования

```
//построение спектра
let __min = gtl.create_moving_min(
  {
    src : gtl.analog_inputs[0],
    name : "min",
    time : 0.1
  }
);

let plot1 = gtl.plots.add("plot1");
let ausp = gtl.create_ausp(
    {
      "src": __min,
      "frequency": 1000,
      "resolution": 1,
      "average": 3,
      "overlap": 0.5,
      "window": gtl.spec.rectangular,
      "view": gtl.spec.unit
    }
);

gtl.diagnostic.interval = gtl.acq_time;

function diagnose()
{
  plot1.add(
    {
      color: 0x0000ff,
      name: "ausp",
      x: ausp.resolution,
      y: ausp.data
    }
  );

  gtl.diagnostic.stop();
};
```

6.5.10 Получение скользящего значения размаха (Пик-Пик) - Gtl.create moving peak to peak.

Функция предназначена для получения скользящего значения размаха амплитуд (Пик-Пик) для последующей обработки различными методами.

Объявление функции

```
let __ptp = gtl.create_moving_peak_to_peak(
  {
    src: gtl.analog_inputs[0],
    name: "peak_to_peak",
    time: 0.1
  }
);
```

Свойства (методы)

`__ptp.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__ptp.name = "peak_to_peak";` - присвоение имени скользящего значения размаха амплитуд;

`__ptp.time = 0.1;` - время выборки данных;

Пример использования

```
//построение спектра
let __ptp = gtl.create_moving_peak_to_peak(
  {
    src: gtl.analog_inputs[0],
    name: "peak_to_peak",
    time: 0.1
  }
);

let plot1 = gtl.plots.add("plot1");
let ausp = gtl.create_ausp(
    {
      "src": __ptp,
      "frequency": 1000,
      "resolution": 1,
      "average": 3,
      "overlap": 0.5,
      "window": gtl.spec.rectangular,
      "view": gtl.spec.unit
    }
);

gtl.diagnostic.interval = gtl.acq_time;

function diagnose()
{
  plot1.add(
    {
      color: 0x0000ff,
      name: "ausp",
      x: ausp.resolution,
      y: ausp.data
    }
```

```
  );

  gtl.diagnostic.stop();
};
```

6.5.11 Получение скользящего значения корреляции - Gtl.create moving corr.

Функция предназначена для получения скользящего значения корреляции для последующей обработки различными методами.

Объявление функции

```
let __corr = gtl.create_moving_corr(
  {
    src: gtl.analog_inputs[0],
    name: "corr",
    method: gtl.corr.fft,
    reference: gtl.io.from_csv("E:/data/gtl/flick.txt")
  }
);
```

Свойства (методы)

`__corr.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`__corr.name = "corr";` - присвоение имени скользящего значения корреляции;

`__corr.method = gtl.corr.fft;` - метод расчета;

```
fft - с помощью БПФ (самый быстрый)
faltung - свертка (результат тот же, но медленней. Нужен для отладки)
factor - рассчитывается коэффициент корреляции в диапазоне [-1, 1];
```

`__corr.reference = gtl.io.from_csv("E:/data/gtl/flick.txt");` - файл референса. Последовательность отсчетов (фрагмент сигнала), который необходимо искать;

Пример использования

```
//построение корреляционной функции
let time = 1;
let __corr = gtl.create_moving_corr(
    {
      src: gtl.analog_inputs[0],
      name: "corr",
      method: gtl.corr.fft,
      reference: gtl.io.from_csv("E:/data/gtl/flick.txt")
    }
);

gtl.analog_inputs[0].history = time;
__corr.history = time;

let plot1 = gtl.plots.add("corr");
```

```
gtl.diagnostic.interval = time;
function diagnose()
{
  plot1.add(
    {
      color: 0x0000ff,
      name: gtl.analog_inputs[0].name,
      x: 1.0/gtl.analog_inputs[0].rate,
      y: gtl.analog_inputs[0].getHistoryArray()
    }
  )

  plot1.add(
    {
      color: 0xff0000,
      name: __corr.name,
      x: 1.0/__corr.rate,
      y: __corr.getHistoryArray()
    }
  )

  gtl.diagnostic.stop();
}
```

6.5.12 Получение массива значений из сигнала (истории) - Gtl.history.

Функции предназначена для получения значений сигнала за указанный временной интервал (прямого или обработанного) в виде массива данных для дальнейшей математической обработки. Размер массива пропорционален времени запроса и частоте дискретизации сигнала;

Источник сигнала вибрации

```
let __input = gtl.analog_inputs[i];
```

i - индекс канала вибрации

Свойства (методы)

`__input.history = 1;` - размер сохраняемых данных (время запроса), сек;

`let __history = __input.getHistoryArray();` - получение массива значений;

Пример использования

```
let __input0 = gtl.analog_inputs[0];
__input0.history = 1; //размер сохраняемых данных (время запроса)

gtl.diagnostic.interval = 1;
function diagnose() {
  let __history = __input0.getHistoryArray(); //получаем массив данных

  for (let i = 0; i < 10; i++) { gtl.log.info("value" + i, __history[i]) }; //выводим первые 10 значений
  gtl.diagnostic.stop();
```

```
};
```

6.5.13 Определение АФЧХ сигнала - Gtl.add apfc.

Функция предназначена для анализа АФЧХ сигналов. Применяется для решения различных задач анализа параметров сигналов вибрации, определения их когерентности и сопутствующих характеристик. Часто используется в виброналадке и для определения собственных частот колебания системы. Входными данными служат сигналы (массивы).

Объявление функции

```
var apfc = gtl.add_apfc(
   {
     src1: gtl.analog_inputs[0],
     src2: gtl.analog_inputs[1],
     name: "coh",
     color: 0xff0000,
     visible: true,
     freq: 1000.0,
     window: gtl.spec.rectangular,
     resolution: 1.0,
     average: 1,
     overlap: 0,
     afc: gtl.apfc.coherence,
     pfc: gtl.apfc.deg
   }
);
```

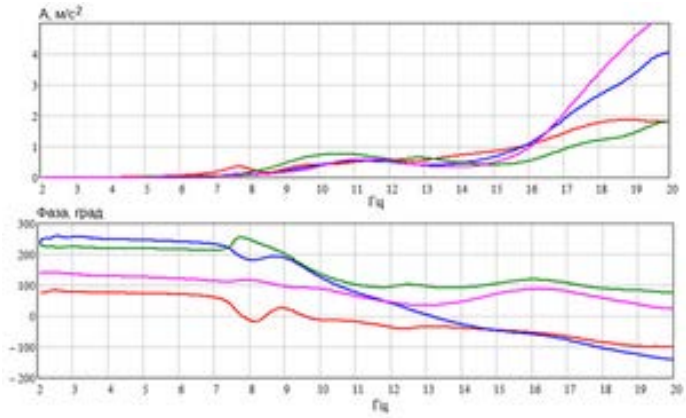
Свойства (методы)

В разработке

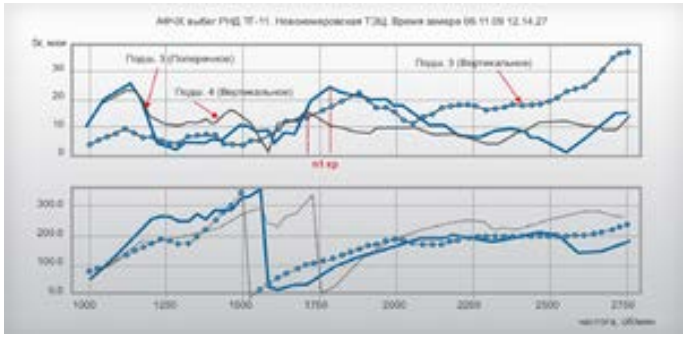
Пример использования

```
var apfc = gtl.add_apfc(
   {
     "src1" : gtl.analog_inputs[0],
     "src2" : gtl.analog_inputs[1],
     "name" : "coh",
     "color" : 0xff0000,
     "visible" : true,
     "freq" : 1000.0,
     "window" : gtl.spec.rectangular,
     "resolution" : 1.0,
     "average" : 1,
     "overlap" : 0,
     "afc" : gtl.apfc.coherence,
     "pfc" : gtl.apfc.deg
   }
);

var apfc1 = gtl.add_apfc(
```

```
    {
      "src1" : gtl.analog_inputs[0],
      "src2" : gtl.analog_inputs[1],
      "name" : "mag",
      "color" : 0x0000ff,
      "visible" : true,
      "freq" : 1000.0,
      "window" : gtl.spec.rectangular,
      "resolution" : 1.0,
      "average" : 1,
      "overlap" : 0,
      "afc" : gtl.apfc.magnitude,
      "pfc" : gtl.apfc.deg
    }
);

gtl.diagnostic.interval = apfc.acq_time+0.1;

function diagnose()
{
  gtl.log.info("afc", apfc.data[50]);
  gtl.log.info("pfc", apfc.phase[50]);
  gtl.log.info("acq_time", apfc.acq_time+0.1);

  gtl.diagnostic.stop();
};
```



Амплитудно-частотная и фазо-частотная характеристики



Метод разгон-выбег на примере АФЧХ сигнала вибрации

6.5.14 Определение АЧХ сигнала - Gtl.create afc.

Функция предназначена для анализа АЧХ сигналов. Применяется для решения различных задач анализа параметров сигналов вибрации. Часто используется в виброналадке и для определения собственых частот колебания системы. Входными данными служат сигналы.

Объявление функции

```
var afc = gtl.create_afc(
   {
     "src0": gtl.analog_inputs[0],
     "src1": gtl.analog_inputs[1],
     "frequency": 1000,
     "resolution": 1,
     "average": 20,
     "overlap": 0.5,
     "window": gtl.spec.rectangular,
     "view": gtl.spec.unit
   }
);
```

Свойства (методы)

`afc.src0 = gtl.analog_inputs[0];` - источник первого сигнала;

`afc.src1 = gtl.analog_inputs[1];` - источник второго сигнала;

`afc.frequency = 1000;` - граничная частота, Гц;

`afc.resolution= 1;` - частотное разрешение;

`afc.average = 20;` - количество усреднений;

`afc.overlap = 0.5;` - коэффициент перекрытия;

`afc.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

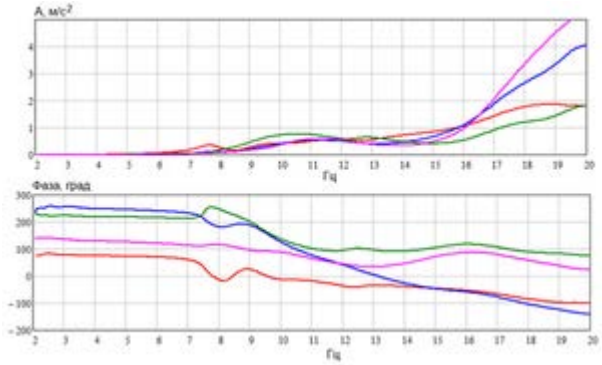
`afc.view = gtl.spec.unit;` - единицы измерения;

```
db;
unit;
rms;
```

`afc.data;` - массив рассчитанных значений;

Пример использования

```
var afc = gtl.create_afc(
   {
     "src0": gtl.analog_inputs[0],
     "src1": gtl.analog_inputs[1],
     "frequency": 1000,
     "resolution": 1,
     "average": 20,
     "overlap": 0.5,
     "window": gtl.spec.rectangular,
     "view": gtl.spec.unit
   }
);

gtl.log.info("acq time", afc.acq_time);
gtl.diagnostic.interval = afc.acq_time + 0.1 // пока так, исправим;

let plot = gtl.plots.add("plot");

function diagnose()
{
  plot.add(
    {
      color: 0x0000ff,
      name: "afc",
      x: afc.resolution,
      y: afc.data
    }
  )

  gtl.diagnostic.stop();
}
```



Амплитудно-частотная и фазо-частотная характеристики

6.5.15 Определение ФЧХ сигнала - Gtl.create pfc.

Функция предназначена для анализа ФЧХ сигналов. Применяется для решения различных задач анализа параметров сигналов вибрации и определения разности фаз колебаний. Входными данными служат сигналы.

Объявление функции

```
var phase = gtl.create_pfc(
   {
     src0: gtl.analog_inputs[0],
     src1: gtl.analog_inputs[1],
     frequency: 1000,
     resolution: 1,
     average: 1,
     overlap: 0,
     window: gtl.spec.rectangular,
     view: gtl.phase.deg,
     range: gtl.phase.positive
   }
);
```

Свойства (методы)

`phase.src0 = gtl.analog_inputs[0];` - источник первого сигнала;

`phase.src1 = gtl.analog_inputs[1];` - источник второго сигнала;

`phase.frequency = 1000;` - граничная частота, Гц;

`phase.resolution = 1;` - частотное разрешение;

`phase.average = 1;` - количество усреднений;

`phase.overlap = 0;` - коэффициент перекрытия;

`phase.window = gtl.spec.rectangular;` - тип оконной функции;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`phase.wiev = gtl.phase.deg;` - отображение фазы в градусах (радианах);

```
deg;
rad;
```

`phase.range = gtl.phase.nagative;` - тип отображения фазы (отрицательные и положительные значения);

```
nagative;
positive;
```

`phase.data;` - массив рассчитанных значений;

Пример использования

```
var phase = gtl.create_pfc(
   {
      src0: gtl.analog_inputs[0],
      src1: gtl.analog_inputs[1],
      frequency: 1000,
      resolution: 1,
      average: 1,
      overlap: 0,
      window: gtl.spec.rectangular,
      view: gtl.phase.deg,
      range: gtl.phase.positive
   }
);

gtl.diagnostic.interval = phase.acq_time + 1;
gtl.log.info("acq time", phase.acq_time);

let plot1 = gtl.plots.add("plot1");

function diagnose() {
   plot1.add(
      {
         color: 0x0000ff,
         name: "delta_phase",
         x: phase.resolution,
         y: phase.data,
      }
   )

   gtl.diagnostic.stop();
}
```

6.5.16 Построение фазового спектра - Gtl.create phase.

Функция предназначена для построения фазового спектра (множество начальных фаз гармонических колебаний кратных частот). Применяется для решения специфических задач анализа параметров сигналов вибрации. Входными данными служат сигналы.

Объявление функции

```
var phase = gtl.create_phase(
   {
     src: gtl.analog_inputs[0],
     frequency: 1000,
     resolution: 1,
     average: 1,
     overlap: 0,
     window: gtl.spec.rectangular,
     view: gtl.phase.deg,
     range: gtl.phase.negative
```

```
    }
);
```

Свойства (методы)

`phase.src = gtl.analog_inputs[0];` - источник сигнала;

`phase.frequency = 1000;` - граничная частота, Гц;

`phase.resolution = 1;` - частотное разрешение;

`phase.average = 1;` - количество усреднений;

`phase.overlap = 0;` - коэффициент перекрытия;

`phase.window = gtl.spec.rectangular;` - тип оконной функции;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`phase.wiev = gtl.phase.deg;` - отображение фазы в градусах (радианах);

```
deg;
rad;
```

`phase.range = gtl.phase.nagative;` - тип отображения фазы (отрицательные и положительные значения);

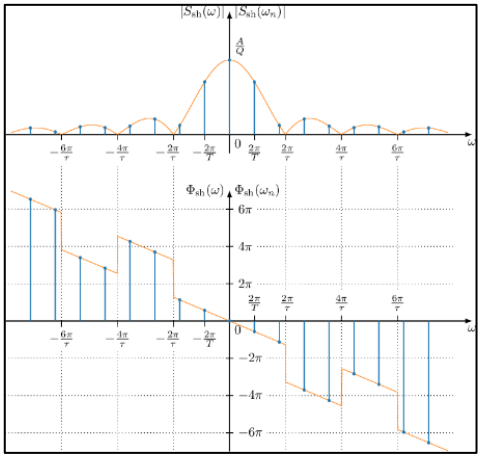
```
nagative;
positive;
```

`phase.data;` - массив рассчитанных значений;

Пример использования

```
var phase = gtl.create_phase(
  {
    src: gtl.analog_inputs[0],
    frequency: 1000,
    resolution: 1,
    average: 1,
    overlap: 0,
    window: gtl.spec.rectangular,
    view: gtl.phase.deg,
    range: gtl.phase.negative
  }
);
```

```
var phase1 = gtl.create_phase(
  {
    src: gtl.analog_inputs[0],
    frequency: 1000,
    resolution: 1,
    average: 1,
    overlap: 0,
    window: gtl.spec.rectangular,
    range: gtl.phase.positive
  }
);

gtl.diagnostic.interval = phase.acq_time;
gtl.log.info("acq time", phase.acq_time);

let plot1 = gtl.plots.add("plot1");

function diagnose() {
  plot1.add(
    {
      color: 0x0000ff,
      name: "phase",
      x: phase.resolution,
      y: phase.data,
    }
  )

  plot1.add(
    {
      color: 0xff0000,
      name: "phase1",
      x: phase1.resolution,
      y: phase1.data
    }
  )

  gtl.diagnostic.stop();
}
```



Амплитудный и фазовый спектры сдвинутой во времени периодической последовательности прямоугольных импульсов.

6.5.17 Разница фаз моногармонических сигналов - Gtl.add delta phase.

Функция предназначена для определения разности фаз колебаний моногармонических сигналов вибрации. Входными данными служат сигналы (массивы).

Объявление функции

`var delta_phase = gtl.add_delta_phase( номер измерительного канала 1, номер измерительного канала 2 );`

Свойства (методы)

`delta_phase.value;` - значение разницы фаз, рад;

Пример использования

```
//разность фаз моногармонических сигналов
var delta = gtl.add_delta_phase(gtl.analog_inputs[0], gtl.analog_inputs[1]);

gtl.diagnostic.interval = delta.acq_time;

function diagnose()
{
  gtl.log.info("Разница фаз", delta.value);

  gtl.diagnostic.stop();
};
```

Фаза колебаний

6.5.18 Разница фаз полигармонических сигналов - Gtl.add delta phase spec.

Функция предназначена для определения разности фаз колебаний полигармонических сигналов вибрации. Входными данными служат сигналы (массивы).

Объявление функции

`var phase = gtl.add_delta_phase_spec( номер измерительного канала 1, номер измерительного канала 2 );`

Свойства (методы)

`phase.max_frequency = 1000;` - граничная частота спектра вычисления фазы, Гц;

`phase.resolution = 1;` - частотное разрешение, Гц;

`phase.frequency = 100;` - частота, для которой вычисляется разница фаз, Гц;

`phase.value;` - значение разницы фаз, рад;

Пример использования

```
//разность фаз полигармонических сигналов
var delta = gtl.add_delta_phase_spec(gtl.analog_inputs[0], gtl.analog_inputs[1]);
delta.max_frequency = 1000; //граничная частота спектров вычисления фазы;
delta.resolution = 10; //частотное разрешение спектров;
delta.frequency = 100; //частота, на которой вычисляется разница фаз;

gtl.diagnostic.interval = delta.acq_time;

function diagnose()
{
  gtl.log.info("Разница фаз на частоте " + delta.max_frequency + " Гц", delta.value);

  gtl.diagnostic.stop();
};
```

6.5.19 Коэффициент когерентности сигналов - Gtl.add coherence.

Функция предназначена для определения коэффициента когерентности полигармонических сигналов вибрации и отражает степень линейной взаимосвязи гармонических компонент рассматриваемых процессов. Входными данными служат сигналы (массивы).

Объявление функции

`var coh = gtl.add_coherence( номер измерительного канала 1, номер измерительного канала 2 );`

Свойства (методы)

`coh.frequency = 1000;` - граничная частота, Гц;

`coh.resolution = 1;` - частотное разрешение, Гц;

`coh.threshold = 1;` - порог обнаружения частотной составляющей;

`coh.acq_time;` - время набора данных и вычисления коэффициента, с;

`coh.data;` - массив значений коэффициента;

Объявление функции (альтернативный метод)

```
var coh = gtl.create_coh(
   {
     "src0" : gtl.analog_inputs[0],
     "src1" : gtl.analog_inputs[1],
     "frequency" : 1000,
     "resolution" : 1,
```

```
    "average" : 6,
    "overlap" : 0.5,
    "window" : gtl.spec.rectangular
  }
);
```

Свойства (методы)

`coh.src0 = gtl.analog_inputs[0];` - источник первого сигнала (номер измерительного канала или переменная параметра);

`coh.src1 = gtl.analog_inputs[1];` - источник второго сигнала (номер измерительного канала или переменная параметра);

`coh.frequency = 1000;` - граничная частота, Гц;

`coh.resolution = 1;` - частотное разрешение, Гц;

`coh.average = 6;` - количество усреднений;

`coh.overlap = 0.5;` - коэффициент перекрытия;

`coh.gtl.spec.rectangular;` - тип оконной функции;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`coh.acq_time;` - время набора данных и вычисления коэффициента, с;

`coh.data;` - массив значений коэффициента;

Пример использования

```
//коэффициент когерентности сигналов
var coh = gtl.add_coherence(gtl.analog_inputs[0], gtl.analog_inputs[1]);
coh.frequency = 1000;
coh.resolution = 1;
coh.threshold = 1;

gtl.diagnostic.interval = coh.acq_time;

function diagnose()
{
//значение коэффициента для частоты 110 Гц
gtl.log.info("Когерентность", coh.data[110]);

gtl.diagnostic.stop();
};
```

```
//коэффициент когерентности сигналов (альтернативный вариант)
var coh = gtl.create_coh(
   {
     "src0" : gtl.analog_inputs[0],
     "src1" : gtl.analog_inputs[1],
     "frequency" : 1000,
     "resolution" : 1,
     "average" : 6,
     "overlap" : 0.5,
     "window" : gtl.spec.rectangular
   }
);

gtl.log.info("Время, необходимое для вычисления когерентности:", coh.acq_time);
gtl.diagnostic.interval = coh.acq_time + 0.1 //в процессе исправления;

function diagnose()
{
  gtl.plot.add(
    {
      color: 0x0000ff,
      name: "coherence",
      x: coh.resolution,
      y: coh.data
    };
  );

  gtl.diagnostic.stop();
};
```

6.5.20 Определение СКЗ - Gtl.add value rms.

Функция предназначена для определения среднего квадратического значения вибрации в указанном диапазоне.

Объявление функции

`var rms = gtl.add_value_rms( фильтр );`

Свойства (методы)

`rms.name = "RMS";` - присвоение имени измерения СКЗ;

`rms.time = 0.5;` - интервал расчета СКЗ, сек;

`rms.avg_cnt = 4;` - количество отсчетов для усреднения;

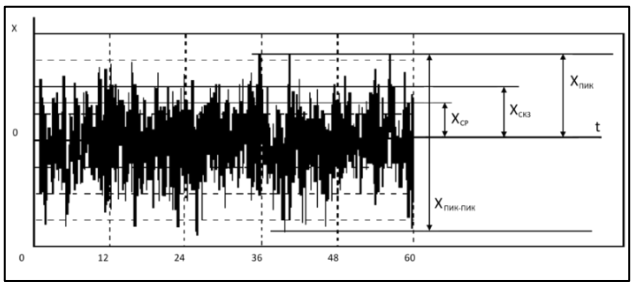
Результат

`rms.value;` - расчетное значение СКЗ;

`rms.values;` - массив, содержащий указанное количество отсчетов для усреднения;

Пример использования

```
//фильтрация диапазона в сигнале для определения СКЗ вибрации
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 1000; //граничная частота фильтра

//определение среднего квадратического значения в диапазоне до 1000 Гц
var rms = gtl.add_value_rms(filter); //объявление переменной СКЗ
rms.time = 1; //время выборки
rms.avg_cnt = 4; //количество усреднений

gtl.diagnostic.interval = rms.time * rms.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
  gtl.log.info("СКЗ в диапазоне до 1000 Гц", rms.value);
  gtl.diagnostic.stop();
};
```



СКЗ

6.5.21 Определение максимального значения СКЗ в заданном временном интервале - Gtl.add max data.

Функция предназначена для определения максимального значения параметра в указанном временном интервале наблюдения.

Объявление функции

```
var max = gtl.add_max_data(
   {
     "src" : rms,
     "name" : "max",
     "color" : 0x00ff00,
     "time" : 1
   }
);
```

Свойства (методы)

`max.src = gtl.analog_inputs[0];` - источник сигнала (номер измерительного канала или переменная параметра);

`max.name = "max";` - присвоение имени параметра;

`max.color = 0xff0000;` - цвет линии в формате HEX;

`max.time = 1.0;` - время выборки, сек.;

`max.value_at;` - массив значений параметра;

`max.back_value;` - доступ к последнему значению массива;

`max.size;` - длина массива значений (зависит от времени выборки и частоты дискретизации сигнала);

Пример использования

```
//определение среднего квадратического значения
var rms = gtl.add_rms_data(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "rms",
     "color" : 0xff0000,
     "time" : .1
   }
);
//определение максимального значения СКЗ в интервале времени 1 сек
var max = gtl.add_max_data(
   {
     "src" : rms,
     "name" : "max",
     "color" : 0x00ff00,
     "time" : 1.0
   }
);

gtl.diagnostic.interval = 1;

function diagnose()
{
  gtl.log.info("size", rms.size);
  gtl.log.info("max", max.value_at(max.size - 1));
  gtl.diagnostic.stop();
};
```

6.5.22 Определение амплитудных (максимальных) значений - Gtl.add value ampl.

Функция предназначена для определения амплитудных (максимальных) значений вибрации в отфильтрованном диапазоне.

Объявление функции

`let __ampl = gtl.add_value_ampl( фильтр );`

Свойства (методы)

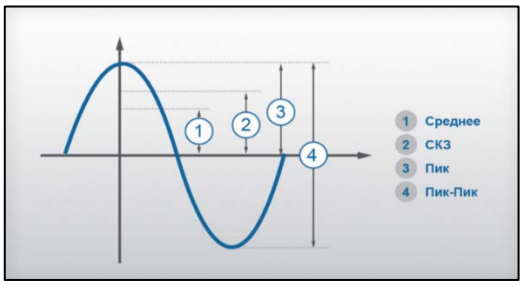
`__ampl.name = "Amax";` - присвоение имени измерения амплитуды;

`__ampl.time = 0.5;` - интервал расчета амплитуды, сек;

`__ampl.avg_cnt = 4;` - количество отсчетов для усреднения;

Пример использования

```
//фильтрация диапазона в сигнале для определения максимальной амплитуды вибрации
var filter = gtl.add_filter_iir(gtl.analog_inputs[2]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 1000; //граничная частота фильтра

//определение амплитудного значения в диапазоне до 1000 Гц
let __ampl= gtl.add_value_ampl(filter); //объявление переменной амплитуды вибрации
__ampl.time = 1; //время выборки
__ampl.avg_cnt = 4; //количество усреднений

gtl.diagnostic.interval = __ampl.time * __ampl.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Амплитуда вибрации в диапазоне до 1000 Гц", __ampl.value);
};
```



Амплитуда вибрации

6.5.23 Определение размаха (Пик-Пик) - Gtl.add value peak to peak.

Функция предназначена для определения размаха колебаний в отфильтрованном диапазоне (сигнале).

Объявление функции

`let __ptp = gtl.add_value_peak_to_peak( фильтр );`

Свойства (методы)

`__ptp.name = "Peak-To-Peak";` - присвоение имени измерения размаха;

`__ptp.time = 0.5;` - интервал расчета размаха, сек;

`__ptp.avg_cnt = 4;` - количество отсчетов для усреднения;

Результат

`__ptp.value;` - расчетное значение размаха колебаний;

`__ptp.values;` - массив, содержащий указанное количество отсчетов для усреднения;

Пример использования

```
//фильтрация диапазона в сигнале для определения максимальной амплитуды вибрации
var filter = gtl.add_filter_iir(gtl.analog_inputs[0]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 1000; //граничная частота фильтра

//определение амплитудного значения в диапазоне до 1000 Гц
let __ptp = gtl.add_value_peak_to_peak(filter); //объявление переменной размаха
__ptp.time = 1; //время выборки
__ptp.avg_cnt = 4; //количество усреднений

gtl.diagnostic.interval = gtl.acq_time; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Размах колебаний в диапазоне до 1000 Гц", __ptp.value);
};
```

6.5.24 Определение коэффициента эксцесса (куртозис) - Gtl.add value kurt.

Функция предназначена для определения коэффициента эксцесса (куртозис) в отфильтрованном диапазоне или по прямому сигналу.

Объявление функции

`var kurt = gtl.add_value_kurt( фильтр );`

Свойства (методы)

`kurt.name = "Kurt";` - присвоение имени измерения коэффициента эксцесса;

`kurt.time = 0.5;` - интервал расчета коэффициента эксцесса, сек;

`kurt.avg_cnt = 4;` - количество отсчетов для усреднения;

Результат

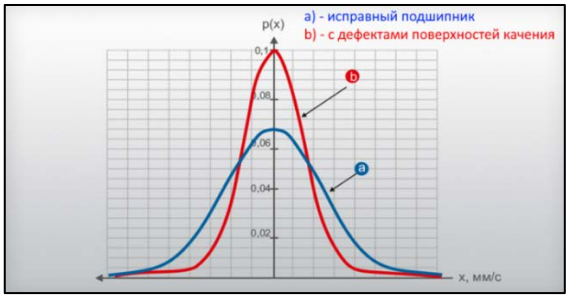
`kurt.value;` - расчетное значение коэффициента эксцесса;

`kurt.values;` - массив, содержащий указанное количество отсчетов для усреднения;

Пример использования

```
//фильтрация сигнала перед определением коэффициента эксцесса
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 1000; //граничная частота фильтра

//определение коэффициента эксцесса в диапазоне до 1000 Гц
var kurt = gtl.add_value_kurt(filter); //объявление переменной частоты вращения
kurt.time = 0.5; //время выборки
```

```
kurt.avg_cnt = 4; //количество усреднений

gtl.diagnostic.interval = kurt.time * kurt.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Коэффициент эксцесса в диапазоне до 1000 Гц", kurt.value);
};
```



Коэффициент эксцесса

6.5.25 Определение коэффициента корреляции - Gtl.add corr.

Функция предназначена для определения коэффициента корреляции двух параметров (сигналов). Применяется для определения взаимосвязи между двумя параметрами. Для графического представления корреляционной связи часто используется прямоугольная система координат с осями, которые соответствуют обеим переменным (диаграмма рассеяния параметров).

Объявление функции

```
var corr = gtl.create_corr(
   {
     "src0" : gtl.analog_inputs[0],
     "src1" : gtl.analog_inputs[1],
     "time" : 1.0,
     "window" : gtl.spec.rectangular
   }
);
```

Свойства (методы)

`corr.src1 = gtl.analog_inputs[0];` - источник первого сигнала (номер измерительного канала или переменная параметра);

`corr.src2 = gtl.analog_inputs[1];` - источник второго сигнала (номер измерительного канала или переменная параметра);

`corr.time = 1.0;` - время выборки, сек.;

`corr.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
cosin;
hann;
bartlett_hann;
```

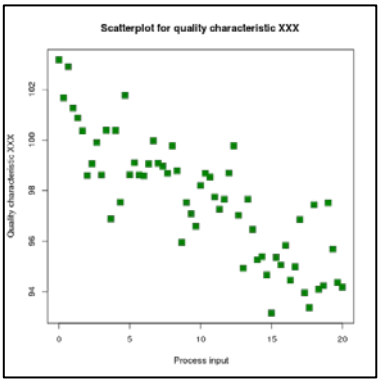
```
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`corr.resolution;` - частотное разрешение;

`corr.data;` - массив результатов вычисления коэффициента корреляции.

Пример использования

```
//определение коэффициента корреляции
var corr = gtl.create_corr(
   {
     "src0" : gtl.analog_inputs[0],
     "src1" : gtl.analog_inputs[1],
     "time" : 0.15635*1,
     "window" : gtl.spec.hann
   }
);

gtl.log.info("acq time", corr.acq_time);
gtl.diagnostic.interval = corr.acq_time+.1;

let plot = gtl.plots.add("plot");

function diagnose() {

  plot.add(
   {
     color: 0x0000ff,
     name: "corr",
     x: corr.resolution,
     y: corr.data
   }
  )

  gtl.diagnostic.stop();
}
```



Корреляция параметров (диаграмма рассеяния)

6.5.26 Определение дисперсии (разброса значений) - Gtl.add value var.

Функция предназначена для определения дисперсии (разброса значений амплитуд вибрации) в отфильтрованном диапазоне сигнала.

Объявление функции

`var variance = gtl.add_value_var( фильтр );`

Свойства (методы)

`variance.name = "Var";` - присвоение имени измерения дисперсии;

`variance.time = 0.5;` - интервал расчета дисперсии, сек;

`variance.avg_cnt = 4;` - количество отсчетов для усреднения;

Результат

`variance.value;` - расчетное значение дисперсии;

`variance.values;` - массив, содержащий указанное количество отсчетов для усреднения;

Пример использования

```
//фильтрация сигнала перед расчетом дисперсии
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //назначение переменной для фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
filter.order = 10; //порядок фильтра
filter.frequency = 6400; //центральная частота полосового фильтра
filter.width = 1482; //ширина полосы фильтра

//определение дисперсии значения в 1/3 октавной полосе
var variance = gtl.add_value_var(filter); //объявление переменной дисперсии

gtl.diagnostic.interval = 5; //интервал запуска функции диагностики

function diagnose() {
  gtl.log.info("Дисперсия в диапазоне", variance.value);
  gtl.diagnostic.stop();
};
```

6.5.27 Построение спектра вибрации - Gtl.add ausp.

Функция предназначена для построения спектра вибрации. Необходима при проведении спектрального анализа составляющих вибрации.

Объявление функции

```
var ausp = gtl.create_ausp(
   {
     src: gtl.analog_inputs[0],
     frequency: 1000,
     resolution: 2,
     average: 3,
```

```
        overlap: .5,
        window: gtl.spec.rectangular,
        view: gtl.spec.db
      }
);
```

Свойства (методы)

`ausp.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`ausp.frequency = 1000;` - граничная частота спектра вибрации, Гц;

`ausp.resolution = 2;` - частотное разрешение (frequency / lines), Гц/линия;

`ausp.average = 3;` - количество отсчетов для усреднения;

`ausp.overlap = 0.5;` - коэффициент перекрытия;

`ausp.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`ausp.view = gtl.spec.db;` - единица измерения амплитуды;

```
db;
unit;
rms;
```

`ausp.acq_time;` - время, необходимое для расчета спектра вибрации;

`ausp.input_data;` - массив входных значений;

`ausp.data;` - массив значений амплитуд составляющих сигнала в спектре вибрации;

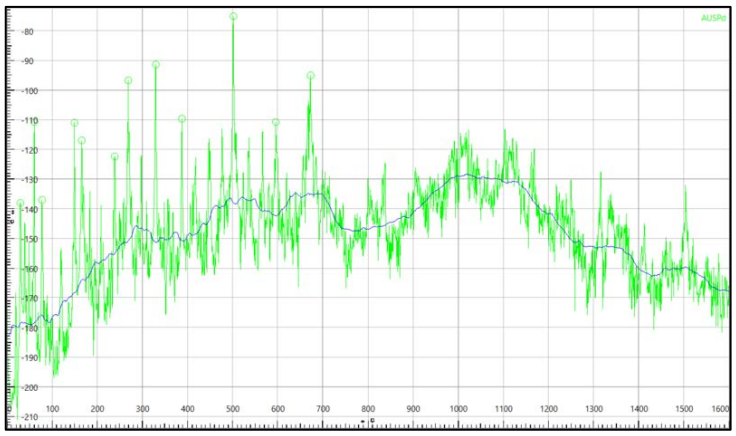
`ausp.real;` - массив действительных значений;

`ausp.imag;` - массив мнимых значений;

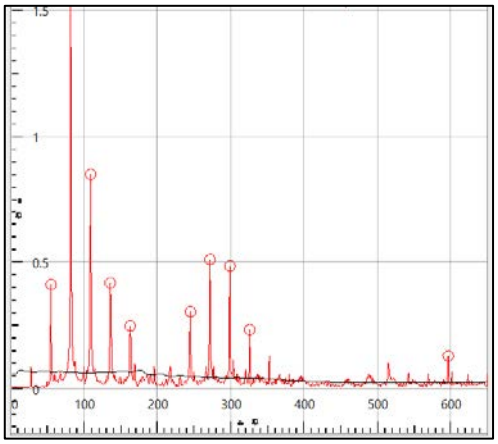
Пример использования

```
//построение спектра вибрации
var ausp = gtl.create_ausp(
   {
      src: gtl.analog_inputs[0],
      frequency: 800,
      resolution: 1,
      average: 6,
      overlap: 0.5,
      window: gtl.spec.rectangular,
```

```
        view: gtl.spec.db
    }
);

let plot = gtl.plots.add("Спектр вибраци");
gtl.diagnostic.interval = ausp.acq_time;

function diagnose() {
    plot.add(
        {
            color: 0x0000FF00,
            name: "AUSP",
            x: ausp.resolution,
            y: ausp.data
        }
    ); //рисуем спектр вибрации на plot

  gtl.diagnostic.stop();
};
```



Спектр вибрации в дБ



Спектр вибрации в абсолютных единицах

6.5.28 Построение спектра огибающей вибрации - Gtl.add spen.

Функция предназначена для построения спектра огибающей высокочастотной вибрации в отфильтрованном диапазоне. Необходима при проведении анализа колебаний мощности составляющих высокочастотной вибрации.

Объявление функции

```
var spen = gtl.create_spen(
   {
     src: gtl.analog_inputs[0],
     frequency: 1000,
     resolution: 1,
     average: 3,
     overlap: .5,
     window: gtl.spec.rectangular,
     view: gtl.spec.db
   }
);
```

Свойства (методы)

`spen.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`spen.frequency = 1000;` - граничная частота спектра огибающей, Гц;

`spen.resolution = 1;` - частотное разрешение (frequency / lines), Гц/линия;

`spen.average = 3;` - количество отсчетов для усреднения;

`spen.overlap = 0.5;` - коэффициент перекрытия;

`spen.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`spen.view = gtl.spec.db;` - единица измерения амплитуды;

```
db;
unit;
rms;
```

`spen.acq_time;` - время, необходимое для расчета спектра огибающей;

`spen.input_data;` - массив входных значений;

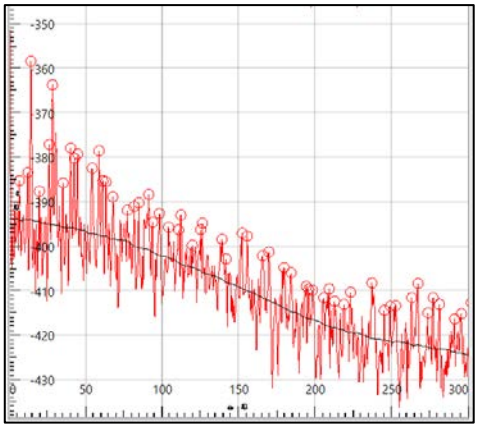
`spen.data;` - массив значений амплитуд составляющих сигнала в спектре огибающей;

`spen.real;` - массив действительных значений;

`spen.imag;` - массив мнимых значений;

Пример использования

```
//фильтрация участка сигнала для формирования спектра огибающей
var filter_spen = gtl.add_filter_iir(gtl.analog_inputs[0]]); //назначение переменной для фильтра
filter_spen.kind = gtl.filter_iir.butterworth; //тип окна
filter_spen.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
filter_spen.order = 10; //порядок фильтра
filter_spen.frequency = 6400; //центральная частота полосового фильтра
filter_spen.width = 1482; //ширина полосы фильтра

//построение спектра огибающей в узком диапазоне
var spen = gtl.create_spen(
   {
      src: filter_spen,
      frequency: 400,
      resolution: 0.25,
      average: 8,
      overlap: 0,
      window: gtl.spec.rectangular,
      view: gtl.spec.db
   }
);

let plot = gtl.plots.add("Спектр огибающей");
gtl.diagnostic.interval = spen.acq_time;

function diagnose() {
   plot.add(
      {
         color: 0x00ff0000,
         name: "SPEN",
         x: spen.resolution,
         y: spen.data
      }
   ); //рисуем спектр огибающей на plot

  gtl.diagnostic.stop();
};
```



Спектр огибающей вибрации в дБ

6.5.29 Построение спектра мощности (спектральная плотность) - Gtl.add psd.

Характеристика сигнала, которая позволяет изучать его составляющие частоты. Эта характеристика позволяет определить, какие частоты присутствуют в сигнале и с какой силой (энергией) они проявляются. Параметр измеряется в единицах мощности на единицу частоты.

Объявление функции

```
var psd = gtl.add_psd(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "psd",
     "color" : 0xff0000,
     "visible" : true,
     "freq" : 1000.0,
     "window" : gtl.spec.rectangular,
     "resolution" : 1.0,
     "average" : 1,
     "overlap" : 0,
     "units" : gtl.spec.unit
   }
);
```

Свойства (методы)

`psd.src1 = gtl.analog_inputs[0];` - источник сигнала (номер измерительного канала или переменная параметра);

`psd.name = "corr";` - присвоение имени параметру;

`psd.color = 0xff0000;` - цвет линии в формате HEX;

`psd.visible = boolean (true/false);` - отображение параметра;

`psd.freq;` - граничная частота сигнала, Гц;

`psd.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
```

`psd.resolution;` - частотное разрешение, Гц;

`psd.average;` - количество усреднений;

`psd.overlap;` - коэффициент перекрытия;

`psd.unit = gtl.spec.unit;` - единица измерения амплитуды;

```
db;
unit;
```

`psd.data[50];` - массив результатов вычисления спектральной плотности, А/Гц;

Пример использования

```
//определение спектральной плотности сигнала
//если параметры не указывать, то будут использоваться значения по-умолчанию
```

```
//так же изменять значения параметров можно в любом месте скрипта и в любое время
var psd = gtl.add_psd(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "psd",
     "color" : 0xff0000,
     "visible" : true,
     "freq" : 1000.0,
     "window" : gtl.spec.rectangular,
     "resolution" : 1.0,
     "average" : 1,
     "overlap" : 0,
     "units" : gtl.spec.unit
   }
);

gtl.diagnostic.interval = psd.acq_time;
function diagnose()
{
  gtl.log.info("psd", psd.data[50]);
  gtl.diagnostic.stop();
};
```

6.5.30 Построение порядкового спектра вибрации - Gtl.add orsp.

Функция предназначена для построения порядкового спектра вибрации. Необходима при проведении спектрального анализа составляющих вибрации объектов, работающих с переменной частотой вращения.

Объявление функции

```
var orsp = gtl.create_orsp(
   {
     src: gtl.analog_inputs[0],
     frequency: 100,
     resolution: 0.05,
     average: 1,
     overlap: 0.5,
     window: gtl.spec.rectangular,
     view: gtl.spec.db,
     harm_freq: 12,
     max_order: 5
   }
);
```

Свойства (методы)

`orsp.src = gtl.analog_inputs[0];` - источник сигнала вибрации;

`orsp.frequency = 100;` - граничная частота спектра вибрации, Гц;

`orsp.resolution = 0.05;` - частотное разрешение (frequency / lines), Гц/линия;

`orsp.average = 1;` - количество отсчетов для усреднения;

`orsp.overlap = 0.5;` - коэффициент перекрытия;

`orsp.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
cosin;
hann;
bartlett_hann;
hamming;
blackman;
blackman_harris;
flattop;
half_rect;
```

`orsp.view = gtl.spec.db;` - единица измерения амплитуды;

```
db;
unit;
rms;
```

`orsp.harm_freq = 12;` - базовая частота (чатота вращения), Гц;

`orsp.max_order = 5;` - максимальное количество порядков, шт;

`orsp.acq_time;` - время, необходимое для расчета порядкового спектра вибрации;

`orsp.input_data;` - массив входных значений;

`orsp.data;` - массив значений амплитуд составляющих в порядковом спектре вибрации;

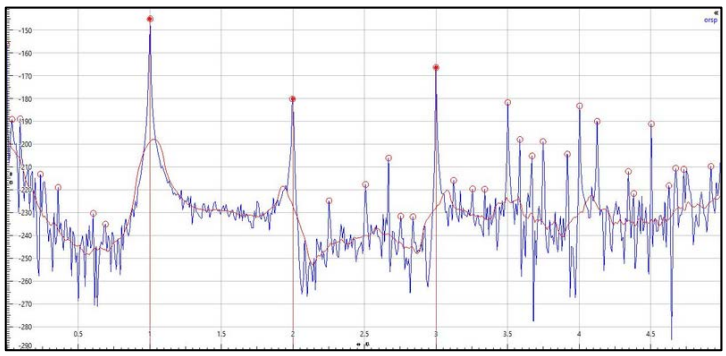
`orsp.real;` - массив действительных значений;

`orsp.imag;` - массив мнимых значений;

Пример использования

```
var freq = 4.94; //задаем базовую частоту (частота порядка)

//построение порядкового спектра вибрации
var orsp = gtl.create_orsp(
   {
      src: gtl.analog_inputs[0], //источник сигнала
      frequency: 100, //граничная частота
      resolution: 0.05, //частотное разрешение
      average: 4, //количество усреднений
      overlap: .5, //коэффициент перекрытия
      window: gtl.spec.rectangular, //тип окна
      view: gtl.spec.db, //единицы отображения дБ
      harm_freq: freq, //базовая частота (частота порядка)
      max_order: 5 //максимальное количество порядков
   }
); //создаем объект порядкового спектра

let plot1 = gtl.plots.add("Порядковый спектр"); //создаем координатную плоскость для отображения спектра
gtl.diagnostic.interval = orsp.acq_time; //указываем интервал запуска диагностики
```

```
function diagnose() {
    plot1.add(
        {
            color: 0x0000ff,
            name: "ORSP",
            x: orsp.resolution,
            y: orsp.data,
            spec_tools: {
                base: {
                    factor: 10,
                    visible: true,
                    color: 0xffff00
                },
                peaks: {
                    color: 0xff0000,
                    visible: true,
                    level: 10
                },
                harms: {
                    tolerance: .1
                }
            }
        }
    ); //рисуем порядковый спектр на plot

  gtl.diagnostic.stop();
};
```



Порядковый спектр

6.5.31 Построение кепстра - Gtl.add ceps.

Функция предназначена для построения кепстра вибрации. Необходима при проведении дополнительных видов анализа, составляющих вибрации.

Объявление функции

```
var ceps = gtl.add_ceps(
    {
      "src": gtl.analog_inputs[0],
      "name": "ceps",
      "color": 0xff0000,
```

```
    "visible": false,
    "time": 1.0,
    "freq": 1000.0,
    "window": gtl.spec.rectangular
  }
);
```

Если параметры функции не указывать, то будут использованы значения по умолчанию.

Свойства (методы)

`ceps.src = gtl.analog_inputs[0];` - источник сигнала (номер измерительного канала);

`ceps.name = "ceps";` - присвоение имени кепстра;

`ceps.color = 0xff0000;` - цвет линии кепстра в формате HEX;

`ceps.visible = true;` - видимость кепстра на графике (true, false);

`ceps.time = 1.0;` - время выборки, сек.;

`ceps.freq = 1000;` - граничная частота кепстра, Гц;

`ceps.window = gtl.spec.rectangular;` - тип окна;

```
rectangular;
```

Пример использования

```
//построение кепстра
var ceps = gtl.add_ceps(
  {
    "src" : gtl.analog_inputs[0],
    "name" : "ceps",
    "color" : 0xff0000,
    "visible" : false,
    "time" : 1.0,
    "freq" : 1000.0,
    "window" : gtl.spec.rectangular
  }
);

//переопределять параметры можно в любом месте и в любое время
ceps.color = 0x0000ff;
ceps.visible = true;
ceps.name = "Кепстр сигнала вибрации";

gtl.diagnostic.interval = ceps.time;

function diagnose()
{
  gtl.log.info("Кепстр", ceps.data[0]);
  gtl.diagnostic.stop();
};
```

6.5.32 Определение альтернативного коэффициента эксцесса (по огибающей сигнала) - Gtl.get kurt value.

Функция предназначена для определения коэффициента эксцесса (куртозис) по огибающей сигнала вибрации. Необходима для исследования колебаний мощности высокочастотных составляющих сигнала вибрации.

Объявление функции

```
var kurt_alt = gtl.get_kurt_value( массив значений линии огибающей );
```

Свойства (методы)

Отсутствуют

Пример использования

```
//фильтрация участка сигнала для формирования спектра огибающей
var filter_spen = gtl.add_filter_iir(gtl.analog_inputs[0]]); //назначение переменной для фильтра
filter_spen.kind = gtl.filter_iir.butterworth; //тип окна
filter_spen.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
filter_spen.order = 10; //порядок фильтра
filter_spen.frequency = 6400; //центральная частота полосового фильтра
filter_spen.width = 1482; //ширина полосы фильтра

//построение спектра огибающей в узком диапазоне
var spen = gtl.add_spen(filter_spen); //назначение переменной спектра огибающей
spen.name = "SPEN"; //присвоение имени спектра огибающей
spen.color = 0x00ff0000; //цвет линии спектра огибающей
spen.frequency = 400; //граничная частота спектра огибающей
spen.lines = 800; //разрешение спектра огибающей (количество линий)
spen.average = 8; //количество усреднений
spen.unit = gtl.spec.db; //отображение в дБ
spen.window = gtl.spec.hann; //окно
spen.smoothing_factor = 100; //коэффициент сглаживания средней линии спектра
spen.smoothed_line_color = 0xff004dff; //цвет средней линии
spen.peak_level = 10; //порог обнаружения гармоник
spen.harm_tolerance = 1; //диапазон поиска гармоник +/-

var kurt_alt = gtl.get_kurt_value(spen.env);

gtl.diagnostic.interval = spen.time * spen.average; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Коэффициент эксцесса огибающей сигнала", kurt_alt.value);
};
```

6.5.33 Вылавливание превышений уровня сигнала - Gtl.add thresh data.

Функция предназначена для фиксации превышений максимального значения параметра в указанном временном интервале наблюдения.

Объявление функции

```
var thresh = gtl.add_thresh_data(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "thresh",
     "color" : 0x00ff00,
     "level" : 0.5,
     "time" : .1
   }
);
```

Свойства (методы)

`thresh.src = gtl.analog_inputs[0];` - источник сигнала (номер измерительного канала или переменная параметра);

`thresh.name = "thresh";` - присвоение имени параметра;

`thresh.color = 0x00ff00;` - цвет линии в формате HEX;

`thresh.level = 0.5;` - пороговый уровень для параметра, при превышении которого фиксируется значение;

`thresh.time = 1.0;` - время выборки, сек.;

`thresh.value_at;` - массив значений параметра;

`thresh.back_value;` - доступ к последнему значению массива;

`thresh.size;` - длина массива значений;

Пример использования

```
var rms = gtl.add_rms_data(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "rms",
     "color" : 0xff0000,
     "time" : .1
   }
);

var max = gtl.add_max_data(
   {
     "src" : rms,
     "name" : "max",
     "color" : 0x00ff00,
     "time" : 1
   }
);

var thresh = gtl.add_thresh_data(
   {
     "src" : gtl.analog_inputs[0],
     "name" : "thresh",
```

```
      "color" : 0x00ff00,
      "level" : 0.5,
      "time" : .1
    }
);

gtl.diagnostic.interval = 1;

function diagnose()
{
  gtl.diagnostic.stop();
  gtl.log.info("size", rms.size);
  gtl.log.info("max", thresh.back_value);
};
```

6.5.34 Загрузка данных из csv файла - Gtl.io.from csv.

Функция предназначена для загрузки массивов данных из внешнего файла формата .csv

Объявление функции

```
var arr = gtl.io.from_csv(
  {
    path: "E:/data/gtl/1.csv",
    col: 0,
    sep: ";"
  }
);
```

Свойства (методы)

`arr.path = "E:/data/gtl/1.csv";` - путь к файлу с данными;

`arr.col = 0;` - ;

`arr.sep = ";";` - разделитель данных;

```
";" - точка с запятой;
"\t" - табуляция (красная строка);
```

Пример использования

```
var arr = gtl.io.from_csv(
  {
    path: "E:/data/gtl/1.csv",
    col: 0,
    sep: ";"
  }
);

gtl.diagnostic.interval = 1;
```

```
function diagnose()
{
  for (let i = 0; i <= arr.length -1; i++)
    { gtl.log.info(i + "-й элемент массива: ", arr[i]); };

  gtl.log.info("Произведение данных из массива: ", arr[1] * arr[2]);
  gtl.diagnostic.stop();

};
```

6.5.35 Экспорт данных в csv файл - Gtl.io.to csv.

Функция предназначена для экспорта данных массива в файл формата .csv

Объявление функции

```
var res = gtl.io.to_csv(
  {
    path: "E:/data/gtl/5.csv",
    values: arr,
    sep: "\t"
  }
);
```

Свойства (методы)

`res.path = "E:/data/gtl/1.csv";` - путь сохранения файла с данными;

`res.values = arr;` - массив данных;

`arr.sep = "\t";` - разделитель данных;

```
";" - точка с запятой;
"\t" - табуляция (красная строка);
```

Пример использования

```
var my_arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]; //массив данных
var res = gtl.io.to_csv(
  {
    path: "E:/data/gtl/5.csv",
    values: my_arr,
    sep: "\t"
  }
);

gtl.diagnostic.interval = 1;

function diagnose()
{
  gtl.diagnostic.stop();
```

```
};
```

6.6 Библиотека диагностических функций GTLd:

6.6.1 Общая информация.

Все функции данного раздела (кроме интервала запуска диагностики) используются внутри основной функции диагностики (в теле функции).

6.6.2 Интервал запуска функции диагностики - Gtl.diagnostic.interval.

Функция предназначена для периодического запуска основной функции диагностики с заданным интервалом. Вызывается перед основной функцией диагностики.

Синтаксис

`gtl.diagnostic.interval = 60;` - интервал запуска функции диагностики, сек;

Пример использования

```
//фильтрация сигнала перед определением частоты вращения из сигнала
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 10; //граничная частота фильтра

//определение частоты вращения
var freq = gtl.add_value_freq(filter); //объявление переменной частоты вращения
freq.time = 1; //время выборки
freq.avg_cnt = 6; //количество усреднений

gtl.diagnostic.interval = freq.time * freq.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Частота вращения", freq.value);
}
```

6.6.3 Определение частоты вращения по источнику данных и автоматическое уточнение - Function diagnose.

Функция предназначена для реализации логики, последовательности и процедур постановки автоматического диагноза (анализа сигналов вибрации) с выдачей определенного результата. Весь логический блок описывается в теле самой функции.

Синтаксис

`function diagnose() { ...тело функции... };`

Пример использования

```
//фильтрация сигнала перед определением частоты вращения из сигнала
var filter = gtl.add_filter_iir(gtl.analog_inputs[1]); //объявление переменной фильтра
```

```
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 10; //граничная частота фильтра

//определение частоты вращения
var freq = gtl.add_value_freq(filter); //объявление переменной частоты вращения
freq.time = 1; //время выборки
freq.avg_cnt = 6; //количество усреднений

gtl.diagnostic.interval = freq.time * freq.avg_cnt; //интервал запуска функции диагностики

function diagnose() {
gtl.log.info("Частота вращения", freq.value);
};
```

6.6.4 Нестабильность частоты вращения - Gtld instability.

Является вспомогательной функцией и используется для определения нестабильности частоты вращения, определенной из тахосигнала. Данное верхнее граничное значение необходимо для контроля разброса определяемых частот относительно расчетных. При превышении предельного значения рекомендуется останавливать процесс диагностирования до стабилизации оборотов. В качестве входного сигнала подается объект, содержащий массив данных по частоте вращения. Для выражения результата в % необходимо умножить значение на 100.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __inst = freqInstab( объект частоты вращения );
```

Содержание функции

```
function freqIstab(src) {
    let __max = Math.max(...src.values);
    let __min = Math.min(...src.values);
    let __instab = (__max - __min) / src.value;
    return __instab;
}; //нестабильность частоты вращения
```

6.6.5 Определение функциональных частот объектов диагностики - Gtld functions.

Функциональные (диагностические) частоты используются для построения портретов дефектов и проведения спектральной диагностики объектов. В качестве аргумента подается значение частоты вращения. Для упрощения записи функций используются вспомогательные расчетные коэффициенты:

```
var rb_k1 = 0.5 * (1 - (rb_roller / rb_cage) * Math.cos(rb_angle));
var rb_k2 = 0.5 * (1 + (rb_roller / rb_cage) * Math.cos(rb_angle));
var bs_k1 = 0.5 * (1 - (bs_roller / bs_cage) * Math.cos(bs_angle));
```

```
var bs_k2 = 0.5 * (1 + (bs_roller / bs_cage) * Math.cos(bs_angle));
```

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __x = FUNCTION( частота вращения );
```

Содержание функций

```
function FREQ(freq) { return freq }; //частота вращения

function FTF(freq) { return rb_k1 * freq }; //частота вращения сепаратора (FTF)
function BPFO(freq) { return rb_k1 * freq * rb_rollerCount }; //частота перекатывания тел качения по наружному кольцу (BPFO)
function BPFI(freq) { return rb_k2 * freq * rb_rollerCount }; //частота перекатывания тел качения по внутреннему кольцу (BPFI)
function BSF(freq) { return 2 * freq * (rb_cage / rb_roller) * rb_k1 * rb_k2 }; //частота вращения (контакта) тел качения (BSF)

function BSFTF(freq) { return bs_k1 * freq }; //частота вращения сепаратора (перемещения тел качения)
function BSNUT(freq) { return bs_k1 * freq * bs_rollerCount }; //частота перекатывания тел качения по гайке
function BSSCR(freq) { return bs_k2 * freq * bs_rollerCount }; //частота перекатывания тел качения по винту
function BSBAL(freq) { return 2 * freq * (bs_cage / bs_roller) * bs_k1 * bs_k2 }; //частота вращения (контакта) тел качения

function GTFZ(freq) { return freq * gtZ1 }; //зубцовая частота
function GTF2(freq) { return freq * (gtZ1 / gtZ2) }; //частота вращения второго вала редуктора

function BDF2(freq) { return freq * (bdD1 / bdD2) }; //частота вращения ведомого шкива
function BDFB(freq) { return freq * (Math.PI * bdD1 / bdL) }; //частота вращения ремня

function CBFZ(freq) { return freq * cbdZ1 }; //зубцовая частота
function CBDF2(freq) { return freq * (cbdZ1 / cbdZ2) }; //частота вращения ведомого шкива
function CBDFB(freq) { return freq * (cbdZ1 / cbdZ3) }; //частота вращения ремня

function PMFBLD(freq) { return freq * pmBlades }; //лопастная частота

function PGF2(freq) { return (0.5 * freq * pgZ1) / (pgZ1 + pgZ2) }; //частота вращения выходного вала планетарной передачи
function PGFSAT(freq) { return (0.5 * freq) * (pgZ1 / pgZ2) * ((pgZ1 + 2 * pgZ2) / (pgZ1 + pgZ2)) }; //частота вращения саттелита
function PGFZ(freq) { return pgZ2 * PGFSAT(freq) }; //зубцовая частота

function TRFBLD(freq) { return freq * trBlades }; //лопастная частота
```

6.6.6 Минимально необходимая частота вращения - Gtld freqness.

Является вспомогательной функцией и используется для расчета минимально необходимой частоты вращения при диагностировании подшипников качения. Данная нижняя граничная

частота необходима для создания и контроля режима "обкатывания" дорожки качения наружного кольца подшипника. Для этого необходимо компенсировать силу тяжести, действующую на тела качения, и радиальный зазор в подшипнике центробежной силой. Данная функция используем расчетные параметры подшипника качения (диаметр внутреннего кольца, диаметр тела качения и вспомогательный коэффициент для расчета функциональных частот).

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __fness = freqNess();
```

Содержание функции

```
function freqNess() {
    let R = (d_inner / 2) + (d_roller / 2);
    let __fness = (Math.sqrt(9.81 / (4 * (Math.PI ** 2) * R / 1000))) / rb_k1;
    return __fness;
};
```

6.6.7 Расчетная центральная частота полосового фильтра - Gtld filter frequency.

Функция расчета центральной частоты полосового фильтра для формирования спектра огибающей высокочастотной вибрации. В качестве аргумента подается значение частоты вращения.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __fltr = bpfFreq( частота вращения );
```

Содержание функции

```
function bpfFreq(freq) {
    let __filter = 1850 * Math.sqrt(freq);
    //let __filter = 6013.41 * Math.log(0.266935 * freq + 1.1201);
    return __filter;
};
```

6.6.8 Ширина полосового фильтра - Gtld filter width.

Функция для расчета ширины полосового фильтра с учетом его октавности. В качестве аргументов подается количество долей октавного фильтра и центральная частота полосового фильтра.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __wdt = bpfWidth( количество долей октавы, центральная частота );
```

Содержание функции

```
function bpfWidth(n, filter) {
    let kf = (2 ** (1 / n) - 1) / ((2 ** (1 / n)) ** 0.5);
    let __wdt = kf * filter;
    return __wdt;
};
```

6.6.9 Расчетные параметры спектра - Gtld spec params.

Функция для расчета рекомендуемых параметров спектра вибрации таких как граничная частота (ширина), частотное разрешение и количество линий в зависимости от исследуемого объекта диагностики. В качестве аргумента подается значение частоты вращения. Функция возвращает объект с указанными параметрами. При этом часть рассчитанных параметров внутри функции приводится к стандартизованным значениям.

Синтаксис

Стандартная конструкция выглядит таким образом:

`let __prms = specParams( частота вращения );`

Результат

`__prms.filter.frequency;` - центральная частота полосового фильтра, Гц;

`__prms.filter.width;` - ширина полосы полосового фильтра, Гц;

`__prms.spec.frequency;` - граничная частота спектра (ширина), Гц;

`__prms.spec.lines;` - количество линий спектра, шт;

`__prms.spec.resolution;` - частотное разрешение спектра;

`__prms.spec.tolerance;` - коридор обнаружения гармоники;

Содержание функции

```
function specParams(freq) {
    let __fltr = {}; //объект расчетных параметров полосового фильтра
    let __spec = {}; //объект расчетных параметров спектра
    let __tol = 0; //коридор обнаружения гармоники
    let __frq = 200; //граничная частота спектра
    let __lines = 400; //количество линий спектра

    let __filter = bpfFreq(freq); //центральная частота полосового фильтра
    let __width = bpfWidth(3, __filter); //ширина полосового фильтра для спектра огибающей

    function bpfFreq(rps) {
        let __filter = 1850 * Math.sqrt(rps);
        //let __filter = 6013.41 * Math.log(0.266935 * rps + 1.1201);
        return __filter;
    };

    function bpfWidth(n, filter) {
        let kf = (2 ** (1 / n) - 1) / ((2 ** (1 / n)) ** 0.5); //коэффициент для полосового фильтра
```

```
    let __wdt = kf * filter;
    return __wdt;
};

function getStandart(value) {
    let arr = [50, 100, 200, 400, 800, 1600, 3200, 6400, 12800];
    let __res = 0;
    arr.some((elem, i) => { if (value <= elem) { __res = arr[i]; return __res; }; });
    return __res;
};

switch (options.objectType) {
    case 0: //объект не выбран
        break;
    case 1: //подшипник скольжения
        __tol = 0;
        __frq = 20 * freq;
        __lines = __frq / (freq / 8);
        break;
    case 2: //подшипник качения
        __tol = (2 * FTF(freq)) / (5 * BPFO(freq));
        __frq = getStandart(5 * BPFI(freq) + 4 * freq);
        __lines = getStandart(__frq / (freq / 8));
        break;
    case 3: //ШВП
        __tol = (2 * BSFTF(freq)) / (5 * BSNUT(freq));
        __frq = getStandart(5 * BSSCR(freq) + 4 * freq);
        __lines = getStandart(__frq / (freq / 8));
        break;
    case 4: //редуктор
        __tol = (2 * freq) / (5 * GTFZ(freq));
        __frq = getStandart(3 * GTFZ(freq) + 4 * freq);
        __lines = getStandart(__frq / (freq / 8));
        break;
    case 5: //ременная передача
        __tol = 0;
        __frq = getStandart(400);
        __lines = getStandart(__frq / (BDFB(freq) / 4));
        break;
    case 6: //зубчатый ремень
        __tol = 0;
        __frq = getStandart(400);
        __lines = getStandart(__frq / (CBDFB(freq) / 4));
        break;
    case 7: //помпа (насос)
        __tol = 0;
        __frq = getStandart(3 * PMFBLD(freq) + 4 * freq);
        __lines = getStandart(__frq / (freq / 8));
        break;
    case 8: //планетарый редуктор
        __tol = (2 * PGF2(freq)) / (5 * PGFZ(freq));
        __frq = getStandart(3 * PGFZ(freq) + 4 * freq);
        __lines = getStandart(__frq / (PGF2(freq) / 8));
```

```
            break;
        case 9: //турбина
            __tol = 0;
            __frq = getStandart(3 * TRFBLD(freq) + 4 * freq);
            __lines = getStandart(__frq / (freq / 8));
            break;
        case 10: //электродвигатель
            __tol = 0;
            __frq = getStandart(400);
            __lines = getStandart(__frq / (freq / 8));
            break;
    };

    __fltr["frequency"] = __filter;
    __fltr["width"] = __width;
    __spec["frequency"] = __frq;
    __spec["lines"] = __lines;
    __spec["resolution"] = __frq / __lines;
    __spec["tolerance"] = __tol;

    return {
        filter: __fltr,
        spec: __spec
    };
}; //рассчетные параметры спектра вибрации
```

6.6.10 Интегральная площадь спектра вибрации - Gtld square.

Функция для расчета площадей спектра вибрации или его части. Необходима для количественной оценки работы (мощности) колебательных сил, проведения мониторинга изменений параметра и прогнозирования. В качестве аргументов подается объект, содержащий массив данных и границы расчета площади. В результате работы функции возвращается объект с расчетными площадями составляющих спектра.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __sqrs = specSquare( объект спектра, начало отсчета, конец отсчета );
```

Результат

`__sqrs.base` - площадь спектра под базовой линией;

`__sqrs.spec` - общая площадь спектра;

`__sqrs.harm` - площадь спектра над базовой линией (случайных и гармонических составляющих);

`__sqrs.peak` - площадь гармонических составляющих спектра;

Содержание функции

```
function specSquare(spec, L, R) {
    let __base = spec.base; //массив значений средней линии
    let __data = spec.data; //массив значений амплитуд
```

```
    let __lines = spec.data.length; //количество линий спектра
    let __res = spec.resolution; //частотное разрешения спектра
    let __start = 0; //стартовый индекс в массиве
    let __end = __lines; //конечный индекс в массиве
    let s0 = 0; //площадь под базовой линией
    let s1 = 0; //площадь всего спектра
    let s2 = 0; //площадь над базовой линией
    let s3 = 0; //площадь обнаруженных гармоник
    if (L != undefined) { __start = Math.round(L / __res) };
    if (R != undefined) { __end = Math.round(R / __res) };

    for (let i = __start; i <= __end - 1; i++) {
        s0 += __base[i] * __res;
        s1 += __data[i] * __res;
        let __delta = __data[i] - __base[i];
        if (__delta >= 0) { s2 += __delta * __res };
        if (__delta >= spec.peak_level) { s3 += __delta * __res };
    };

    return {
        base: s0,
        spec: s1,
        harm: s2,
        peak: s3
    };
}; //определение площадей спектра
```

6.6.11 Перевод значений массива в дБ - Gtld todb.

Функция предназначена для преобразования массива абсолютных значений в массив относительных единиц (дБ). В качестве аргументов подается массив данных и указывается тип преобразования (соответствующий базовому уровню).

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __dB = function todB( массив данных, тип преобразования );
```

```
0 - преобразование виброускорения в дБ (опорное значение 1e-6);
1 - преобразование виброскорости в дБ (опорное значение 1e-9);
2 - преобразование виброперемещения в дБ (опорное значение 1e-12);
```

Содержание функции

```
function todB(arr, type) {
    let __limit = 3e-4; //опорное значение
    if (type != undefined) {
        switch (type) {
            case 0: __limit = 1e-6; break;
            case 1: __limit = 1e-9; break;
            case 2: __limit = 1e-12; break;
```

```
            default:
                break;
        };
    };

    let __result = arr.map((item) => (item = 20 * Math.log10(item / __limit)));
    return __result;
}; //перевод значений массива в дБ
```

6.6.12 Глубина модуляции гармоники - Gtld mod factor.

Функция для расчета глубины модуляции гармонической составляющей сигнала в спектре огибающей высокочастотной вибрации. Данный метод предназначен для определения относительного параметра степени развития дефекта.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
var __mod = modFactor(
    {
      spec: spen, //спектр огибающей (объект)
      fltr: filter_spen, //полосовой фильтр для формирования спектра огибающей
      ampl: 110, //амплитудное значение гармонической составляющей
      base: 98 //уровень случайной составляющей (средней линии под гармоникой)
    }
);
```

Содержание функции

```
function modFactor(options) {
    let __dl = (options.ampl - options.base); //разность уровней гармонической и случайной составляющей вибрации
    let __df = options.spec.resolution / options.fltr.width; //отношение частотного разрешения спектра к ширине полосы фильтра
    let __mod = Math.sqrt((10 ** (__dl / 10) - 1) * __df);
    return __mod;
}; //определение глубины модуляции ВЧ составляющих
```

6.6.13 Условная глубина модуляции гармоники - Gtld deep factor.

Функция для расчета условной глубины модуляции гармонической составляющей сигнала в спектре. Данный метод предназначен для определения относительного параметра степени выраженности дефекта.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __deep = deepFactor( амплитуда, средний уровень );
```

Содержание функции

```
function deepFactor(ampl, base) {
    let __deep = (ampl - base) / (ampl + base) * 100;
    return __deep;
};
```

6.6.14 Амплитудный коэффициент - Gtld ampl factor.

Функция для расчета амплитудного коэффициента гармонической составляющей в спектре огибающей. Необходима для относительной оценки изменения колебательных сил, проведения мониторинга изменений параметра (аналогично методу ПФ) и прогнозирования.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
let __ampl = amplFactor( амплитуда, средний уровень );
```

Содержание функции

```
function amplFactor(ampl, base) {
    let __crest = ampl / base;
    return __crest;
};
```

6.6.15 Набор стандартных измерений параметров вибрации - Gtld std measures.

Функция предназначена для реализации набора стандартных измерений параметров вибрации. Функция возвращает набор объектов с результатами измерений, по ключу value которых можно получить значение рассчитанного параметра.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
var __mes = getStdMeasures(
    {
      src: gtl.analog_inputs[0], //источник сигнала вибрации
      time: 0.25, //интервал расчета параметров
      avg: 8 //количество отсчетов для усреднения
    }
);
```

Результат

`__mes.rms_A2_3000.value` - СКЗ виброускорения в диапазоне 2-3000 Гц;

`__mes.rms_A2_10000.value` - СКЗ виброускорения в диапазоне 2-10000 Гц;

`__mes.rms_A100_10000.value` - СКЗ виброускорения в диапазоне 100-10000 Гц;

`__mes.ampl_A100_10000.value` - амплитуда виброускорения в диапазоне 100-10000 Гц;

`__mes.ampl_full.value` - амплитуда виброускорения во всем диапазоне измерения;

__mes.rms_V2_1000.value - СКЗ виброскорости в диапазоне 2-1000 Гц;

__mes.rms_V10_1000.value - СКЗ виброскорости в диапазоне 10-1000 Гц;

__mes.rms_S2_200.value - СКЗ виброперемещения в диапазоне 2-200 Гц;

__mes.ptp_S2_200.value - размах виброперемещения в диапазоне 2-200 Гц;

__mes.kurt_full.value - эксцесс во всем диапазоне измерения;

__mes.kurt_2_5k.value - эксцесс в полосе 2.5-5 кГц;

__mes.kurt_5_10k.value - эксцесс в полосе 5-10 кГц;

__mes.kurt_10_20k.value - эксцесс в полосе 10-20 кГц;

__mes.kurt_30_40k.value - эксцесс в полосе 30-40 кГц;

__mes.kurt_40_80k.value - эксцесс в полосе 40-80 кГц;

Содержание функции

```
function getStdMeasures(options) {
    let __source = options.src;
    let __time = 0.1;
    let __avg = 5;

    if (options.time != undefined) { __time = options.time };
    if (options.avg != undefined) { __avg = options.avg };

    function getFilter(L, R) {
        let __filter = gtl.add_filter_iir(__source);
        __filter.kind = gtl.filter_iir.butterworth;
        __filter.order = 10;

        switch (L) {
            case "lowpass":
                __filter.type = gtl.filter_iir.lowpass;
                __filter.frequency = R;
                break;

            case "highpass":
                __filter.type = gtl.filter_iir.highpass;
                __filter.frequency = R;
                break;

            default:
                __filter.type = gtl.filter_iir.bandpass;
                __filter.frequency = (R - L) / 2 + L;
                __filter.width = R - L;
                break;
        };

        return __filter;
    }; //формирование фильтра

    function getIntg(src, taps, scale) {
        let __intg = gtl.add_intg(src);
```

```
    __intg.taps = taps;
    __intg.scale = scale;

    return __intg;
}; //интегрирование сигнала

function getRMS(src) {
    let __rms = gtl.add_value_rms(src);
    __rms.time = __time;
    __rms.avg_cnt = __avg;

    return __rms;
}; //получение СКЗ

function getAmpl(src) {
    let __ampl = gtl.add_value_ampl(src);
    __ampl.time = __time;
    __ampl.avg_cnt = __avg;

    return __ampl;
}; //получение амплитуды

function getPtP(src) {
    let __ptp = gtl.add_value_peak_to_peak(src);
    __ptp.time = __time;
    __ptp.avg_cnt = __avg;

    return __ptp;
}; //получение размаха

function getKurt(src) {
    let __kurt = gtl.add_value_kurt(src);
    __kurt.time = __time;
    __kurt.avg_cnt = __avg;

    return __kurt;
}; //получение эксцесса

//[Набор фильтров]
let __filter_2_200 = getFilter(2, 200);
let __filter_2_1000 = getFilter(2, 1000);
let __filter_10_1000 = getFilter(10, 1000);
let __filter_2_3000 = getFilter(2, 3000);
let __filter_2_10000 = getFilter(2, 10000);
let __filter_100_10000 = getFilter(100, 10000);
let __filter_2_5k = getFilter(2500, 5000);
let __filter_5_10k = getFilter(5000, 10000);
let __filter_10_20k = getFilter(10000, 20000);
let __filter_30_40k = getFilter(30000, 40000);
let __filter_40_80k = getFilter(40000, 80000);

//[Набор интеграторов]
let __pre_int1 = getIntg(__filter_2_1000, 1, 1000);
```

```
    let __pre_int2 = getIntg(__filter_10_1000, 1, 1000);
    let __pre_int3 = getIntg(__filter_2_200, 2, 1);

    //[Расчет набора параметров]
    let __rms_A2_3000 = getRMS(__filter_2_3000); //СКЗ виброускорения в диапазоне 2-3000 Гц
    let __rms_A2_10000 = getRMS(__filter_2_10000); //СКЗ виброускорения в диапазоне 2-10000 Гц
    let __rms_A100_10000 = getRMS(__filter_100_10000); //СКЗ виброускорения в диапазоне 100-10000 Гц

    let __ampl_A100_10000 = getAmpl(__filter_100_10000); //амплитуда виброускорения в диапазоне 100-10000 Гц
    let __ampl_full = getAmpl(__source); //амплитуда виброускорения во всем диапазоне измерения

    let __rms_V2_1000 = getRMS(__pre_int1); //СКЗ виброскорости в диапазоне 2-1000 Гц
    let __rms_V10_1000 = getRMS(__pre_int2); //СКЗ виброскорости в диапазоне 10-1000 Гц
    let __rms_S2_200 = getRMS(__pre_int3); //СКЗ виброперемещения в диапазоне 2-200 Гц

    let __ptp_S2_200 = getPtP(__pre_int3); //размах виброперемещения в диапазоне 2-200 Гц

    let __kurt_full = getKurt(__source); //эксцесс во всем диапазоне измерения
    let __kurt_2_5k = getKurt(__filter_2_5k); //эксцесс в полосе 2.5-5 кГц
    let __kurt_5_10k = getKurt(__filter_5_10k); //эксцесс в полосе 5-10 кГц
    let __kurt_10_20k = getKurt(__filter_10_20k); //эксцесс в полосе 10-20 кГц
    let __kurt_30_40k = getKurt(__filter_30_40k); //эксцесс в полосе 30-40 кГц
    let __kurt_40_80k = getKurt(__filter_40_80k); //эксцесс в полосе 40-80 кГц

    let __result = {
        rms_A2_3000: __rms_A2_3000,
        rms_A2_10000: __rms_A2_10000,
        rms_A100_10000: __rms_A100_10000,
        ampl_A100_10000: __ampl_A100_10000,
        ampl_full: __ampl_full,
        rms_V2_1000: __rms_V2_1000,
        rms_V10_1000: __rms_V10_1000,
        rms_S2_200: __rms_S2_200,
        ptp_S2_200: __ptp_S2_200,
        kurt_full: __kurt_full,
        kurt_2_5k: __kurt_2_5k,
        kurt_5_10k: __kurt_5_10k,
        kurt_10_20k: __kurt_10_20k,
        kurt_30_40k: __kurt_30_40k,
        kurt_40_80k: __kurt_40_80k,
    };

    return __result;
}; //измерение стандартных параметров
```

6.6.16 Построение компонентов и частотных линий на спектре - Gtl.create tools.

Функция предназначена для упрощенного построения компонентов (маркеров гармоник, базовой линии) и частотных линий на координатной плоскости спектра вибрации.

Объявление функции

Для создания компонентов используется запись:

```
let __tools = createTools(
    {
        spec: spen, //спектр для построения модели (объект)
        set: __set, //источник данных для построения частотных линий
        tol: 1 //коридор обнаружения гармоники на портрете, %
    }
);
```

Свойства (методы)

Аргумент по ключу "set" принимает объект определенной структуры, содержащий массив данных, необходимых для построения частотных линий. Если объект не передается, то частотные линии не строятся.

```
let __set = {
    "Частота вращения сепаратора": [0xFF2400, 2.05, 4, 6, 0],
    "Частота вращения": [0x89AC76, 4.94, 4, 6, 0],
    "Частота контакта тел качения": [0xFFA000, 14.25, 5, 10, 2.05],
    "Частота перекатывания тел качения по наружному кольцу": [0x42AAFF, 30.81, 6, 13, 2.05],
    "Частота перекатывания тел качения по внутреннему кольцу": [0x34C924, 43.29, 6, 8, 4.94],
}; //набор подшипниковых частот
```

где в массиве под индексом:
0 - цвет линии в формате HEX;
1 - основная частота, Гц;
2 - количество гармоник, шт;
3 - пороговый уровень сильного дефекта;
4 - модулирующая частота, Гц.

Содержание функции

```
function createTools(options) {
    let __spec = options.spec; //источник данных спектра
    let __set = {}; //источник данных для построения частотных линий
    if (options.set != undefined) { __set = options.set };

    let __tools = gtl.create_spec_tools(
        {
            data: __spec.data, //массив значений амплитуд спектра
            df: __spec.resolution, //частотное разрешение спектра
            base: {
                factor: 100, //коэффициент сглаживания базовой линии
                visible: true, //отображение базовой линии
                color: 0xFFFF00 //цвет базовой линии в формате HEX
            },
```

```
            peaks: {
                color: __spec.color, //цвет маркеров обнаруженных гармоник
                visible: false, //отображение маркеров
                level: __spec.peak_level //уровень обнаружения гармоник в спектре
            },
            harms: {
                tolerance: options.tol //коридор обнаружения гармоник в спектре
            }
        }
    );

    let __rows = Object.keys(__set); //массив ключей объекта (наименование частот)
    for (let i = 0; i < __rows.length; i++) {
        let __name = __rows[i]; //название гармонического ряда
        let __arr = __set[__name]; //массив значений
        let __color = __arr[0]; //цвет ряда в формате HEX
        let __freq = __arr[1]; //расчетная частота
        let __count = __arr[2]; //количество гармоник
        let __lvl = __arr[3]; //пороговый уровень сильного дефекта
        let __mod = __arr[4]; //модулирующая частота

        let __row = __tools.harms.add(
            {
                frequency: __freq, //функциональная частота
                count: __count, //количество гармоник
                color: __color, //цвет линий
                weight: 2, //толщина линий
                visible: false //отображение линий
            }
        );
        __row.name = __name;

        if (__mod != undefined) {
            __row.modulate(
                {
                    frequency: __mod, //частота амплитудной модуляции
                    count: 2, //количество боковых составляющих слева и справа
                    color: __color, //цвет линий в формате HEX
                    weight: 0.5 //толщина линий
                }
            );
        };
    };

    return __tools;
}; //Построение компонентов и частотных линий на спектре
```

6.6.17 Коэффициент корреляции дискретных сигналов (массивов) - Gtld corr.

Функция предназначена для расчета коэффициента корреляции дискретных сигналов (массивов). В качестве аргументов на вход подаются массивы данных. В результате возвращается расчетное значение коэффициента корреляции.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
var __corr = getCorr( массив 1, массив 2 );
```

Содержание функции

```
function getCorr(arr1, arr2) {
    let X = 0; //аргумент 1
    let Y = 0; //аргумент 2
    let Z = 0; //аргумент 3
    let __avg1 = arr1.reduce((acc, item) => (acc + item)) / arr1.length; //среднее значение массива 1
    let __avg2 = arr2.reduce((acc, item) => (acc + item)) / arr2.length; //среднее значение массива
2

    for (let i = 0; i < arr1.length; i++) {
        X += (arr1[i] - __avg1) * (arr1[i] - __avg2);
        Y += (arr1[i] - __avg1) ** 2;
        Z += (arr2[i] - __avg2) ** 2;
    };

    return X / (Math.sqrt(Y) * Math.sqrt(Z));
}; //рассчет корреляции
```

6.6.18 Функция автокорреляции дискретных сигналов (массивов) - Gtld autocorr.

Функция предназначена для проведения автокорреляционного анализа сигнала на предмет наличия периодических составляющих. В качестве основного источника подается массив данных пиковой формы сигнала. В результате работы функции возвращается объект с параметрами расчетов.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
var __autocorr = getAutoCorr(
    {
      name: "Correlation form", //имя графика функции
      src: __max, //массив данных пиковой формы сигнала
      lag: 0.5, //коэффициент смещения сигнала
      color: 0x6A5ACD, //цвет отрисовки графика в формате HEX
      time: 0.01, //цена деления оси времени, с
      canvas: canvas3 //координатная плоскость для отрисовки графика функции
    }
);
```

Результат

`__autocorr.avg` - среднее значение модуля коэффициента автокорреляции;

`__autocorr.rms` - СКЗ коэффициента автокорреляции;

__autocorr.ampl - максимальное значение коэффициента автокорреляции после 1% формы сигнала;

__autocorr.data - массив значений коэффициента автокорреляции (для построения графика);

Содержание функции

```
function getAutoCorr(options) {
    let result = {}; //результат
    let plot = []; //массив значений корреляции для графика
    let arr = options.src;
    let arr2 = arr.concat(arr); //расширяем массив данных
    let lag = options.lag;

    let T = Math.floor(arr.length * lag); //определяем количество индексов (шагов) для смещения массива
    let avg = arr.reduce((acc, item) => (acc + item)) / arr.length; //среднее значение массива
    let X = 0; //числитель функции
    let Y = arr.reduce((acc, item) => (acc + (item - avg) ** 2), 0); //рассчитываем знаменатель функции

    const arrDiff = arr.map(value => value - avg); //разница от avg
    for (let i = 0; i < T; i++) {
        X = 0;
        for (let j = 0; j < arr.length; j++) { X += arrDiff[j] * (arr2[j + i] - avg) }; //рассчитываем числитель функции
        plot.push(X / Y); //записываем значение в массив коэффициентов
    }; //смещение массива

    let plot0 = plot.slice(Math.floor(0.01 * plot.length));
    let akf_avg = plot0.reduce((acc, item) => (acc + Math.abs(item)), 0) / plot0.length; //среднее значение коэффициента
    let akf_sqr = plot0.reduce((acc, item) => (acc + item ** 2), 0); //сумма квадратов значений
    let akf_rms = Math.sqrt(akf_sqr / plot0.length); //СКЗ коэффициента
    let akf_max = Math.max(...plot0); //определяем максимальное значение коэффициента

    result["avg"] = akf_avg;
    result["rms"] = akf_rms;
    result["ampl"] = akf_max;
    result["data"] = plot;

    //отрисовка графика на plot
    if (options.canvas != undefined) {
        options.canvas.add(
            {
                color: options.color,
                name: options.name,
                x: options.time,
                y: plot
            }
        );
    };
```

```
    return result;
}; //рассчет автокорреляции
```

6.6.19 Формирование наборов признаков дефектов - Gtld getDefSet.

Функция предназначена для формирования наборов признаков дефектов для проведения спектрального анализа сигнала вибрации в зависимости от указанного объекта диагностики. Функция возвращает объекты (set и mtx), которые используются в методе Модель-Маска и для формирования матрицы нейронной сети.

Синтаксис

Стандартная конструкция выглядит таким образом:

```
var __sets = getDefSet( частота вращения );
```

Результат

`__sets.set` - набор признаков дефектов для метода Модель-Маска;

`__sets.mtx` - набор признаков дефектов для формирования матрицы нейронной сети;

Содержание функции

```
function getDefSet(freq) {
    let __mtx = {};
    let __set = {};

    //Название: [цвет, частота, кол-во гармоник, уровень сильного дефекта, тип ряда, коэфф затухания]
    switch (options.objectType) {
        case 0: //объект не выбран
            break;
        case 1: //подшипник скольжения
            __mtx = {
                "FREQ": [0x4285B4, FREQ(freq), 6]
            };

            __set = {
                // "Бой вала": [0x4285B4, FREQ(freq), 6, 20, 0, 0.7],
                // "Автоколебания вала": [0x6A5ACD, FREQ(freq) / 2, 12, 20, 1, 0.4],
                "Перекос подшипника": [0x89AC76, FREQ(freq), 6, 20, 1, 0.4],
                "Износ подшипника": [0x34C924, FREQ(freq), 6, 20, 0, 0.5],
                "Удары в подшипнике": [0x008000, FREQ(freq), 6, 20, 0, 0.07],
            };
            break;
        case 2: //подшипник качения
            __mtx = {
                "FREQ": [0x89AC76, FREQ(freq), 6],
                "BPFO": [0x42AAFF, BPFO(freq), 6],
                "BPFI": [0x008000, BPFI(freq), 6],
                "BSF": [0xFFA000, BSF(freq), 6],
                "FTF": [0xFF2400, FTF(freq), 6]
```

```
        };

        __set = {
            "Износ наружного кольца": [0x42AAFF, BPFO(freq), 6, 13, 0, 0.3],
            "Перекос наружного кольца": [0x4285B4, BPFO(freq), 4, 12, 1, 0.1],
            "Дефект наружного кольца": [0x6A5ACD, BPFO(freq), 6, 13, 0, 0.07],
            "Износ внутреннего кольца": [0x89AC76, FREQ(freq), 6, 13, 0, 0.3],
            "Перекос внутреннего кольца": [0x34C924, FREQ(freq), 6, 12, 1, 0.1],
            "Дефект внутреннего кольца": [0x008000, BPFI(freq), 6, 8, 0, 0.07],
            "Дефект тел качения": [0xFFA000, BSF(freq), 5, 10, 0, 0.3],
            "Износ тел качения и сепаратора": [0xFF2400, FTF(freq), 4, 6, 0, 0.5]
        };
        break;
    case 3: //ШВП
        __mtx = {
            "FREQ": [0x89AC76, FREQ(freq), 6],
            "BPFO": [0x42AAFF, BPFO(freq), 6],
            "BPFI": [0x008000, BPFI(freq), 6],
            "BSF": [0xFFA000, BSF(freq), 6],
            "FTF": [0xFF2400, FTF(freq), 6]
        };

        __set = {
            "Износ гайки": [0x42AAFF, BPFO(freq), 6, 13, 0, 0.3],
            "Перекос гайки": [0x4285B4, BPFO(freq), 4, 12, 1, 0.1],
            "Дефект гайки": [0x6A5ACD, BPFO(freq), 6, 13, 0, 0.07],
            "Износ винта": [0x89AC76, FREQ(freq), 6, 13, 0, 0.3],
            "Перекос винта": [0x34C924, FREQ(freq), 6, 12, 1, 0.1],
            "Дефект винта": [0x008000, BPFI(freq), 6, 8, 0, 0.07],
            "Дефект тел качения": [0xFFA000, BSF(freq), 5, 10, 0, 0.3],
            "Износ тел качения и сепаратора": [0xFF2400, FTF(freq), 4, 6, 0, 0.5]
        };
        break;
    case 4: //редуктор
        __mtx = {
            "FREQ": [0x42AAFF, FREQ(freq), 6],
            "GTF2": [0x89AC76, GTF2(freq), 6],
            "GTFZ": [0xFFA000, GTFZ(freq), 6]
        };

        __set = {
            "Биение шестерни": [0x42AAFF, FREQ(freq), 6, 20, 0, 0.7],
            "Перекос шестерни": [0x4285B4, FREQ(freq), 6, 20, 1, 0.1],
            "Дефект шестерни": [0x6A5ACD, FREQ(freq), 6, 20, 0, 0.07],
            "Биение зубчатого колеса": [0x89AC76, GTF2(freq), 6, 20, 0, 0.7],
            "Перекос зубчатого колеса": [0x34C924, GTF2(freq), 6, 20, 1, 0.1],
            "Дефект зубчатого колеса": [0x008000, GTF2(freq), 6, 20, 0, 0.07],
            "Дефект зацепления": [0xFFA000, GTFZ(freq), 6, 20, 0, 0.07],
        };
        break;
    case 5: //ременная передача
        __mtx = {
            "FREQ": [0x42AAFF, FREQ(freq), 6],
```

```
            "BDF2": [0x89AC76, BDF2(freq), 6],
            "BDFB": [0xFFA000, BDFB(freq), 6]
        };

        __set = {
            "Биение ведущего шкива": [0x42AAFF, FREQ(freq), 6, 20, 0, 0.7],
            "Перекос ведущего шкива": [0x4285B4, FREQ(freq), 6, 20, 1, 0.1],
            "Износ ведущего шкива": [0x6A5ACD, FREQ(freq), 6, 20, 0, 0.3],
            "Биение ведомого шкива": [0x89AC76, BDF2(freq), 6, 20, 0, 0.7],
            "Перекос ведомого шкива": [0x34C924, BDF2(freq), 6, 20, 1, 0.1],
            "Износ ведомого шкива": [0x008000, BDF2(freq), 6, 20, 0, 0.3],
            "Дефект ремня": [0xFFA000, BDFB(freq), 6, 20, 0, 0.07],
        };
        break;
    case 6: //зубчатый ремень (цепная передача)
        __mtx = {
            "FREQ": [0x42AAFF, FREQ(freq), 6],
            "CBDF2": [0x89AC76, CBDF2(freq), 6],
            "CBFZ": [0xFFA000, CBFZ(freq), 6],
            "CBDFB": [0xFF2400, CBDFB(freq), 6]
        };

        __set = {
            "Биение ведущего шкива (звездочки)": [0x42AAFF, FREQ(freq), 6, 20, 0, 0.7],
            "Перекос ведущего шкива (звездочки)": [0x4285B4, FREQ(freq), 6, 20, 1, 0.1],
            "Дефект зубьев ведущего шкива (звездочки)": [0x6A5ACD, FREQ(freq), 6, 20, 0,
0.07],
            "Биение ведомого шкива (звездочки)": [0x89AC76, CBDF2(freq), 6, 20, 0, 0.7],
            "Перекос ведомого шкива (звездочки)": [0x34C924, CBDF2(freq), 6, 20, 1, 0.1],
            "Дефект зубьев ведомого шкива (звездочки)": [0x008000, CBDF2(freq), 6, 20, 0,
0.07],
            "Равномерный износ ремня (цепи)": [0xFFA000, CBFZ(freq), 6, 20, 0, 0.3],
            "Дефект зубьев ремня (цепи)": [0xFF2400, CBDFB(freq), 6, 20, 0, 0.07],
        };
        break;
    case 7: //помпа (насос)

        break;
    case 8: //планетарый редуктор
        __mtx = {
            "FREQ": [0x42AAFF, FREQ(freq), 6],
            "FREQ - PGF2": [0x6A5ACD, pgN * (FREQ(freq) - PGF2(freq)), 6],
            "PGFSAT": [0x89AC76, PGFSAT(freq), 6],
            "PGF2": [0x34C924, PGF2(freq), 6]
        };

        __set = {
            "Биение солнечной шестерни": [0x42AAFF, FREQ(freq), 6, 20, 0, 0.7],
            "Перекос солнечной шестерни": [0x4285B4, FREQ(freq), 6, 20, 1, 0.1],
            "Дефект зубьев солнечной шестерни": [0x6A5ACD, pgN * (FREQ(freq) - PGF2(freq)), 6,
20, 0, 0.07],
            "Дефект зубьев сателлита": [0x89AC76, PGFSAT(freq), 6, 20, 1, 0.07],
            "Перекос короны": [0x34C924, pgN * PGF2(freq), 6, 20, 1, 0.1],
```

```
            "Дефект зубьев короны": [0x008000, pgN * PGF2(freq), 6, 20, 0, 0.07],
            "Биение водила": [0xFFA000, PGF2(freq), 6, 20, 0, 0.7],
        };
        break;
    case 9: //турбина

        break;
    case 10: //электродвигатель

        break;
  };

  return {
    set: __set,
    mtx: __mtx
  };
}; //набор признаков предполагаемых дефектов
```

6.7 Библиотека функций GTLd-daemon:

6.7.1 Асинхронная обработка данных в демоне - Gtld async.

```
let device = gtl.add_device("ni");
device.data_block = 1;

device.start("Dev1", 51200);

gtl.log.info("inputs cnt", device.analog_inputs.length);

var rms = gtl.add_rms_data(
  {
    "src" : device.analog_inputs[3],
    "name" : "rms",
    "color" : 0xff0000,
    "time" : 1
  }
);



device.received_data.connect(
 function()

 {

  gtl.log.info("rms", rms.back_value);

 }

);
```

6.7.2 Использование рекордера в движке - Gtld recorder.

```
let recorder = gtl.add_recorder(
{
  "src":[
    gtl.analog_inputs[0],
    gtl.analog_inputs[1]
  ],
  "prehistory" : 15.5,

}
);



recorder.finished.connect(
  function()
  {

    gtl.log.info("file", "finished");

    gtl.free(recorder);

     gtl.diagnostic.stop();
  }
);



gtl.diagnostic.interval = .1;

function diagnose()
{

  gtl.log.info("time", recorder.time());

  if(gtl.diagnostic.time > 8 && !recorder.is_writing())
  {
    if(!recorder.start(10, "no comments", "E:/data/gtl", "record from script"))
      gtl.log.error("file", "error writing file");

  }



  if(gtl.diagnostic.time > 20)
  {
    gtl.log.info("diagnostic", "stop");

    gtl.free(recorder);

    gtl.diagnostic.stop();
  }
```

```
}
```

6.8 Примеры

6.8.1 Варианты компоновки результатов диагностики - Gtl.results var.

В разработке

6.8.2 Примеры диагностических скриптов - Gtl.scripts var.

Определение разности фаз определенного набора частот

```
//определяем переменные
var frq_arr = [200, 300, 400, 500]; //массив частот для которых определяем разность фаз
var i = 0; //
var res = {}; //объект результатов вычислений разности фаз

//определение разности фаз полигармонических сигналов
var delta = gtl.add_delta_phase_spec(gtl.analog_inputs[0], gtl.analog_inputs[1]);
delta.max_frequency = 1000; //граничная частота спектра вычисления фазы, Гц;
delta.resolution = 1; //частотное разрешение, Гц;
delta.frequency = frq_arr[0]; //частота, для которой вычисляется разница фаз, Гц;

gtl.diagnostic.interval = delta.acq_time; //интервал запуска функции диагностики

function diagnose() {
    gtl.log.info("Индекс", i);
    gtl.log.info("Частота", frq_arr[i]);
    gtl.log.info("Разница фаз на частоте", delta.value);

    res[frq_arr[i]] = delta.value; //записываем данные в объект

    if (i < frq_arr.length - 1) {
        i = i + 1;
        delta.frequency = frq_arr[i];
    } else {
        gtl.results = res; //записываем объект в результат
        gtl.diagnostic.stop(); //останавливаем диагностику
    };
};
```

Определение параметров последовательно по нескольким каналам вибрации

```
var record = gtl.options.record;
var signals = gtl.options.record.signalsModel;

//фильтр для расчета СКЗ
var idx = 0; //индекс канала вибрации
```

```
var filter = gtl.add_filter_iir(gtl.analog_inputs[signals[idx].portNumber]); //объявление переменной фильтра
filter.kind = gtl.filter_iir.butterworth; //тип окна
filter.type = gtl.filter_iir.lowpass; //тип фильтра (ФНЧ)
filter.order = 8; //порядок фильтра
filter.frequency = 1000; //граничная частота фильтра

//определение среднего квадратического значения в диапазоне до 1000 Гц
var rms = gtl.add_value_rms(filter); //объявление переменной СКЗ
rms.time = 1; //время выборки
rms.avg_cnt = 4; //количество усреднений

//[Диагностика]
gtl.diagnostic.interval = rms.time * rms.avg_cnt;
var cnt = record.signalsModel.length; //считаем количество каналов вибрации

function diagnose() {
    if (idx < cnt) {
        filter = gtl.add_filter_iir(gtl.analog_inputs[signals[idx].portNumber]);
        gtl.log.info("RMS channel " + idx, rms.value);
        idx = idx + 1;

    } else { gtl.diagnostic.stop(); }
};
```

6.8.3 Примеры построения трендов параметров - Gtl.trends var

Построение тренда изменения СКЗ виброскорости в диапазоне 30-400 Гц

```
let time = 10; //указанная длительность записи или полная длина gtl.options.record.playerTime;
let interval = 1; //интервал расчета параметра
let counter = 0; //счетчик
let rms_arr = []; //массив значений СКЗ

// создаём фильтр 30-400 Гц
let __BP = gtl.add_filter_iir(gtl.analog_inputs[0]); //объявление переменной фильтра
__BP.kind = gtl.filter_iir.butterworth; //тип окна
__BP.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
__BP.order = 6; //порядок фильтра
__BP.frequency = 215; //центральная частота фильтра
__BP.width = 370; //ширина полосы

// выполняем интегрирование
let __int= gtl.add_intg(__BP); //интегрирование
__int.taps = 1; //степень интегрирования (скорость из ускорения - 1-нарное интегрирование)
__int.scale = 1000; //переводим метры в миллиметры

// определение СКЗ виброскорости
let rms = gtl.add_value_rms(__int); //объявление переменной СКЗ
rms.time = interval; //время выборки

let plot = gtl.plots.add("Тренд СКЗ виброскорости");
```

```
gtl.diagnostic.interval = gtl.acq_time; //интервал запуска функции диагностики
gtl.log.info("Необходимое время измерения", gtl.acq_time);
gtl.log.info("Длительность сигнала", time);

function diagnose() {
  counter += 1;
  rms_arr.push(rms.value);
  gtl.log.info("СКЗ виброскорости в диапазоне 30-400 Гц", rms.value);

  if (counter >= Math.round(time / interval)) {
    plot.add(
      {
        color: 0x0000FF,
        name: "RMS(V)",
        x: interval,
        y: rms_arr
      }
    ); //рисуем тренд СКЗ виброскорости на plot

    gtl.diagnostic.stop()
  }
};
```

6.9 Скрипты анализатора с портретами дефектов

6.9.1 maskMethod

```
"use strict";
var options = gtl.options;
var point = gtl.options.point;
var record = gtl.options.record;
var signals = gtl.options.record.signalsModel;
var tacho = gtl.options.record.tachoOptions;

var fnc = gtl.import("userFunctions.js");
let maskClass = gtl.import("maskClass.js").maskClass;
let mtxClass = gtl.import("mtxClass.js").mtxClass;
let freqClass = fnc.freqClass;

let src = gtl.analog_inputs[signals[0].portNumber];
var __freq = new freqClass(
    {
        // src: gtl.analog_inputs[tacho.tachoChannel], //источник сигнала
        // freq: 15, //центральная частота
        width: 10, //диапазон разброса частоты, Гц
        time: 2, //интервал измерения, сек
        avg: 10 //количество усреднений
    }
); //определяем частоту вращения
```

```
let plot0 = gtl.plots.add("Мониторинговый спектр");
let plot1 = gtl.plots.add("Спектр вибрации");
let plot2 = gtl.plots.add("Спектр огибающей");

//фильтр для формирования спектра огибающей
var filter_spen = gtl.add_filter_iir(src); //назначение переменной фильтра
filter_spen.kind = gtl.filter_iir.butterworth; //тип окна
filter_spen.type = gtl.filter_iir.bandpass; //тип фильтра (полосовой)
filter_spen.order = 10; //порядок фильтра
filter_spen.frequency = 8000; //центральная частота полосового фильтра
filter_spen.width = 1840; //ширина полосы фильтра

var ausp0 = gtl.create_ausp(
   {
      src: src,
      frequency: 25600,
      resolution: 8,
      average: 6,
      overlap: 0.5,
      window: gtl.spec.rectangular,
      view: gtl.spec.db
   }
);
gtl.log.info("Построение мониторингового спектра: ", "Выполнение...");

var ausp = gtl.create_ausp(
   {
      src: src,
      frequency: 1600,
      resolution: 1,
      average: 6,
      overlap: 0.5,
      window: gtl.spec.rectangular,
      view: gtl.spec.db
   }
);
gtl.log.info("Построение спектра вибрации: ", "Выполнение...");

var spen = gtl.create_spen(
   {
      src: filter_spen,
      frequency: 400,
      resolution: 0.25,
      average: 8,
      overlap: 0.5,
      window: gtl.spec.rectangular,
      view: gtl.spec.db
   }
);
gtl.log.info("Построение спектра огибающей: ", "Выполнение...");

var __mes = fnc.getStdMeasures(
   {
```

```
        src: src,
        time: 5,
        avg: 4
    }
);
gtl.log.info("Формирование стандартных измерений: ", "Выполнение...");

//[Диагностика]
gtl.diagnostic.interval = gtl.acq_time;
let __result = {}; //объект для формирования результат

function diagnose() {
    let __frq = __freq.getResult().value; //получаем результат расчета частоты вращения
    gtl.log.info("Расчетная частота вращения, Гц: ", __frq);

    plot0.add(
        {
            color: 0x001E90FF,
            name: "AUSPm",
            x: ausp0.resolution,
            y: ausp0.data
        }
    ); //рисуем мониторинговый спектр вибрации на plot
    gtl.log.info("Построение мониторингового спектра: ", "Выполнено!");

    plot1.add(
        {
            color: 0x0000FF00,
            name: "AUSP",
            x: ausp.resolution,
            y: ausp.data
        }
    ); //рисуем спектр вибрации на plot
    gtl.log.info("Построение спектра вибрации: ", "Выполнено!");

    let __dset = fnc.getDefSet(__frq).set; //получаем набор портретов предполагаемых
дефектов
    let __spen_tools = fnc.createTools(
        {
            spec: spen, //спектр для построения модели (объект)
            set: __dset, //источник данных для построения частотных линий
            lvl: 10, //уровень обнаружения гармоники
            tol: 1 //коридор обнаружения гармоники
        }
    ); //формируем компоненты и частотные линии на спектре огибающей

    plot2.add(
        {
            color: 0x00ff0000,
            name: "SPEN",
            x: spen.resolution,
            y: spen.data,
            spec_tools: __spen_tools.to_json()
```

```
        }
    ); //рисуем спектр огибающей на plot
    gtl.log.info("Построение спектра огибающей: ", "Выполнено!");

    let __mask = new maskClass(
        {
            spec: spen, //спектр для построения модели (объект)
            set: __dset, //источник данных для построения портретов дефектов
            filter: filter_spen, //полосовой фильтр (для определения разницы амплитуд гармонической и случайной составляющей)
            level: 10, //уровень обнаружения гармоник
            tol: 1, //коридор обнаружения гармоники
            color: 0xFF0000, //цвет отрисовки модели спектра в формате HEX
            visible: true //отображение графиков
        }
    );

    let __maskResult = __mask.getResult(); //обнаруженные дефекты
    __result["correlation"] = __maskResult.correlation;
    __result["modulation"] = __maskResult.modulation;
    __result["defects"] = __maskResult.defects;

    let __mset = fnc.getDefSet(__frq).mtx;
    let __mtx = new mtxClass(
        {
            spec: spen, //спектр для анализа гармоник
            set: __mset, //источник данных для построения портретов дефектов
            filter: filter_spen, //полосовой фильтр (для определения разницы амплитуд гармонической и случайной составляющей)
            lvl: 10, //уровень обнаружения гармоники
            tol: 1, //коридор обнаружения гармоники
        }
    )

    __result["x_matrix"] = __mtx.getMatrix(); //датасет для нейросети
    gtl.log.info("Формирование матрицы гармоник: ", "Выполнено!");

    let measures = {
        rms_A2_3000: __mes.rms_A2_3000.value,
        rms_A2_10000: __mes.rms_A2_10000.value,
        rms_A100_10000: __mes.rms_A100_10000.value,
        ampl_A100_10000: __mes.ampl_A100_10000.value,
        ampl_full: __mes.ampl_full.value,
        rms_V2_1000: __mes.rms_V2_1000.value,
        rms_V10_1000: __mes.rms_V10_1000.value,
        rms_S2_200: __mes.rms_S2_200.value,
        ptp_S2_200: __mes.ptp_S2_200.value,
        kurt_full: __mes.kurt_full.value,
        kurt_2_5k: __mes.kurt_2_5k.value,
        kurt_5_10k: __mes.kurt_5_10k.value,
        kurt_10_20k: __mes.kurt_10_20k.value,
        kurt_30_40k: __mes.kurt_30_40k.value,
        kurt_40_80k: __mes.kurt_40_80k.value,
```

```
    };

    __result["const"] = spen.data[0]; //уровень постоянной составляющей спектра огибающей
    __result["measures"] = measures; //набор стандартных измерений

    gtl.log.info("Формирование результата: ", "Выполнено!");
    gtl.results = __result;
    gtl.diagnostic.stop();
};
```

6.9.2 mipMethod

```
"use strict";
var options = gtl.options;
var point = gtl.options.point;
var record = gtl.options.record;
var signals = gtl.options.record.signalsModel;
var tacho = gtl.options.record.tachoOptions;

let fnc = gtl.import("userFunctions.js");
let mipClass = gtl.import("mipClass.js").mipClass;
let freqClass = fnc.freqClass;

let src = gtl.analog_inputs[signals[0].portNumber];
var __freq = new freqClass(
    {
        // src: gtl.analog_inputs[tacho.tachoChannel], //источник сигнала
        // freq: 15, //центральная частота
        width: 10, //диапазон разброса частоты, Гц
        time: 2, //интервал измерения, сек
        avg: 10 //количество усреднений
    }
); //определяем частоту вращения

//фильтр для формирования сигнала
let __fltr = gtl.add_filter_iir(src);
__fltr.kind = gtl.filter_iir.butterworth; //тип окна
__fltr.type = gtl.filter_iir.bandpass; //тип фильтра (ФНЧ)
__fltr.order = 10; //порядок фильтра
__fltr.frequency = 32000; //центральная частота полосового фильтра
__fltr.width = 7360; //ширина полосового фильтра

//формирование амплитуд для расчета коврового уровня dBc (за 200 импульсов в сек)
let __cpt = gtl.add_value_ampl(__fltr);
__cpt.name = "Общий уровень";
__cpt.color = 0x2B6CC4;
__cpt.time = 0.005; //время выборки
__cpt.avg_cnt = 200; //количество отсчетов

//формирование амплитуд для расчета максимального уровня dBm (максимум за 2 сек)
let __max = gtl.add_value_ampl(__fltr);
```

```
__max.name = "Микроударные импульсы";
__max.color = 0x2B6CC4;
__max.time = 0.001; //время выборки
__max.avg_cnt = 2000; //количество отсчетов

let __imp = gtl.create_moving_max(
    {
        src: __fltr,
        name: "Форма импульсов",
        time: 0.001
    }
); //формируем сигнал максимальных амплитуд для построения спектра
gtl.log.info("Построение формы ипульсов: ", "Выполнение...");

var spen = gtl.create_spen(
    {
        src: __imp,
        frequency: 200,
        resolution: 0.25,
        average: 8,
        overlap: 0.5,
        window: gtl.spec.rectangular,
        view: gtl.spec.db
    }
); //вычисляем спектр огибающей формы импульсов
gtl.log.info("Построение спектра огибающей формы импульсов: ", "Выполнение...");

let plot1 = gtl.plots.add("Spectral");

//[Диагностика]
gtl.diagnostic.interval = gtl.acq_time;
function diagnose() {
    let __frq = __freq.getResult().value; //получаем результат расчета частоты вращения
    gtl.log.info("Расчетная частота вращения, Гц: ", __frq);

    let __dset = {
        "Дефект наружного кольца": [0x6A5ACD, fnc.BPFO(__frq), 6, 13, 0, 0.07],
        "Износ внутреннего кольца": [0x89AC76, fnc.FREQ(__frq), 6, 13, 0, 0.3],
        "Дефект внутреннего кольца": [0x008000, fnc.BPFI(__frq), 6, 8, 0, 0.07],
        "Дефект тел качения": [0xFFA000, fnc.BSF(__frq), 5, 10, 0, 0.3],
        "Износ тел качения и сепаратора": [0xFF2400, fnc.FTF(__frq), 4, 6, 0, 0.5]
    }

    let __spen_tools = fnc.createTools(
        {
            spec: spen, //спектр для построения модели (объект)
            set: __dset, //источник данных для построения частотных линий
            lvl: 10, //уровень обнаружения гармоники
            tol: 1 //коридор обнаружения гармоники
        }
    ); //формируем компоненты и частотные линии на спектре огибающей

    let spenPlot = plot1.add(
```

```
        {
            color: 0x00ff0000,
            name: "SPEN",
            x: spen.resolution,
            y: spen.data,
            spec_tools: __spen_tools.to_json()
        }
    ); //рисуем спектр огибающей формы импульсов
    gtl.log.info("Построение спектра огибающей формы импульсов: ", "Выполнено!");

    let __mip = new mipClass(
        {
            src1: __cpt, //объект с массивом данных для расчета коврового уровня
            src2: __max, //объект с массивом данных определения маскимума импульсов
            freq: __frq, //частота вращения
            d_inner: options.rbInnerD, //диаметр внутреннего кольца подшипника
            visible: true //отображение формы импульсов
        }
    );

    gtl.results = __mip.getResult();
    gtl.log.info("Формирование результата: ", "Выполнено!");
    gtl.diagnostic.stop();
};
```

6.9.3 peakMethod

```
"use strict";
var options = gtl.options;
var point = gtl.options.point;
var record = gtl.options.record;
var signals = gtl.options.record.signalsModel;
var tacho = gtl.options.record.tachoOptions;

let fnc = gtl.import("userFunctions.js");
let getParams = gtl.import("peakClass.js").getParams;
let peakClass = gtl.import("peakClass.js").peakClass;
let freqClass = fnc.freqClass;

let src = gtl.analog_inputs[signals[0].portNumber];
var __freq = new freqClass(
    {
        // src: gtl.analog_inputs[tacho.tachoChannel], //источник сигнала
        // freq: 15, //центральная частота
        width: 10, //диапазон разброса частоты, Гц
        time: 2, //интервал измерения, сек
        avg: 10 //количество усреднений
    }
); //определяем частоту вращения

let __params = getParams(__freq); //получаем параметры фильтра и спектра
```

```
//фильтр для формирования пиковой формы сигнала
let __fltr = gtl.add_filter_iir(src);
__fltr.kind = gtl.filter_iir.butterworth; //тип окна
__fltr.type = gtl.filter_iir.highpass; //тип фильтра (ФНЧ)
__fltr.order = 10; //порядок фильтра
__fltr.frequency = 500; //__params.filter.frequency; //граничная частота фильтра

//формирование амплитуд для построения пиковой формы сигнала
let __ampl = gtl.add_value_peak_to_peak(__fltr); //gtl.add_value_ampl(__fltr);
__ampl.name = "Пиковая форма";
__ampl.color = 0x2B6CC4;
__ampl.time = 0.001; //время выборки
__ampl.avg_cnt = 7700; //количество отсчетов

let __max = gtl.create_moving_peak_to_peak(
   {
      src: __fltr,
      name: "PtP форма",
      time: 0.001
   }
); //формируем сигнал максимальных амплитуд для построения спектра
gtl.log.info("Построение пиковой формы: ", "Выполнение...");

var ausp = gtl.create_ausp(
   {
      src: __max,
      frequency: 200, //__params.spec.frequency
      resolution: 0.25, //__params.spec.resolution
      average: 1,
      overlap: 0.5,
      window: gtl.spec.rectangular,
      view: gtl.spec.unit
   }
); //вычисляем спектр пиковой формы
gtl.log.info("Построение спектра пиковой формы: ", "Выполнение...");

let plot = gtl.plots.add("Spectral");

//[Диагностика]
gtl.diagnostic.interval = gtl.acq_time;
function diagnose() {
   let __frq = __freq.getResult().value; //получаем результат расчета частоты вращения
   gtl.log.info("Расчетная частота вращения, Гц: ", __frq);

   let __dset = {
      "Дефект наружного кольца": [0x6A5ACD, fnc.BPFO(__frq), 6, 13, 0, 0.07],
      "Износ внутреннего кольца": [0x89AC76, fnc.FREQ(__frq), 6, 13, 0, 0.3],
      "Дефект внутреннего кольца": [0x008000, fnc.BPFI(__frq), 6, 8, 0, 0.07],
      "Дефект тел качения": [0xFFA000, fnc.BSF(__frq), 5, 10, 0, 0.3],
      "Износ тел качения и сепаратора": [0xFF2400, fnc.FTF(__frq), 4, 6, 0, 0.5]
   }
   let __ausp_tools = fnc.createTools(
```

```
        {
            spec: ausp, //спектр для построения модели (объект)
            set: __dset, //источник данных для построения частотных линий
            lvl: 0.01, //уровень обнаружения гармоники
            tol: 1 //коридор обнаружения гармоники
        }
    ); //формируем компоненты и частотные линии на спектре

    let auspPlot = plot.add(
        {
            color: 0x00ff0000,
            name: "AUSP",
            x: ausp.resolution,
            y: ausp.data,
            spec_tools: __ausp_tools.to_json()
        }
    ); //рисуем спектр пиковой формы
    gtl.log.info("Построение спектра пиковой формы: ", "Выполнено!");

    let __peak = new peakClass(
        {
            src: __ampl, //объект с массивом данных пиковой формы
            freq: __frq, //частота вращения, Гц
            spec: ausp, //спектр пиковой формы
            level: 0.01, //уровень обнаружения гармоник
            tol: 1, //коридор обнаружения гармоники
            visible: true, //отображение графиков
        }
    );

    let __result = __peak.getResult();
    gtl.results = __result;

    gtl.log.info("Формирование результата: ", "Выполнено!");
    gtl.diagnostic.stop();
};
```

6.9.4 testAutoFreq

```
"use strict";
var options = gtl.options;
var point = gtl.options.point;
var record = gtl.options.record;
var signals = gtl.options.record.signalsModel;
var tacho = gtl.options.record.tachoOptions;

var freqClass = gtl.import("userFunctions.js").freqClass;

var __frq = new freqClass(
    {
        // src: gtl.analog_inputs[tacho.tachoChannel], //источник сигнала
```

```
        width: 10, //граничная частота определения, Гц
        // time: 1, //интервал измерения, сек
        // avg: 100 //количество усреднений
    }
); //определяем частоту вращения



//[Диагностика]
gtl.log.info("time", __frq.getTime());
gtl.diagnostic.interval = __frq.getTime() + 1;

function diagnose() {
    let __result = __frq.getResult();

    gtl.results = __result;
    gtl.diagnostic.stop();
};
```

Решаем задачи и реализуем идеи
безопасного будущего промышленности

gtlab.pro
+7 (83130) 4-94-44
info@gtlab.pro

Нижегородская область,
г. Саров, ул. Шверника, 17 «Б»