

# ZCAD Руководство пользователя

## Содержание

1. Общая информация .....	4
1.1. Лицензия .....	4
1.2. Отказ от ответственности .....	4
2. Быстрый старт .....	6
2.1. Системные требования .....	6
2.2. Установка .....	6
2.3. Запуск .....	6
3. Работа с программой .....	7
3.1. Интерфейс пользователя .....	7
3.1.1. Общий обзор .....	7
3.1.2. AnchorDocking .....	11
Панели инструментов .....	11
Окна .....	12
3.1.3. Навигаторы .....	14
3.2. Примитивы .....	17
3.2.1. Device .....	18
3.3. Расширения примитивов .....	20
3.3.1. extdrLayerControl .....	20
3.3.2. extdrSmartTextEnt .....	21
3.3.3. extdrVariables .....	22
Перечень простых типов данных .....	22
Централизация .....	23
3.4. Команды .....	23
3.4.1. 3DPoly .....	23
3.4.2. About .....	23
3.4.3. Arc .....	24
3.4.4. BEdit .....	24
3.4.5. Cam_reset .....	25
3.4.6. Cancel .....	25
3.4.7. Circle .....	25
3.4.8. Copy .....	25
3.4.9. CopyBase .....	26
3.4.10. CopyClip .....	26
3.4.11. CutClip .....	26
3.4.12. DataExport .....	27
Формат фильтра по типам примитивов .....	28

Формат фильтра по свойствам примитивов .....	29
Формат скрипта экспорта .....	30
3.4.13. DataImport .....	32
3.4.14. DevDefSync .....	32
3.4.15. DockingOptions .....	33
3.4.16. DWGClose .....	33
3.4.17. DWGNew .....	33
3.4.18. DWGNext .....	34
3.4.19. DWGPrev .....	34
3.4.20. Erase .....	34
3.4.21. extdrRemove .....	34
3.4.22. extdrAllList .....	35
3.4.23. extdrEntsList .....	35
3.4.24. extdrAdd .....	35
3.4.25. Layer .....	35
3.4.26. Load .....	36
3.4.27. LoadActions .....	36
3.4.28. LoadLayout .....	37
3.4.29. LoadMenus .....	37
3.4.30. LoadPalettes .....	37
3.4.31. LoadToolbars .....	37
3.4.32. MatchProp .....	38
3.4.33. Merge .....	38
3.4.34. MergeBlocks .....	38
3.4.35. Mirror .....	39
3.4.36. Move .....	39
3.4.37. Rotate .....	39
3.4.38. SaveAs .....	39
3.4.39. SaveLayout .....	40
3.4.40. SaveOptions .....	40
3.4.41. UpdatePO .....	40
3.4.42. VarsEd .....	40
3.4.43. VarsEdBD .....	41
3.4.44. VarsEdSel .....	41
3.4.45. VarsLink .....	41
4. Настройка программы .....	42
4.1. Параметры работы .....	42
4.1.1. Пути .....	42
4.1.2. Графика .....	43
4.1.3. Отображение .....	44
4.1.4. Система .....	44

4.1.5. Сохранение .....	45
4.1.6. Черчение .....	45
4.2. Структура директорий .....	46
4.3. Переключатели командной строки .....	48
5. Для разработчика .....	50
5.1. Сборка программы из исходников .....	50
5.1.1. Установка Lazarus .....	50
5.1.2. Получение ZCAD .....	51
5.1.3. Установка пакетов от которых зависит ZCAD .....	51
5.1.4. Компиляция ZCAD .....	51
5.2. Локализация программы .....	52
5.3. Написание документации ZCAD .....	53
5.3.1. Установка AsciiDoctor для Windows .....	53
5.3.2. Установка GraphViz .....	54
5.3.3. Генерация руководства пользователя .....	54

# 1. Общая информация

ZCAD - программа (далее по тексту - программа) с открытым исходным кодом, разрабатывается на языке Pascal с использованием IDE Lazarus и компилятора Free Pascal.

Автор - Андрей Зубарев (далее по тексту - автор) адрес электронной почты [zamtmn@yandex.ru](mailto:zamtmn@yandex.ru)

Участник - Владимир Бобров

Участник - Владимир Абрамов адрес электронной почты [vl-sx@yandex.ru](mailto:vl-sx@yandex.ru)

Есть два варианта сборки программы:

ZCAD - базовый набор CAD инструментов; ZCADElectroTech - расширенный набор инструментов для электротехнического проектирования

## 1.1. Лицензия

ZCAD - это программное обеспечение с открытым исходным кодом. Вы можете получить исходный код, а также разрешается копировать, перераспределять и изменять его в соответствии с условиями лицензии LGPL2 с исключениями. Для получения дополнительной информации, в том числе о доступности исходного кода, посетите страницу программы <https://github.com/zamtmn/zcad> или обратитесь к автору.

## 1.2. Отказ от ответственности

В соответствии с действующим законодательством, автор отказывается от каких-либо заверений и гарантий, предоставление которых может иным образом подразумеваться, и отказывается от ответственности в отношении программы, поставляемой вместе с программой информации и их использования.

1. Поставляемой вместе с программой информации предназначена для свободного ознакомления пользователей с вопросами, которые могут представлять для них интерес.
2. Вся информация предоставляется в исходном виде, без гарантий полноты или своевременности, и без иных, явно выраженных или подразумеваемых гарантий. Использование программы и поставляемой вместе с программой информации осуществляется исключительно по вашему усмотрению и на ваш риск.
3. Автор прикладывает все усилия, чтобы обеспечить пользователей точной и достоверной информацией, но в то же время не исключает возможности возникновения ошибок.
4. ZCAD — это программа, работающая по принципу «как есть», без заключения каких-либо договорённостей или договоров между пользователями программы и автором, либо кем-то ещё, любым образом связанными с этим или родственными ему проектами, которые (договора и договоренности) могут стать предметом прямых претензий.
5. Автор не дает каких-либо заверений или гарантий в отношении программы, в том числе, без ограничения, в отношении своевременности, актуальности, точности,

полноты, достоверности, доступности или соответствия для какой-либо конкретной цели, в отношении того, что: при использовании программы не возникнет ошибок, а также, что поставляемая вместе с программой информация не нарушает прав третьих лиц.

6. В соответствии с действующим законодательством, Автор отказывается от каких-либо заверений и гарантий, предоставление которых может иным образом подразумеваться, и ответственности в отношении программы. Ни при каких обстоятельствах автор не будет нести ответственности ни перед какой стороной за какой-либо прямой, не прямой, особый или иной косвенный ущерб в результате любого использования программы, даже если автор будет явно поставлен в известность о возможности такого ущерба.
7. Автор оставляет за собой право вносить изменения без уведомления о них пользователей.
8. Если в соответствии с действующими законами какие-либо условия будут признаны недействительными, остальные условия остаются в полной силе.

Используя ZCAD и информацию поставляемую вместе с программой, вы выражаете свое согласие с «Отказом от ответственности» и установленными Правилами и принимаете всю ответственность, которая может быть на вас возложена.

Автор в любое время вправе внести изменения в Правила, которые вступают в силу немедленно. Продолжение пользования программой после внесения изменений означает ваше автоматическое согласие на соблюдением новых правил.

## 2. Быстрый старт

### 2.1. Системные требования

Программа работает под управлением операционных систем Windows и Linux, на 32 разрядных и 64 разрядных процессорах. Для работы программы требуется 4GB оперативной памяти

### 2.2. Установка

ZCAD не требует какой-либо процедуры установки на компьютер. Вы можете скачать дистрибутив программы с страницы <https://github.com/zamtmn/zcad/releases>, для использования достаточно разархивировать полученный архив

### 2.3. Запуск

Исполняемый файл находится в подпапке соответствующей Вашей платформе в папке дистрибутива **bin**. Достаточно просто его запустить

# 3. Работа с программой

## 3.1. Интерфейс пользователя

### 3.1.1. Общий обзор

Интерфейс программы по умолчанию представлен на скриншоте:

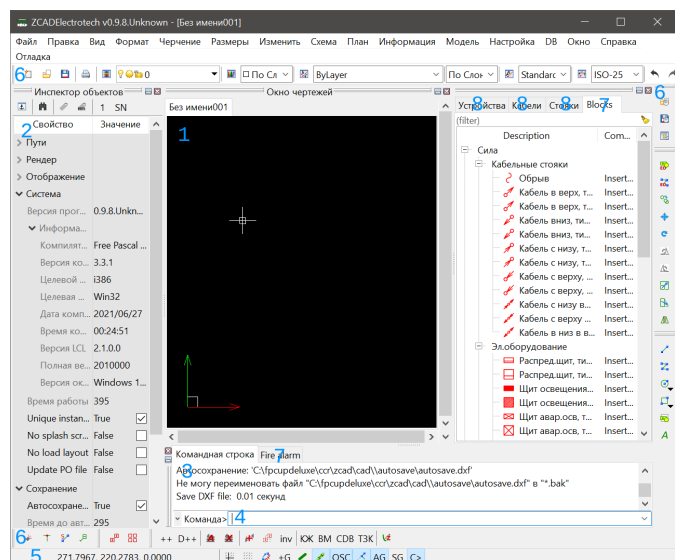


Рисунок 1. Внешний вид окна программы

Можно выделить следующие основные элементы:

#### 1. Окно отображения

Тут производится отображение и редактирование графической информации.

Навигация по чертежу осуществляется мышкой. Масштабировать чертеж можно вращая колесо мыши, если при этом нажать и удерживать клавишу **Shift** коэффициент масштабирования будет меньше и точки временных привязок не будут сброшены. Панорамировать чертеж можно двигая мышью с нажатой средней кнопкой. Если при панорамировании удерживать клавишу **Ctrl**, будет осуществляться вращение камеры. Также для навигации доступны некоторые сочетания клавиш !!NEEDLINK!!

Редактирование чертежа также производится курсором мыши, при этом его форма зависит от ожидаемого действия: **перекрестие** - указание точки на чертеже, **прямоугольник** - выбор графического примитива. Комбинация прямоугольника и перекрестия - можно как выбрать примитив, так и указать точку на чертеже.

Программа позволяет выбирать примитивы несколькими способами:

- кликнув прямоугольником по примитиву - примитив будет добавлен к текущему набору выбора
- рамкой выбора - при клике по пустой области чертежа начинает чертиться прямоугольная рамка выбора, при следующем клике построение рамки будет закончено. Рамка начерченная слева на право называется **прямой** и рисуется сплошными

линиями с синей заливкой, справа на лево - называется **обратной** и рисуется пунктирными линиями с зеленой заливкой. По завершению процедуры будут выбраны все примитивы полностью попавшие в прямую рамку или хотя бы частично попавшие в обратную рамку. При черчении рамки чертеж можно масштабировать и панорамировать, если начальная точка при этом выйдет за границы экрана, рамка будет подрезана по экрану. Данное поведение можно изменить нажав и удерживая клавишу **control** перед указанием второй точки рамки. О наличии и отсутствии подрезки можно судить по диагональным линиям рисуемой рамки.

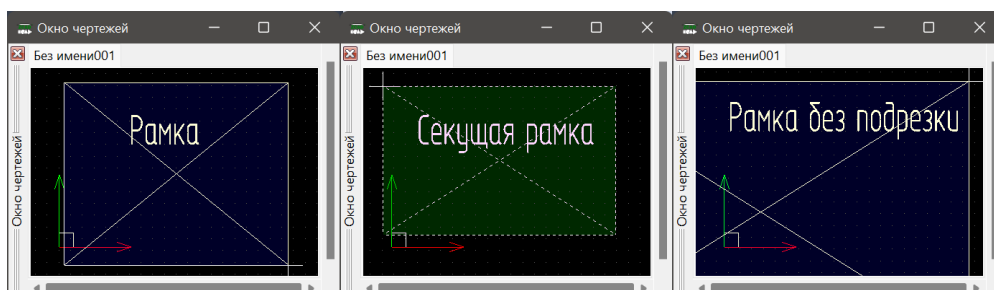


Рисунок 2. Варианты рамок выбора

## 2. Инспектор объектов

Тут отображаются и редактируются свойства различных сущностей. По умолчанию в нем отображаются настройки программы, но в зависимости от ситуации могут отображаться например свойства выделенных примитивов или параметры запущенной команды.

На рис. **Внешний вид окна программы** в инспекторе отображаются настройки программы. Если в области отображения выделить несколько примитивов, инспектор примет примерно следующий вид:

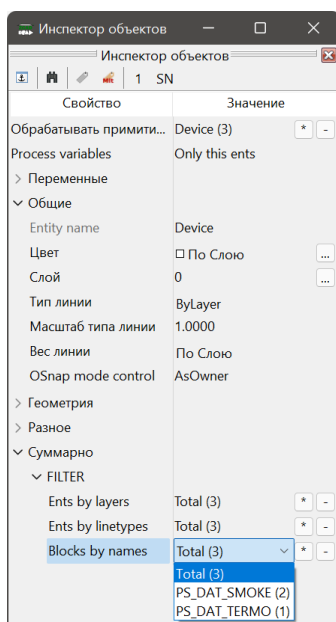


Рисунок 3. Свойства выделенных примитивов

Имена свойств доступных только для чтения отображаются серым цветом. Если выделенные примитивы имеют различные значения одноименных свойств, данные значения отображаются как "Разный". Если данные значения фактически отличаются, но из за настроек отображения !!NEEDLINK!! (округления) выглядят одинаково, данные значения помечаются знаком "≈".



При клике на значение свойства откроется строка редактирования и можно будет изменить значение свойства. Для некоторых свойств доступны специализированные редакторы, открывающиеся в отдельном окне по нажатию кнопки [...]

Свойства примитивов структурированы следующим образом:

```
Failed to generate image: Could not load PlantUML. Either require 'asciidoctor-
diagram-plantuml' or specify the location of the PlantUML JAR(s) using the
'DIAGRAM_PLANTUML_CLASSPATH' environment variable. Alternatively a PlantUML binary can
be provided (plantuml-native in $PATH).
!include styles/defaulttree-style.iuml
legend
Инспектор
|_ Обрабатывать_примитивы
|_ Обработка_переменных
|_ Связанные переменные
|_ Переменные связанных примитивов
|_ Переменные
|_ Свои переменные
|_ Расширения
|_ Свойства расширений примитивов
|_ Общие
|_ Основные свойства примитивов
|_ Геометрия
|_ Геометрические свойства примитивов
|_ Разное
|_ Не геометрические свойства примитивов
|_ Суммарно
|_ FILTER
|_ Свойства_для_фильтрации
|_ Просуммированные свойства
end legend
```

- a. **Обрабатывать примитивы** - тут можно выбрать тип примитивов свойства которых отображаются в инспекторе, а нажатием кнопок **[\*]** или **[-]** оставить только примитивы данного типа в текущем выборе или исключить их из текущего выбора
- b. **Обработка переменных** - Управляет способом отображения набора переменных примитива (см. [extdrVariables](#) и [Централизация](#)) . **Выбранный примитив** - только переменные выбранных примитивов; **Связанные примитивы** - только переменные связанных примитивов; **Все примитивы** - обрабатываются переменные и примитивов из текущего выбора и связанных с ними примитивов, все переменные отображаются на одной вкладке; **Все примитивы раздельно** - обрабатываются переменные и примитивов из текущего выбора и связанных с ними примитивов, переменные отображаются на разных вкладках;
- c. **Связанные переменные** - на этом уровне структуры будут отображены переменные связанных примитивов, при с значением Process Обработка\_переменных=**Все примитивы раздельно**
- d. **Переменные** - на этом уровне структуры будут отображены все доступные в соответствии с

значением Process variables переменные. При отображении переменных связанных примитивов возможна цветовая раскраска значений !!NEEDLINK!!

- e. **Общие** - тут перечислены общие свойства примитивов: слой, цвет, вес и т.п.
- f. **Расширения** - если к выделенным примитивам добавлены расширения, их свойства отображаются здесь.
- g. **Геометрия** - различные геометрические свойства: точки вставки, координаты, длины и т.п.
- h. **Суммарно** - некоторые свойства допускающие суммирование, например при выборе нескольких отрезков тут будет их суммарная длина. Здесь же в ветке **FILTER** будут некоторые свойства допускающие фильтрацию, например можно кнопкой **[\*]** оставить в текущем выборе только примитивы лежащие на определенном слое.

### 3. Окно сообщений

Тут отображаются различные сообщения по ходу работы программы. Информация о процессах, сообщения о ошибках, предупреждения, подсказки для пользователя и т.п.

### 4. Командная строка

Тут можно ввести имя команды, тем самым запустив ее, либо ввести координату точки на запрос уже выполняемой команды.

Поле ввода имеет подсказку меняющуюся по текущей ситуации. Когда программа ожидает ввода команды подсказка имеет вид **Команда>** и **>** когда ожидается координата. Также некоторые команды имеют контекстную подсказку, в которой можно выбрать мышью опции команды

Перечень доступных команд приведен в **Команды** команда может быть введена как просто по имени, так с операндом. Операнд указывается в скобках после имени команды. Например ввод **Load** вызовет диалог открытия файла чертежа и последующую его загрузку. Ввод **Load(D:\file.dxf)** сразу вызовет загрузку файла **D:\file.dxf**. Парсинг операндов выполняется силами команды, поэтому синтаксис в разных командах отличается.

Координаты можно вводить как 2D, так и 3D, при этом 2D будут переведены в 3D подстановкой 0 в качестве координаты Z. Также можно вводить как абсолютные, так и относительные (относительно последней указанной точки) значения. Относительные координаты задаются указанием знака **@** перед координатой X. Если включен режим трассировки !!NEEDLINK!! и имеется привязка к оси трассировки, можно указать точку введя расстояние от точки трассировки, отложенное по оси трассировки.

Например если в открытом чертеже на запрос **Команда>** ввести **Line**, затем **10,30**, затем **@1,2** будет построена линия с координатами (10,30)-(11,32)

Командная строка может быть отключена. !!NEEDLINK!! В режиме с выключенной командной строкой становятся доступны буквенные сочетания клавиш - с включенной нажатия букв обрабатываются командной строкой

### 5. Статусная строка.

Здесь отображаются координаты курсора и прогрессбары долгих процессов. Также есть быстрый доступ к кнопкам переключения различных режимов работы программы

!!NEEDLINK!!

## 6. Панели инструментов

Здесь сгруппированы иконки различных команд для их быстрого запуска. !!NEEDLINK!!

## 7. Палитры

Могут отображаться в древовидном виде и в виде списка иконок. К каждому листу дерева или иконке списка может быть привязана произвольная команда, запускаемая при клике по элементу. Как правило это команды вставки устройств или блоков. Древовидные палитры дополнительно оснащены полем фильтра для быстрого нахождения элементов !!NEEDLINK!!

## 8. Навигаторы

Навигаторы служат для быстрой навигации по чертежу, нахождению на нем тех или иных элементов. Представляют из себя настраиваемую древовидную структуру отображения данных. На данный момент в программе доступны навигаторы устройств, кабелей, стояков и примитивов !!NEEDLINK!!

### 3.1.2. AnchorDocking

ZCAD в своей работе использует библиотеку [AnchorDocking](#) !!NEEDLINK!! данная библиотека позволяет склеить (пристыковывать) различные окна в одно. На [Внешний вид окна программы](#) 1,2,3,7,8 являются отдельными окнами склеенными в одно окно. Пустое окно программы выглядит следующим образом:

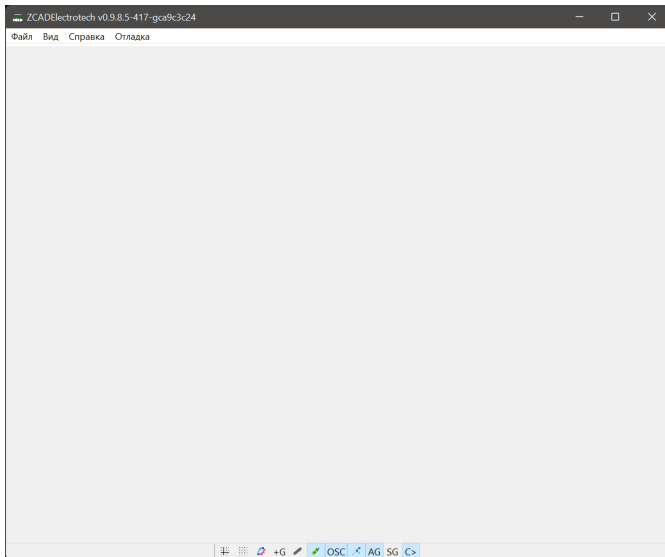


Рисунок 4. Пустое окно программы

По периметру окна расположены области для пристыковки панелей инструментов, в центре область пристыковки окон.

## Панели инструментов

Включить панель инструментов можно командой [ShowToolBar](#) передав ей в качестве операнда имя панели инструментов (или в меню [Вид/Показать окно/Панели инструментов](#)). Например включаем панель [View](#), по умолчанию она отображается в непристыкованном

СОСТОЯНИИ:

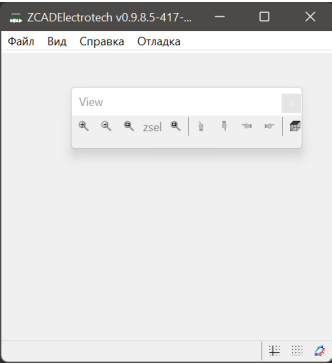


Рисунок 5. Панель инструментов

Для пристыковки панели начинаем ее перетаскивать за свободное место (не за заголовок!), при этом будет подсвечиваться предполагаемое новое место расположения панели. При подведении мышки к краю окна будет подсвечена область вдоль края, при отпускании кнопки мыши панель инструментов будет пристыкована .Стыковка панели инструментов

Действие	Процесс	Результат
Стыковка панели инструментов	A screenshot of the ZCADElectrotech v0.9.8.5-417... application window. The 'View' menu is open, and the 'zsel' toolbar is being moved from its original position to a new location, indicated by a blue rectangular highlight.	A screenshot of the ZCADElectrotech v0.9.8.5-417... application window. The 'View' menu is open, and the 'zsel' toolbar is now docked to the right side of the window, integrated into the main interface.

В пристыкованном состоянии панели инструментов имеют заголовок в виде двух полосок в начале панели, за этот заголовок панель можно передвигать вдоль края пристыковки и отстыковывать, двойной клик по заголовку панели приведет к ее полному раскрытию. Закрывать панель инструментов можно только в отстыкованном состоянии

Окна

Включить окно программы можно командой **Show** передав ей в качестве операнда имя окна (или в меню **Вид/Показать окно**). Например включаем окно **ObjectInspector**, по умолчанию окно отображается в непристыкованном состоянии:

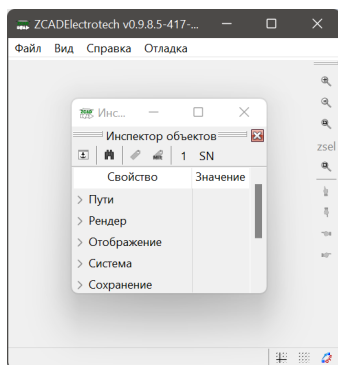
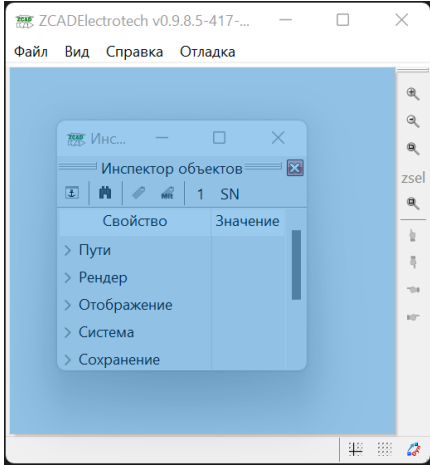
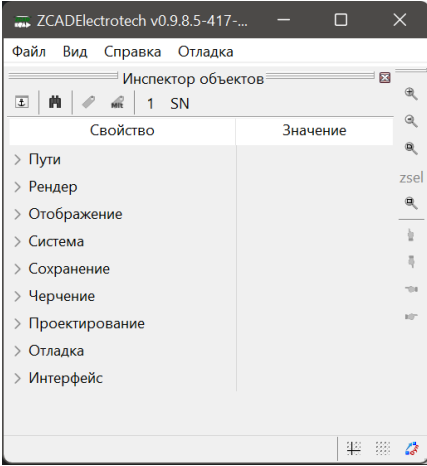
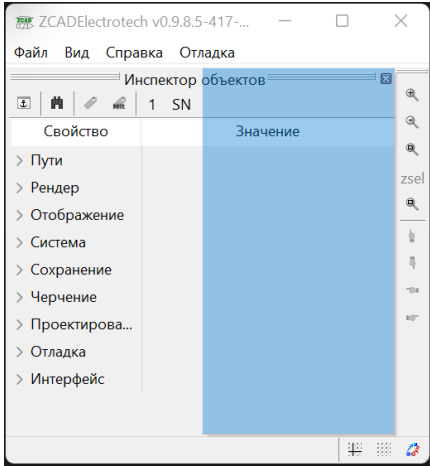
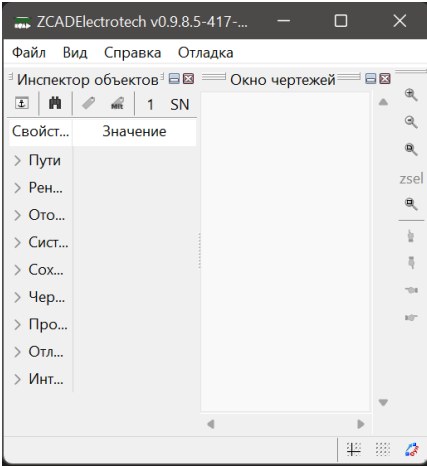
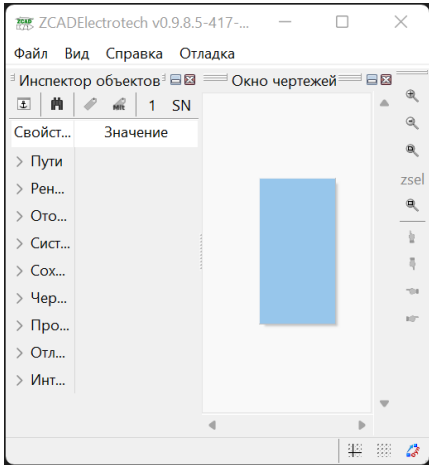
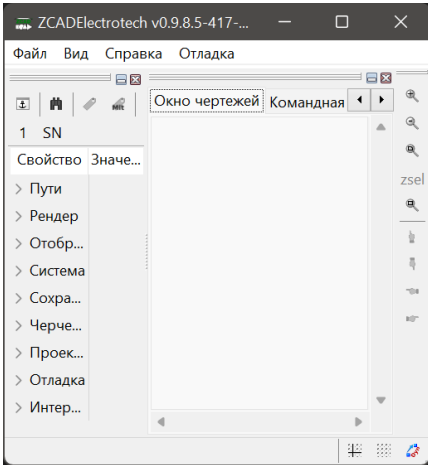


Рисунок 6. Не пристыкованное окно инспектора объектов

Не пристыкованные окна программы имеют два заголовка - стандартный и нестандартный. За стандартный окна можно только перемещать, за нестандартный - перемещать и пристыковывать. Стыковка окон осуществляется перетаскиванием окна на желаемое место стыковки, при этом область стыковки будет подсвечена.

Таблица 1. Варианты стыковки окон программы

Действие	Процесс	Результат
Стыковка первого окна		
Стыковка следующего окна справа от первого		

Действие	Процесс	Результат
Стыковка следующего окна поверх второго		

В таблице приведены возможные варианты стыковки окон. Первое окно может быть пристыковано только в центр главного окна. Последующие окна могут быть пристыкованы с любой стороны от уже имеющихся, либо поверх них, при этом будут переключаться вкладки, как показано в третьей строке таблицы. При стыковке окон слева\справа\сверху\снизу между ними появляется разделительный сплитер которым можно регулировать размер окон.

В заголовке пристыкованных окон появляется дополнительная кнопка минимизации, которая позволяет свернуть окно в тонкий заголовок и разворачивать его при наведении мыши

При щелчке правой кнопкой мыши по любому разделительному сплитеру можно вызвать контекстное меню стыковки для настройки ее параметров.

Сохранить раскладку окон и тулбаров можно командой **SaveLayout** или в меню **настройка\Сохранить разбивку окон по умолчанию**

### 3.1.3. Навигаторы

Отдельно стоит рассмотреть элемент интерфейса - навигаторы. Навигатор представляет из себя полностью настраиваемую древовидную структуру отображающую определенные свойства определенных примитивов чертежа. Каждая строка в структуре навигатора - отдельный примитив. На данный момент доступны навигаторы устройств, кабелей и стояков

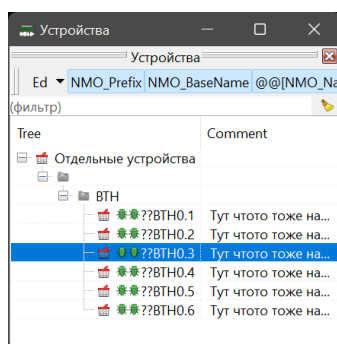


Рисунок 7. Навигатор устройств

Настройка навигатора производится следующим образом:

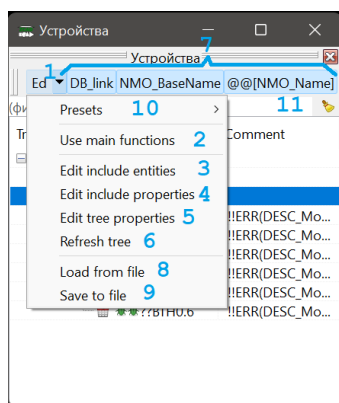


Рисунок 8. Органы управления навигатора

#### 1. Кнопка настройки ветвлений древовидной структуры

Вызывает редактор описания ветвлений древовидной структуры навигатора, либо открывает меню действий с навигатором при нажатии на стрелку

#### 2. Кнопка главной функции

Включает-выключает использование в навигаторе "Главной функции" (Централизация)

#### 3. Кнопка редактора фильтра примитивов по типу

Вызывает редактор скрипта фильтрации по типу примитивов см. [Формат фильтра по типам примитивов](#)

#### 4. Кнопка редактора фильтра примитивов по свойствам

Вызывает редактор скрипта фильтрации по свойствам примитивов см. [Формат фильтра по свойствам примитивов](#)

#### 5. Кнопка редактора скрипта настройки отображения дерева

Вызывает редактор скрипта настройки отображения дерева. Ветвление дерева зависит от ([1. Кнопка настройки ветвлений древовидной структуры](#)), в данном скрипте настраивается например количество столбцов и их заполнение. Данный скрипт выполняется при создании навигатора, при изменении скрипта выполняется пересоздание навигатора для применения изменений В общем случае скрипт выглядит так:

```
Оператор(Операнд[ ,Операнд,Операнд ...])
```

Доступные типы операндов:

'Строка'	Строковой параметр - любой текст заключенный в апострофы
Целое положительное число	

Операторы могут разделяться точкой с запятой, запятой, пробелом, переводом строки. Доступны следующие операторы:

SetColumnsCount(Операнд1,Операнд2)	<p>Задаёт количество столбцов в навигаторе, должна присутствовать 1 раз в начале скрипта</p> <p><b>Операнд1</b> - Целое положительное число. Количество столбцов в навигаторе</p> <p><b>Операнд2</b> - Целое положительное число. Номер столбца чья ширина будет рассчитываться автоматически (нумерация с 0)</p>
SetColumnParams(Операнд1,Операнд2,Операнд3,Операнд4,Операнд5)	<p>Задаёт параметры столбцов, должна присутствовать для каждого столбца</p> <p><b>Операнд1</b> - Целое положительное число. Номер столбца для которого задатются параметры (нумерация с 0)</p> <p><b>Операнд2</b> - Строка. Название столбца</p> <p><b>Операнд3</b> - Строка. Содержимое столбца</p> <p><b>Операнд4</b> - Строка. Имя переменной в которой сохраняется ширина столбца (в файле <code>rtl\savevar.pas</code>) между сессиями работы. Если переменной пока нет, она будет создана с начальным значением 50</p> <p><b>Операнд5</b> - Целое положительное число. Пока не используется</p>

Примеры скриптов экспорта:

Два столбца Tree и Comment, содержимое переменной NMO\_Name примитива в первом и статичная надпись 'Тут что то тоже надо сделать' во втором

```
SetColumnsCount(2,0);
SetColumnParams(0,'Tree','@[NMO_Name]','tmpGUIParamSave_NavDev_C0',1);
SetColumnParams(1,'Comment','Тут что то тоже надо
сделать','tmpGUIParamSave_NavDev_C1',1)
```

Три столбца Tree Elevation и Text, с содержимым переменных RiserName, Elevation и Text примитивов

```
SetColumnsCount(3,0);
SetColumnParams(0,'Tree','@[RiserName]','tmpGUIParamSave_NavRis_C0',1);
SetColumnParams(1,'Elevation','@[Elevation]','tmpGUIParamSave_NavRis_C1',1);
SetColumnParams(2,'Text','@[Text]','tmpGUIParamSave_NavRis_C2',1)
```

## 6. Кнопка обновления

Вызывает перестроение дерева в навигаторе. Обычно при изменениях на чертеже навигаторы обновляются автоматически, но могут быть ситуации когда требуется в ручную вызвать обновление навигатора

## 7. Кнопки управления ветвлением

Количество и название кнопок зависит от (1. [Кнопка настройки ветвлений древовидной структуры](#)) нажатое или отжатое состояние показывает включен или нет данный узел в



текущий момент. Узлы можно включать\выключать.

#### 8. Сохранение текущих настроек навигатора в файл

Открывает окно выбора файла

#### 9. Загрузка настроек навигатора из файла

Открывает окно выбора файла

#### 10. Подменю выбора вариантов настроек навигатора

Подменю выбора заранее подготовленных вариантов настроек навигатора. Для того чтобы вариант появился в данном подменю настройку необходимо сохранить в папку \$(ZCADPath)/configs

#### 11. Быстрый фильтр содержимого навигатора

Поле ввода для быстрой фильтрации содержимого инспектора по тексту. 5Допускает применение символов ? и \*

## 3.2. Примитивы

Основной формат файла хранения графических данных программы – DXF версии 2000, со следующими ограничениями:

- Не поддерживается THICKNESS
- Не поддерживаются листы, только МОДЕЛЬ

Перечень доступных примитивов:

Таблица 2. Примитивы DXF поддерживаемые программой

Примитив	ZCAD имя	DXF имя	Предоста вляющая версия	Ограничения
Точка	Point	POINT	ZCAD	
Линия	Line	LINE	ZCAD	
3D Полилиния	3DPolyLine	POLYLINE	ZCAD	Дуговые сегменты, тип линии
Полилиния	LWPolyline	LWPOLYLINE	ZCAD	Сглаживание
Сплайн	Spline	SPLINE	ZCAD	Экспериментально
3D Фэйс	3DFace	3DFACE	ZCAD	
Солид	Solid	SOLID	ZCAD	
Штриховка	Hatch	HATCH	ZCAD	Дуговые сегменты, сплайны, ассоциативность
Дуга	Arc	ARC	ZCAD	

Примитив	ZCAD имя	DXF имя	Предоставляющая версия	Ограничения
Окружность	Circle	CIRCLE	ZCAD	
Эллипс	Ellipse	ELLIPSE	ZCAD	
Текст	Text	TEXT	ZCAD	
МТекст	MText	MTEXT	ZCAD	Коды форматирования
Вставка блока	BlockInsert	INSERT	ZCAD	
Устройство	Device		ZCAD	
Выровненный размер	AlignedDimension	DIMENSION	ZCAD	Экспериментально
Повернутый размер	RotatedDimension	DIMENSION	ZCAD	Экспериментально
Диаметральный размер	DiametricDimension	DIMENSION	ZCAD	Экспериментально
Радиальный размер	RadialDimension	DIMENSION	ZCAD	Экспериментально
Кабель	Cable		ZCADElectroTech	
Электрическая выноска	Leader		ZCADElectroTech	
Трасса	Net		ZCADElectroTech	
Суперлиния	SuperLine		ZCADElectroTech	Экспериментально

Данный список будет расширен, но не до полного охвата примитивов DXF.



Файлы, обработанные ZCAD, можно редактировать в AutoCAD (и других CAD программах) не применяя команды, очищающие расширенные данные примитивов. Если расширенные данные примитива будут разрушены, при последующей обработке файла ZCAD, он будет воспринят как стандартный примитив DXF, т.е. кабель станет просто полилинией, а устройство обычным блоком

### 3.2.1. Device

Условное графическое обозначение (далее УГО) оборудования на чертеже - это примитив **DEVICE** (устройство) в терминах ZCAD. В отличие от стандартного примитива DXF – **INSERT** (вставка блока) содержит в себе как жестко определенные в описании блока примитивы, так и динамические, которые можно двигать относительно точки вставки. Динамические примитивы уникальны для каждой вставки устройства, их количество и конфигурация

могут отличаться и настраиваться. Все имена описаний устройств начинаются с приставки **DEVICE\_**, при вставке блока с названием, начинающимся с **DEVICE\_**, он автоматически будет конвертирован в примитив **DEVICE**

Например, чтобы вставить на план дымовой пожарный извещатель как устройство, нужно вставить блок **DEVICE\_PS\_DAT\_SMOKE**. Если вставить просто блок **PS\_DAT\_SMOKE**, он не будет конвертирован в примитив **DEVICE** и останется примитивом DXF – **INSERT** (вставка блока), не приобретая свойств устройства.

Устройство **DEVICE\_PS\_DAT\_SMOKE** состоит из 2х блоков:

- **DEVICE\_PS\_DAT\_SMOKE** – динамическая часть устройства
- **PS\_DAT\_SMOKE** постоянная часть устройства

Постоянная часть устройства содержит в себе собственно УГО:

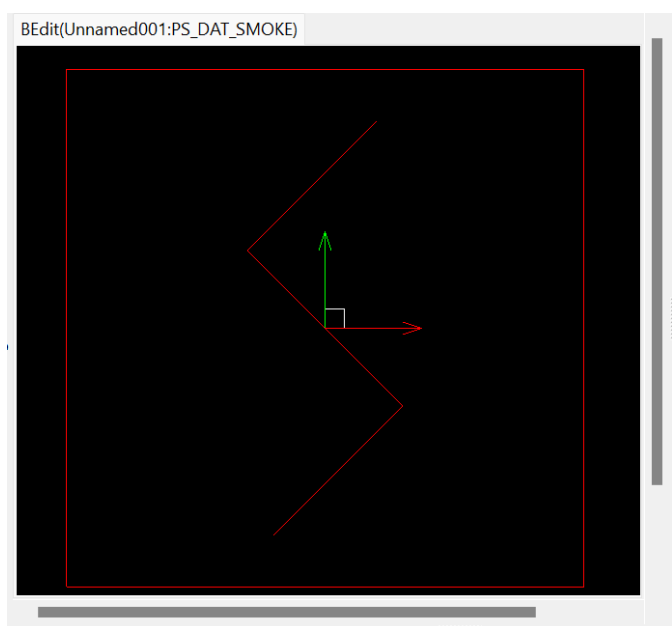


Рисунок 9. Постоянная часть устройства

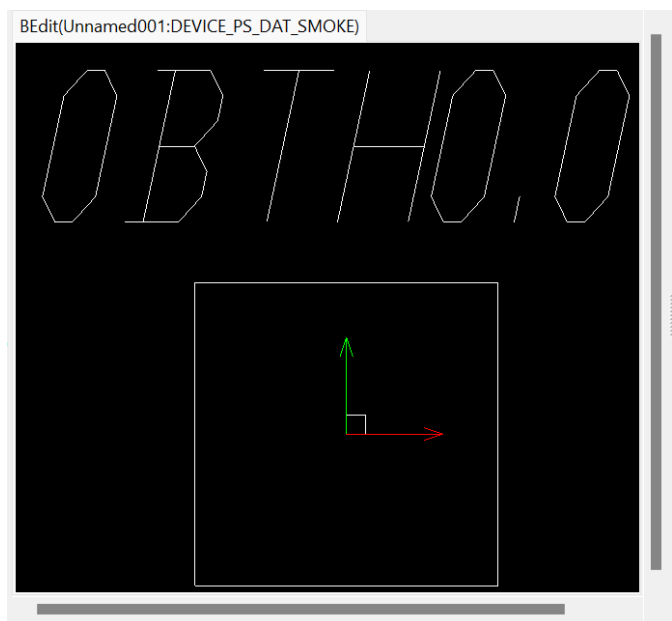


Рисунок 10. Динамическая часть устройства

Динамическая часть устройства содержит дополнительные примитивы например в данном случае:

- Обозначение имени устройства
- контур подрезки присоединенных кабелей
- коннектор для присоединения кабелей (коннекторов может быть несколько, это самостоятельное устройство).

Как создать новое устройство показано на данном видео:

► <https://www.youtube.com/watch?v=zUKcJVM55fQ> (YouTube video)

## 3.3. Расширения примитивов

Для расширения функционала примитивов ZCAD предусматривает механизм расширений. К любому примитиву может быть привязано расширение добавляющее некоторую функциональность. Также расширения могут быть привязаны к типу примитивов, например к примитиву **DEVICE** привязано расширение **extdrVariables** Для работы с расширениями предусмотрены следующие команды:

- **extdrAdd** - привязать расширение к примитиву/примитивам
- **extdrRemove** - отвязать расширение от примитив/примитивов
- **extdrAllList** - вывести список всех доступных расширений
- **extdrEntsList** - вывести список расширений привязанных к примитиву/примитивам

### 3.3.1. extdrLayerControl

Управление слоем применяется для управления видимостью примитивов в динамической части устройств. У примитива с данным расширением появляются дополнительные параметры

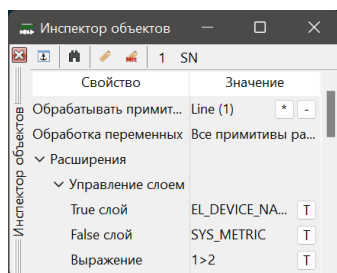


Рисунок 11. Параметры расширения *extdrLayerControl*

- **True слой** - Имя слоя при **True** результате выражения
- **False слой** - Имя слоя при **False** результате выражения
- **Выражение** - Выражение имеющее Boolean результат

Выражение может состоять из простых математических функция и включать имена переменных. При вычислении переменные ищутся в расширении **extdrVariables** данного примитива, при отсутствии запрашиваются у **централи** (см. раздел **Централизация**) и далее

у владельца примитива по иерархии.

При ошибке в выражении или отсутствии требуемого слоя в чертеже текущий слой примитива не меняется.

### 3.3.2. extdrSmartTextEnt

Управление текстовыми примитивами для управления параметрами текстов в динамической части устройств. У примитива с данным расширением появляются дополнительные параметры

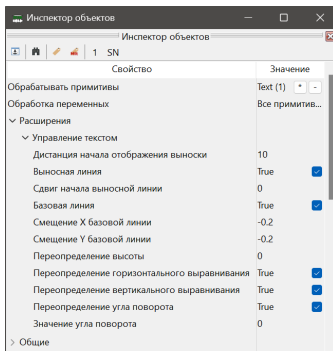


Рисунок 12. Параметры расширения extdrSmartTextEnt

- **Дистанция начала отображения выноски** - После удаления текста от точки вставки устройства на данную дистанцию к тексту начинает пририсовываться выноска
- **Выносная линия** - Отображение выносной линии
- **Сдвиг начала выносной линии** - Сдвиг начала выносной линии от точки вставки устройства. Положительное значение - радиус круга, отрицательное - половина стороны квадрата
- **Базовая линия** - Отображение базовой линии
- **Смещение X базовой линии** - Смещение X координаты начала базовой линии от точки вставки текста. Положительное значение - абсолютное смещение, отрицательное - относительное смещение, домножается на высоту текста
- **Смещение Y базовой линии** - Смещение Y координаты начала базовой линии от точки вставки текста. Положительное значение - абсолютное смещение, отрицательное - относительное смещение, домножается на высоту текста
- **Переопределение высоты** - Если не нулевое значение, высота текста всегда будет равна этой величине, независимо от масштаба устройства
- **Переопределение горизонтального выравнивания** - Разрешение переопределения горизонтальной составляющей выравнивания текста в зависимости от точки вставки текста относительно устройства
- **Переопределение вертикального выравнивания** - Разрешение переопределения вертикальной составляющей выравнивания текста в зависимости от точки вставки текста относительно устройства
- **Переопределение угла поворота** - Разрешение переопределения угла поворота текста
- **Значение угла поворота** - Абсолютное значение угла поворота

### 3.3.3. extdrVariables

Представляет собой хранилище дополнительных типизированных данных (**Переменных**) привязанных к примитиву. Каждая переменная имеет имя, тип и "пользовательское" имя. \* **Имя** - идентификатор состоящий из латинских букв и цифр, начинается с буквы. По имени программа обращается к переменным и различает их. Имя регистронезависимо \* **Тип** - определяет содержимое переменной, может быть как простым ([Перечень простых типов данных](#)), так и сложным. сложные типы являются комбинацией простых и определены в файле `rtl\system.pas` на паскалеподобном скриптовом языке \* **Пользовательское имя** - имя переменной для пользователя, пользователь видит его например в инспекторе объектов, может быть любым

#### Перечень простых типов данных

Double	Число двойной точности
Single	Число одинарной точности
UnicodeString	Строка UTF16
String	Платформозависимо. UTF8/16 На данный момент строка UTF8, возможно станет UTF16
AnsiString	Строка UTF8
Boolean	False/True
Byte	8 бит, без знака
ShortInt	8 бит, с знаком
SmallInt	16 бит, с знаком
LongInt	32 бит, с знаком
Int64	64 бит, с знаком
Cardinal	Алиас к LongWord
Integer	Платформозависимо. 64/32/16 бит, с знаком
PtrInt	Размерность указателя, с знаком
Word	16 бит, без знака
LongWord	32 бит, без знака
QWord	64 бит, без знака
DWord	Алиас к LongWord
UInt64	Алиас к QWord
Pointer	Платформозависимо. Указатель 64/32/16 бит, с знаком

Набор и значения переменных у примитивов можно редактировать командами [VarsEd](#) и [VarsEdSel](#) Значения переменных доступны для изменения в инспекторе объектов (см. раздел [2. Инспектор объектов](#)) при выделении примитивов. Значения переменных можно экспортировать-импортировать командами [DataExport](#) и [DataImport](#)

Примитивы в чертеже имеют древовидную структуру, так все примитивы являются дочерними объектами чертежа, примитивы в динамической части устройства это дочерние объекты этого устройства и т.д. При поиске у примитива какойлибо переменной и не нахождении ее зкад обращается к родительскому примитиву и ищет ее там, так пока не доййдет до вершины дерева - чертежа !!NEEDLINK!!

## Централизация

Дополнительно к древовидной иерархии примитивов расширения переменных у разных примитивов могут ссылаться друг на друга. Этот механизм называется **централизация**


Например при выполнении схемы автоматизации и плана расположения оборудования можно примитивы **DEVICE** обозначающие один датчик на схеме и на плане связать между собой. Это позволит хранить характеризующий этот датчик набор переменных в одном примитиве, но иметь к ним доступ из другого.

При таком объединении один примитив в связке выбирается главным (**центрль**), другие сылаются на него (**представители**) В идеальном случае все переменные связки хранятся в централи, у представителей нет своих переменных, они лишь предоставляют доступ к централи, при такой организации у пользователя не возникает путаницы с фактическим местом нахождения переменной. Но также возможно хранить переменные и в представителях.

Создать централизованные примитивы можно командой [VarsLink](#)


## 3.4. Команды

### 3.4.1. 3DPoly

Команда:	<a href="#">3DPoly</a>
Иконка:	
Меню:	<a href="#">Черчение &gt; 3D Полилиния</a>

Черчение примитива **3DPolyLine**. После запуска команды последовательно указать мышью вершины полилинии. Завершение черчения - ESC или запуск другой команды

### 3.4.2. About

Команда:	<a href="#">About</a>
Иконка:	
Меню:	<a href="#">Справка &gt; О программе</a>

Показ окна "О программе"


AddToOwner

### 3.4.3. Arc

Команда:	<b>Arc</b>
Иконка:	
Меню:	<b>Черчение &gt; Дуга</b>

Черчение примитива **Arc** (дуга) по 3м точкам. После запуска команды последовательно указать 3 точки дуги

### 3.4.4. BEdit

Команда:	<b>BEdit</b>
Иконка:	
Меню:	<b>Изменить &gt; Блок/Устройство &gt; Редактор блоков</b>
Предвыбор примитивов:	Не требуется, но можно заранее выделить вставку редактируемого блока или устройства

Редактирование определения блока из чертежа. В случае если до запуска команды выбрана вставка блока, редактор переходит в режим редактирования определения этого блока. Если выделение отсутствует, команда позволяет выбрать редактируемое определение в инспекторе объектов.

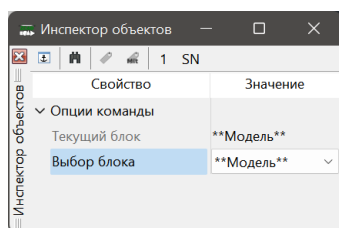


Рисунок 13. Опции команды BEdit

**Текущий блок** тут показан текущий редактируемый блок или **Модель** если редактирование блока не осуществляется, в окне чертежа отображено пространство модели. В поле **Выбор блока** можно выбрать блок для редактирования. При этом в окне чертежа откроется определение выбранного блока, а название вкладки чертежа поменяется на **BEdit(ИмяЧертежа:ИмяБлока)** После завершения редактирования повторный запуск **BEdit** вернет редактор в модель. Все внесенные изменения будут сохранены в определении блока без запроса на сохранение. Для обновления вставок блока на чертеже требуется выполнить регенерацию чертежа **Regen** Если редактировалась динамическая часть определения устройств, обновить ее для уже вставленных устройств возможно только повторной вставкой **BlockReplace** (заменить вставленные устройства на самих себя)



При переключении из пространства модели в пространство определения блока и обратно положение камеры не меняется, возможна ситуация когда после переключения в окне чертежа ничего не видно. Исправить ситуацию можно командой **Zoom** с опцией **All** - **Zoom(All)**



### 3.4.5. Cam\_reset

Команда:	<b>Cam_reset</b>
Меню:	<b>Вид &gt; Камера в начало</b>

Сброс текущих настроек видового окна чертежа (положения камеры), после выполнения команды в центре видового окна отображается начало координат, направление взгляда совпадает с осью Z


### 3.4.6. Cancel

Команда:	<b>Cancel</b>
----------	---------------

Прерывает текущую выполняемую команду

ChangeProjType

### 3.4.7. Circle

Команда:	<b>Circle</b>
Иконка:	
Меню:	<b>Черчение &gt; Окружность &gt;</b>
Операнды:	CR, CD, 2P, 3P. Не обязательный (CR используется в случае не указания)
Пример использования:	<b>Circle</b>
Пример использования 2:	<b>Circle(3P)</b>

Черчение примитива **Circle**. После запуска команды последовательно задать точки на окружности, в зависимости от операнда команды.


- **CR** - указывается центр и радиус окружности
- **CD** - указывается центр и диаметр окружности
- **2P** - указывается центр 2 точки на окружности (лежащие на концах диаметра)
- **3P** - указывается центр 3 точки на окружности

Circle2 ClearFileHistory Colors

Connection2Dot


### 3.4.8. Copy

Команда:	<b>Copy</b>
----------	-------------

<b>Иконка:</b>	
<b>Сочетания клавиш:</b>	Ctrl + ALT + C; C
<b>Предвыбор примитивов:</b>	Требуется


Копирование выбранных примитивов. Исходные примитивы нужно выбрать до запуска команды. После запуска требуется базовую точку для копирования и базовые точки для вставки. Завершение копирования - ESC или запуск другой команды

### 3.4.9. CopyBase

<b>Команда:</b>	<i>CopyBase</i>
<b>Иконка:</b>	
<b>Сочетания клавиш:</b>	Ctrl + Shift + C
<b>Меню:</b>	<i>Правка &gt; Копировать в буфер обмена с базовой точкой</i>
<b>Предвыбор примитивов:</b>	Требуется

Копирует выбранные примитивы в буфер обмена, перед копированием нужно указать базовую точку


### 3.4.10. CopyClip

<b>Команда:</b>	<i>CopyClip</i>
<b>Иконка:</b>	
<b>Сочетания клавиш:</b>	Ctrl + C; Ctrl + INS
<b>Меню:</b>	<i>Правка &gt; Копировать в буфер обмена</i>
<b>Предвыбор примитивов:</b>	Требуется

Копирует выбранные примитивы в буфер обмена

CopyFromOwner

### 3.4.11. CutClip

<b>Команда:</b>	<i>CutClip</i>
<b>Иконка:</b>	
<b>Сочетания клавиш:</b>	Ctrl + X
<b>Меню:</b>	<i>Правка &gt; Вырезать в буфер обмена</i>
<b>Предвыбор примитивов:</b>	Требуется

Копирует выбранные примитивы в буфер обмена и стирает их из чертежа

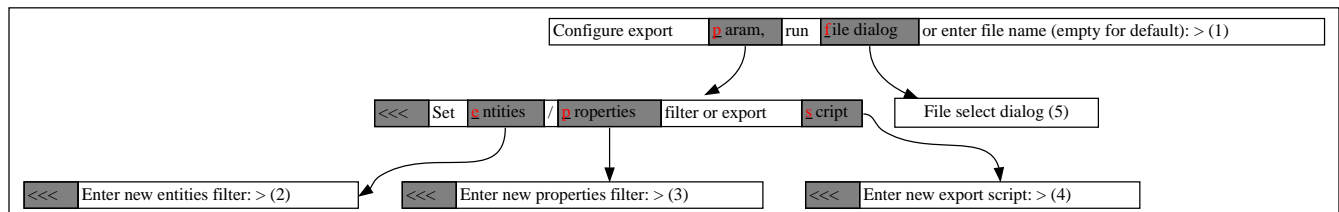
DBaseAdd DBaseLink DBaseRename

### 3.4.12. DataExport

Команда:	<b>DataExport</b>
----------	-------------------

Команда экспорта параметров графических примитивов в внешний файл CSV. Команда применяет к всем примитивам текущего чертежа фильтр типа примитивов, передавая на дальнейшую обработку только примитивы определенного типа. Далее применяется фильтр свойств примитивов, оставляя только примитивы имеющие требуемые свойства. Над прошедшими фильтрацию примитивами выполняется скрипт экспорта записывая требуемые параметры во внешний файл

При запуске подсказка командной строки принимает вид:



Что позволяет выполнить в командной строке следующие действия:

#### 1. Задать имя файла с экспортируемыми значениями

Ожидание ввода имени файла или пустой строки. Можно ввести имя файла, введенное значение будет использовано в дальнейшем как файл по умолчанию. Пустой ввод не изменяет текущее значение файла по умолчанию. После ввода значения экспорт будет выполнен и команда завершится

#### 2. Задать фильтр примитивов

Ожидание ввода значения фильтра примитивов по типу. см. [Формат фильтра по типам примитивов](#)

#### 3. Задать фильтр свойств примитивов

Ожидание ввода значения фильтра примитивов по параметрам. см. [Формат фильтра по свойствам примитивов](#)

#### 4. Задать скрипт экспорта

Ожидание ввода скрипта выполняющего экспорт. см. [Формат скрипта экспорта](#)

#### 5. Задать имя файла в диалоговом режиме

Откроет диалог выбора файла, можно выбрать любой файл csv. Выбор файла сделает его файлом по умолчанию и также как (1) выполнит экспорт и завершит команду

Также все это можно сделать с помощью инспектора объектов, который при выполнении команды имеет вид:

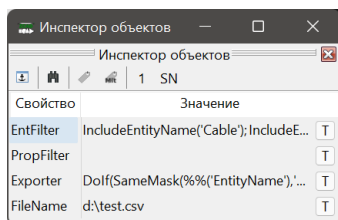


Рисунок 14. Опции команды DataExport

## Формат фильтра по типам примитивов

Данный фильтр позволяет включить в набор или исключить из набора для экспорта определенные типы примитивов и содержащие определенные расширения. Включать/исключать типы примитивов/расширения можно как по имени, так и по маске. Изначально в набор для обработки не включены никакие примитивы. Для добавления тип примитива/расширение должен быть добавлен и не должен быть исключен. Причем для примитивов с несколькими расширениями достаточно попадания в разрешенный список хотябы одного расширения. Если не задано не одного условия для расширений, включение\выключение в список происходит только по типу примитива. Фильтр задается в текстовом виде и в общем случае выглядит так:

```
Оператор(Операнд)[;Оператор(Операнд)]
```

Операторы могут разделяться точкой с запятой, запятой, пробелом, переводом строки. Доступны следующие операторы:

IncludeEntityMask('Mask*')	Включить типы примитивов по маске
IncludeEntityName('Name')	Включить тип примитивов по имени
ExcludeEntityMask('Mask*')	Исключить типы примитивов по маске
ExcludeEntityName('Name')	Исключить тип примитивов по имени
IncludeExtenderMask('Mask*')	Включить примитив с расширением по маске
IncludeExtenderName('Name')	Включить примитив с расширением по имени
ExcludeExtenderMask('Mask*')	Исключить примитив с расширением по маске
ExcludeExtenderName('Name')	Исключить примитив с расширением по имени

Все операторы в качестве операнда принимают строковой параметр заключенный в

апострофы.

Примеры фильтров примитивов:

Включить в экспорт примитивы **Cable** и **Device**:

```
IncludeEntityName('Cable');  
IncludeEntityName('Device')
```

Включить в экспорт все типы примитивов:

```
IncludeEntityMask('*')
```

Включить в экспорт примитивы **PolyLine** и **Line**:

```
IncludeEntityName('*Line') ExcludeEntityName('3DPolyLine')
```

### Формат фильтра по свойствам примитивов

Данный фильтр применяется последовательно к каждому примитиву прошедшему фильтрацию по типу. Фильтр задается в текстовом виде и в общем случае выглядит так:

```
Оператор(Операнд[,Операнд])[;Оператор(Операнд[,Операнд])]
```

Операторы могут разделяться точкой с запятой, запятой, пробелом, переводом строки. Доступны следующие операторы:

IncludeIfMask(Операнд, 'Маска*')	Включить примитив в результат если операнд соответствует маске
IncludeIfSame(Условие)	Включить примитив в результат в случае выполнения условия

Условный операторы пока доступен только один:

SameMask(Операнд1, 'Маска')	Условие считается выполненным если текстовое значение операнда1 соответствует маске
Or(Условие1,Условие2)	Объединение условий 1 и 2 по закону ИЛИ

Доступные типы операндов:

'Строка'	Строковый параметр - любой текст заключенный в апострофы
----------	--

<code>%%('ИмяПараметра')</code>	Возвращает значение параметра текущего примитива в виде строки
---------------------------------	--

Имена параметров для разных примитивов могут отличаться, в общем случае это параметры доступные в инспекторе объектов для данного типа примитивов, в случае отсутствия у примитива данного параметра возвращается пустая строка. Примерный перечень доступных параметров смотри в [https://github.com/zamtmn/zcad/blob/master/cad\\_source/zcad/gui/objectinspector/uzcoiregistermultiproperties.pas](https://github.com/zamtmn/zcad/blob/master/cad_source/zcad/gui/objectinspector/uzcoiregistermultiproperties.pas)

Примеры фильтров по свойствам примитивов:

Включить в результат только примитивы типа кабель, или примитивы с параметром 'Name' удовлетворяющим маске 'CABLE\_\*' (это блоки\устройства, т.к. только у них есть имя)

```
IncludeIfSame(
    Or(SameMask(%%('Name'),'CABLE_*'),
      SameMask(%%('EntityName'),'Cable'))
)
```

Включить в результат только примитивы с параметром 'Name' удовлетворяющим маске 'EL\_CABLE\_\*' (это блоки\устройства, т.к. только у них есть имя)

```
IncludeIfMask(%%('Name'),'EL_CABLE_*')
```

## Формат скрипта экспорта

Данный скрипт применяется последовательно к каждому примитиву прошедшему фильтрацию предыдущими двумя. Скрипт задается в текстовом виде и в общем случае выглядит так:

```
Оператор(Операнд[,Операнд])[;Оператор(Операнд[,Операнд])]
```

Операторы могут разделяться точкой с запятой, запятой, пробелом, переводом строки. Доступны следующие операторы:

<code>Export(Операнд1[,Операнд2, ...])</code>	записать строку операндов с разделителями в новую строку файла csv
<code>DoIf(Условие,Оператор)</code>	Выполнить оператор в случае выполнения условия

Условный операторы пока доступен только один:

<code>SameMask(Операнд1,'Маска')</code>	Условие считается выполненным если текстовое значение операнда1 соответствует маске
---	---

Доступные типы операндов:

'Строка'	Строковой параметр - любой текст заключенный в апострофы
%( 'ИмяПараметра' )	Возвращает значение параметра текущего примитива в виде строки
@@( 'ИмяПеременной' )	Возвращает значение переменной текущего примитива в виде строки

Имена переменных могут быть любыми, в случае отсутствия у примитива данной переменной возвращается значение '!!ERR(ИмяПеременной)!!'. Имена параметров для разных примитивов могут отличаться, в общем случае это параметры доступные в инспекторе объектов для данного типа примитивов, в случае отсутствия у примитива данного параметра возвращается пустая строка. Примерный перечень доступных параметров смотри в [https://github.com/zamtmn/zcad/blob/master/cad\\_source/zcad/gui/objectinspector/uzcoiregistermultiproperties.pas](https://github.com/zamtmn/zcad/blob/master/cad_source/zcad/gui/objectinspector/uzcoiregistermultiproperties.pas)

#### Примеры скриптов экспорта:

Если текущий примитив устройство, записываем в csv строку 'Device','NMO\_Name',Значение переменной NMO\_Name,'Position',Значение переменной Position

```
DoIf(  
    SameMask(%('EntityName'),'Device'),  
    Export(%('EntityName'),'NMO_Name',@@('NMO_Name'),'Position',@@('Position'))  
)
```

Если текущий примитив устройство, записываем в csv строку 'Device','NMO\_Name',Значение переменной NMO\_Name,'Power',Значение переменной Power

```
DoIf(  
    SameMask(%('EntityName'),'Device'),  
    Export(%('EntityName'),'NMO_Name',@@('NMO_Name'),'Power',@@('Power'))  
)
```

Если текущий примитив кабель, записываем в csv строку 'Cable','NMO\_Name',Значение переменной NMO\_Name,'AmountD',Значение переменной AmountD

```
DoIf(  
    SameMask(%('EntityName'),'Cable'),  
    Export(%('EntityName'),'NMO_Name',@@('NMO_Name'),'AmountD',@@('AmountD'))  
)
```

Если текущий примитив кабель, записываем в csv строку 'Cable','NMO\_Name',Значение

переменной NMO\_Name,'CABLE\_Segment',Значение переменной CABLE\_Segment

```
DoIf(SameMask(%('EntityName'),'Cable'),  
    Export(%('EntityName'),  
        'NMO_Name',@@('NMO_Name'),  
        'CABLE_Segment',@@('CABLE_Segment'))  
    )
```

### 3.4.13. DataImport

Команда:	<b>DataImport</b>
Операнды:	Не обязательный. Путь и имя файла CSV
Пример использования:	<b>DataImport</b>
Пример использования 2:	<b>DataImport(E:\myfile.csv)</b>

Импорт данных примитивов из внешнего файла CSV. При вызове без параметров будет открыто окно выбора файла, после выбора данные будут импортированы из соответствующего файла. Имя файла можно передать параметром, тогда данные из файла будут импортированы сразу, без окна выбора. В составе строки параметра могут использоваться макросы !!NEEDLINK!!.

Разделителем в CSV файле должна быть точка с запятой, файл должен состоять из строк следующей структуры:

ИмяПримитива	Переменная1	Значение1	...	...	ПеременнаяN	ЗначениеN
--------------	-------------	-----------	-----	-----	-------------	-----------

Где **ИмяПримитива** - внутреннее имя примитива ZCAD, например **DEVICE**, **CABLE** и т.д.

Далее попарно идут **Переменная** - имя переменной, **Значение** - значение данной переменной. Причем все переменные кроме последней являются идентифицирующими, последняя переменная - собственно импортируемое значение

Например при импорте следующей строки:

DEVICE	NMO_Name	M1	Position	10	Power	1.5
--------	----------	----	----------	----	-------	-----

Произойдет следующее: с текущего чертежа будут выбраны все устройства (примитивы **DEVICE**) среди них будут отобраны имеющие имя M1 (переменная **NMO\_Name='M1'**) и позицию 10 (переменная **Position=10**) при наличии у отобранных устройств переменной **Power** ей будет присвоено значение 1.5 Если переменной **Power** у устройства нет с ним никаких действий произведено не будет.

dbgAppExplorer   dbgBlocksList   dbgClipboard   dbgCmdList   dbgGetAV   dbgMemSummary  
dbgPlaceAllBlocks DeSelectAll

### 3.4.14. DevDefSync

Команда:	<b>DevDefSync</b>
----------	-------------------



<b>Предвыбор примитивов:</b>	Требуется
------------------------------	-----------

Синхронизация динамической части выбранных на чертеже устройств с определением устройства. Устройство на чертеже может быть изменено. Добавлены\убраны примитивы в динамическую часть, к примитивам в динамической части привязаны различные расширения. Для того чтобы данные изменения были доступны для вновь вставленных устройств необходимо перенести изменения из вставки устройства в определение устройства

DimAligned DimDiameter DimLinear DimRadius DimStyles Dist

### 3.4.15. DockingOptions

<b>Команда:</b>	<b><i>DockingOptions</i></b>
-----------------	------------------------------

Показ модального окна настройки параметров стыковки. Также окно настройки можно вызвать в контекстном меню сплитеров и заголовков окон докинга

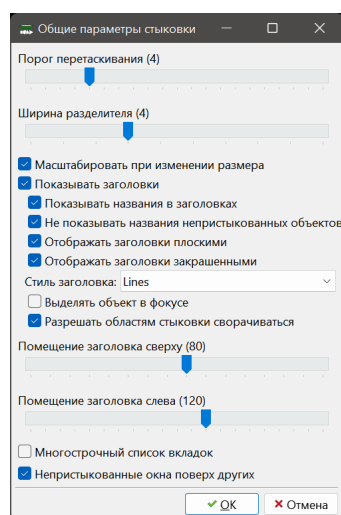


Рисунок 15. Окно настройки параметров стыковки


!!NEEDDETAILS!!

### 3.4.16. DWGClose

<b>Команда:</b>	<b><i>DWGClose</i></b>
<b>Иконка:</b>	
<b>Сочетания клавиш:</b>	<b>Ctrl + F4</b>
<b>Меню:</b>	<b><i>Файл &gt; Заккрыть</i></b>

Заккрыть текущий чертеж. В случае присутствия изменений в чертеже, будет предложено сохранить файл чертежа на диск

### 3.4.17. DWGNew

Команда:	<i>DWGNew</i>
Иконка:	
Сочетания клавиш:	Ctrl + N
Меню:	<i>Файл &gt; Новый чертеж</i>

Создать новый чертеж. !!NEEDEDDETAILS!!

### 3.4.18. DWGNext

Команда:	<i>DWGNext</i>
Сочетания клавиш:	Ctrl + Tab
Меню:	<i>Окно &gt; Следующее окно</i>

Если в программе открыто несколько чертежей текущим становится следующий открытый чертеж

### 3.4.19. DWGPrev

Команда:	<i>DWGPrev</i>
Сочетания клавиш:	Ctrl + Shift + Tab
Меню:	<i>Окно &gt; Предыдущее окно</i>

Если в программе открыто несколько чертежей текущим становится предыдущий открытый чертеж

EL\_AutoGen\_Cable\_Remove EL\_Cable EM\_SEPBUILD EM\_SRBUILD El\_CableMan El\_Cable\_Invert  
El\_Cable\_Join El\_Cable\_Legend El\_Cable\_RenN\_ON El\_Cable\_Select El\_ExternalKZ El\_Find El\_Leader  
El\_Material\_Legend El\_Wire El\_Wire

### 3.4.20. Erase

Команда:	<i>Erase</i>
Иконка:	
Сочетания клавиш:	Del
Меню:	<i>Правка &gt; Стереть</i>
Предвыбор примитивов:	Требуется

Стирает выбранные примитивы на чертеже

ExampleCreateLayer ExampleInsertDevice ExecuteFile ExportDevWithAxis ExportTextToCSV

### 3.4.21. extdrRemove

Команда:	<i>extdrRemove</i>
Предвыбор примитивов:	Требуется
Операнды:	Обязательный. Имя расширения
Пример использования:	<i>extdrRemove(extdrVariables)</i>

Отвязка указанного расширения (см. [Расширения примитивов](#)) от выбранных примитивов. В случае к выбранному примитиву не привязано данное расширение, он будет проигнорирован

### 3.4.22. extdrAllList

Команда:	<i>extdrAllList</i>
----------	---------------------

Команда выводит список доступных для привязки расширений (см. [Расширения примитивов](#))

### 3.4.23. extdrEntsList

Команда:	<i>extdrEntsList</i>
Предвыбор примитивов:	Требуется

Команда выводит список привязанных к выбранным примитивам расширений (см. [Расширения примитивов](#))

### 3.4.24. extdrAdd

Команда:	<i>extdrAdd</i>
Предвыбор примитивов:	Требуется
Операнды:	Обязательный. Имя расширения
Пример использования:	<i>extdrAdd(extdrVariables)</i>

Привязка указанного расширения (см. [Расширения примитивов](#)) к выбранным примитивам. В случае к выбранному примитиву уже привязано данное расширение, он будет проигнорирован

FindAllIntersections Get3DPoint Get3DPoint\_DrawRect GetLength GetPoint GetRect GetVertexX GetVertexY GetVertexZ Help Import Insert Insert2 InsertLayersFromBase InsertTestTable InverseSelected KIP\_CDBuild KIP\_Cable\_Mark KIP\_LugTableBuild

### 3.4.25. Layer

Команда:	<i>Layer</i>
Иконка:	
Меню:	<i>Format &gt; Слой ...</i>

## Показ модального окна управления слоями

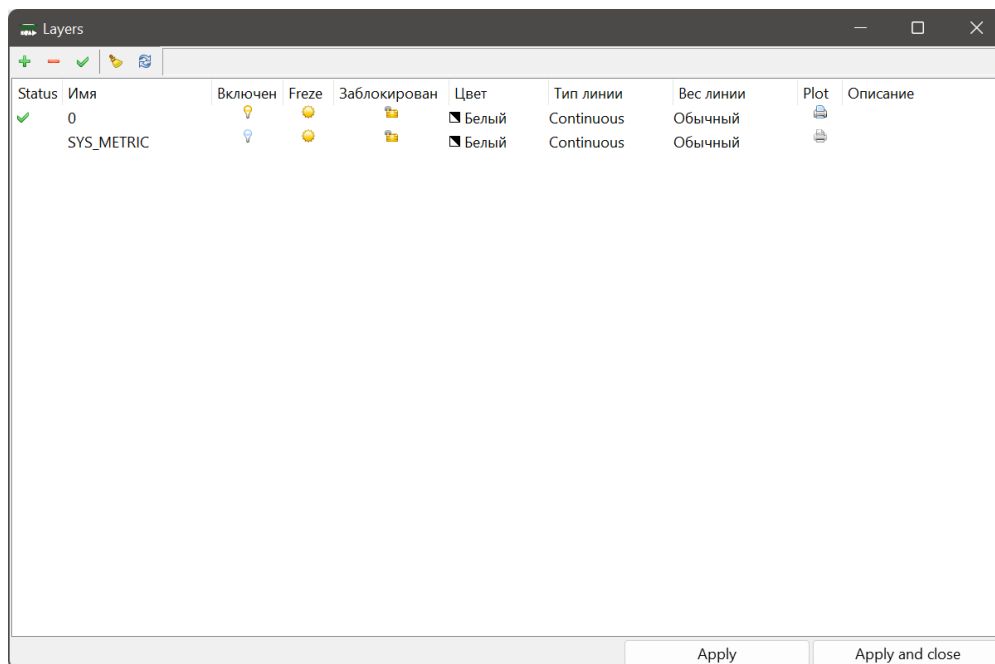


Рисунок 16. Окно управления слоями

!!NEEDEDDETAILS!!

LayerOff LayerOn LayOff Line LineOld LineTypes

### 3.4.26. Load

Команда:	<b>Load</b>
Иконка:	
Сочетания клавиш:	Ctrl + 0
Меню:	<b>Файл &gt; Открыть</b>
Операнды:	Не обязательный. Путь и имя файла dxf
Пример использования:	<b>Load</b>
Пример использования 2:	<b>Load(E:\myfile.dxf)</b>

Загрузка файла DXF. При вызове без параметров будет открыто окно выбора файла, после выбора будет открыт данный файл. Имя файла можно передать параметром, тогда файл будет открыт сразу, без окна выбора. В составе строки параметра могут использоваться [Макросы сокращений путей](#), например команда **Load\$(LastAutoSaveFile))** приведет к загрузке последнего файла автосохранения

### 3.4.27. LoadActions

Команда:	<b>LoadActions</b>
Операнды:	Обязательный. Путь и имя файла с описанием экшенов
Пример использования:	<b>LoadActions(E:\actionscontent.xml)</b>

Загрузка файла описаний экшенов !!NEEDLINK!!. Загрузка обычно происходит на ранней стадии запуска программы, поэтому команда обычно используется в скрипте `$(ZCADPath)/preload/stage0.cmd0 !!NEEDLINK!!`.

### 3.4.28. LoadLayout

<b>Команда:</b>	<b><i>LoadLayout</i></b>
<b>Операнды:</b>	Не обязательный. Имя файла раскладки окон и тулбаров
<b>Пример использования:</b>	<b><i>LoadLayout</i></b>
<b>Пример использования 2:</b>	<b><i>LoadLayout(defaultlayout.xml)</i></b>

Загрузка и применение файла раскладки окон !!NEEDLINK!!. При отсутствии операнда будет произведена загрузка файла раскладки по умолчанию !!NEEDLINK!!. При отсутствии файла раскладки указанного по умолчанию будет загружен файл **defaultlayout.xml**. Файлы раскладки ищутся в папке `$(ZCADPath)/components`.

### 3.4.29. LoadMenus

<b>Команда:</b>	<b><i>LoadMenus</i></b>
<b>Операнды:</b>	Обязательный. Путь и имя файла с описанием меню
<b>Пример использования:</b>	<b><i>LoadMenus(E:\menuscontent.xml)</i></b>

Загрузка файла описаний меню !!NEEDLINK!!. Загрузка обычно происходит на ранней стадии запуска программы, поэтому команда обычно используется в скрипте `$(ZCADPath)/preload/stage0.cmd0 !!NEEDLINK!!`.

### 3.4.30. LoadPalettes

<b>Команда:</b>	<b><i>LoadPalettes</i></b>
<b>Операнды:</b>	Обязательный. Путь и имя файла с описанием палитр
<b>Пример использования:</b>	<b><i>LoadPalettes(E:\palettescontent.xml)</i></b>

Загрузка файла описаний палитр !!NEEDLINK!!. Загрузка обычно происходит на ранней стадии запуска программы, поэтому команда обычно используется в скрипте `$(ZCADPath)/preload/stage0.cmd0 !!NEEDLINK!!`.

### 3.4.31. LoadToolbars

<b>Команда:</b>	<b><i>LoadToolbars</i></b>
<b>Операнды:</b>	Обязательный. Путь и имя файла с описанием тулбаров
<b>Пример использования:</b>	<b><i>LoadMenus(E:\toolbarscontent.xml)</i></b>

Загрузка файла описаний тулбаров !!NEEDLINK!!. Загрузка обычно происходит на ранней стадии запуска программы, поэтому команда обычно используется в скрипте

`$(ZCADPath)/preload/stage0.cmd0 !!NEEDLINK!!`.

### 3.4.32. MatchProp

Команда:	<i>MatchProp</i>
Иконка:	
Пример использования:	<i>MatchProp</i>

Перенос свойств примитива на другие примитивы. После запуска предлагается выбрать исходный примитив, чьи свойства будут взяты как исходные. После выбора в инспекторе объектов будут показаны опции команды

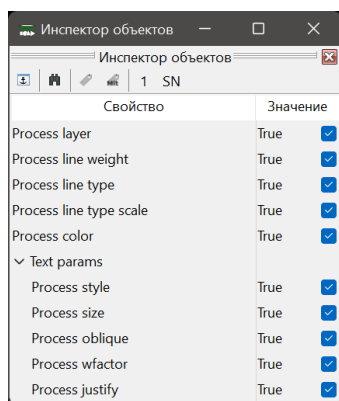


Рисунок 17. Опции команды *MatchProp*

Далее предлагается выбрать целевые примитивы, свойства исходного примитива отмеченные в опциях будут присвоены целевым примитивам.

### 3.4.33. Merge

Команда:	<i>Merge</i>
Операнды:	Обязательный. Путь и имя файла dxf
Пример использования:	<i>Merge(E:\myfile.dxf)</i>


Подгружает файл DXF в текущий чертеж. Повторные определения блоков игнорируются

### 3.4.34. MergeBlocks

Команда:	<i>MergeBlocks</i>
Операнды:	Обязательный. Путь и имя DXF файла
Пример использования:	<i>MergeBlocks (E:\actionscontent.xml)</i>

Подгружает файл DXF в библиотеку блоков !!NEEDLINK!!. Повторные определения блоков игнорируются !!NEEDLINK!!. Загрузка обычно происходит на стадии запуска программы, поэтому команда обычно используется в скрипте `$(ZCADPath)/preload/autorun.cmd` !!NEEDLINK!!.

### 3.4.35. Mirror

Команда:	<i>Mirror</i>
Иконка:	
Предвыбор примитивов:	Требуется

Зеркальное отражение выбранных примитивов относительно прямой. Исходные примитивы нужно выбрать до запуска команды. После запуска требуется указать 2 точки лежащие на прямой относительно которой будет произведено отражение. Во время указания прямой в инспекторе объектов можно указать действие над исходными примитивами - удалить или отавить в чертеже

### 3.4.36. Move

Команда:	<i>Move</i>
Иконка:	
Сочетания клавиш:	Ctrl + ALT + M; M
Предвыбор примитивов:	Требуется

Перенос выбранных примитивов. Исходные примитивы нужно выбрать до запуска команды. После запуска требуется указать 2 точки вектора на который будет произведен перенос.

MultiSelect2ObjIbbsp NavSelectSubNodes NumDevices OPS\_SPBuild OPS\_Sensor\_Mark  
ObjInspCopyToClip OnDrawingEd Options OrtoDevPlace Pan PasteClip PlaceSmokeDetectorOrto  
PolyDiv PolyEd PolyTest Polygon Print ProfileBuild ProjectTree QSave Quit ReadBlockLibrary  
RebuildTree Rectangle Redo Regen RegenZEnts


### 3.4.37. Rotate

Команда:	<i>Rotate</i>
Иконка:	
Сочетания клавиш:	Ctrl + ALT + R; R
Предвыбор примитивов:	Требуется

Поворот выбранных примитивов. Исходные примитивы нужно выбрать до запуска команды. После запуска требуется указать точку определяющую угол поворота.

RotateEnts

### 3.4.38. SaveAs

Команда:	<i>SaveAs</i>
Иконка:	

Сочетания клавиш:	Shift + Ctrl + S
Меню:	<b>Файл &gt; Сохранить как ...</b>

Сохранить текущий чертеж под новым именем. Будет открыто окно выбора файла, после чего произойдет сохранение. Имя чертежа будет изменено, дальнейшие команды **QSave** будут сохранять файл под новым именем

### 3.4.39. SaveLayout

Команда:	<b>SaveLayout</b>
Меню:	<b>Настройки &gt; Сохранить разбивку окон по умолчанию</b>

Сохранить текущую разбивку окон как разбивку по умолчанию в файл \$(ZCADPath)/components/defaultlayout.xml

### 3.4.40. SaveOptions

Команда:	<b>SaveOptions</b>
Меню:	<b>Настройки &gt; Сохранить параметры</b>

Сохранить текущие настройки программы в файлы \$(ZCADPath)/rtl/sysvar.pas !!NEEDLINK!! и \$(ZCADPath)/rtl/config.xml !!NEEDLINK!!

Scale SelMat SelObjChangeColorToCurrent SelObjChangeDimStyleToCurrent  
 SelObjChangeLTypeToCurrent SelObjChangeLWToCurrent SelObjChangeLayerToCurrent  
 SelObjChangeTStyleToCurrent SelSim SelectAll SelectFrame SelectObjectByAddress  
 SelectOnMouseObjects SetObjInsp Show ShowPage ShowToolBar SnapProperties StoreFrustum  
 Stretch Text TextStyles TreeStat Undo Units UnitsMan

### 3.4.41. UpdatePO

Команда:	<b>UpdatePO</b>
----------	-----------------


Обновление файлов локализации (**Локализация программы**), выполнить команду можно только запустив программу с ключом командной строки **updatepo** (**Переключатели командной строки**) Команда позволяет записать в файл languages\rtzcad.po все новые строки требующие перевода обнаруженные в файлах данных при этом запуске программы и стереть из данного файла строки которые требовали перевода раньше, но при этом запуске обнаружены не были. Команда выводит диалоговое окно на подтверждение действий, в случае подтверждения, rtzcad.po перезаписывается

VarReport

### 3.4.42. VarsEd

Команда:	<b>VarsEd</b>
----------	---------------



<b>Иконка:</b>	
<b>Меню:</b>	<i>Изменить &gt; Редактор переменных примитива</i>
<b>Предвыбор примитивов:</b>	Примитив, редактирование переменных которого будет осуществляться


!!NEEDEDDETAILS!!

### 3.4.43. VarsEdBD

<b>Команда:</b>	<i>VarsEdBD</i>
<b>Операнды:</b>	Обязательный. Имя блока набор переменных которого будет редактироваться
<b>Пример использования 2:</b>	<i>Load(DEVICE_PS_SMOKE)</i>

!!NEEDEDDETAILS!!

### 3.4.44. VarsEdSel

<b>Команда:</b>	<i>VarsEdSel</i>
<b>Иконка:</b>	
<b>Меню:</b>	<i>Изменить &gt; Редактор переменных нескольких примитивов</i>
<b>Предвыбор примитивов:</b>	Примитивы, редактирование переменных которых будет осуществляться

!!NEEDEDDETAILS!!

### 3.4.45. VarsLink

<b>Команда:</b>	<i>VarsLink</i>
-----------------	-----------------

Создание централизованной связки примитивов (см. [Централизация](#)) После запуска команды предлагается указать главный примитив звязки (**центральный**) затем его **представителей**. Указанные примитивы должны иметь расширение `extdrVariables` и не участвовать в других связках

!!NEEDEDDETAILS!!

View Zoom ZoomWindow ft rt test

## 4. Настройка программы

### 4.1. Параметры работы

ZCAD пока не имеет отдельного окна настроек, настройки постоянно доступны в инспекторе объектов (см. раздел [2. Инспектор объектов](#)) когда на чертеже не выбрано никаких примитивов.

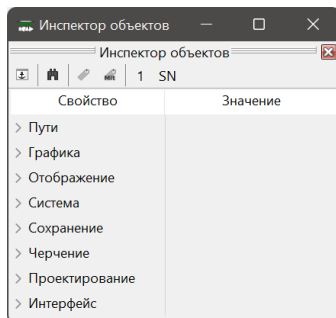


Рисунок 18. Настройки программы

Настройки сгруппированы следующим образом

- **Пути** - настройка путей программы
- **Графика** - настройка путей программы
- **Отображение** - настройка путей программы
- **Система** - настройка путей программы
- **Черчение** - настройка путей программы
- **Проектирование** - настройка путей программы
- **Интерфейс** - настройка путей программы

#### 4.1.1. Пути

На данной вкладке сгруппированы пути используемые программой для поиска файлов.

- **Файлы поддержки** - пути поиска различных файлов программы
- **Шрифты** - пути поиска шрифтов
- **Файл альтернативного шрифта** - файл шрифта используемый для замены отсутствующих шрифтов
- **Шаблоны** - путь для поиска файлов шаблонов чертежей
- **Шаблон по умолчанию** - файл шаблона используемый программой при создании нового чертежа
- **Текущая раскладка окон** - файл раскладки окон загружаемый при запуске программы
- **Программа** - путь к бинарному файлу (вычисляется при запуске, доступен только для чтения)
- **Временные файлы** - путь к папке временных файлов (вычисляется при запуске,

доступен только для чтения)

- **База описаний устройств** - путь к "базе данных" оборудования программы

В качестве разделителя для путей используется ";", разделитель путей "/", также доступны следующие макросы:

Таблица 3. Макросы сокращений путей

<b>\$(ZCADPath)</b>	Путь к бинарному файлу
<b>\$(LastAutoSaveFile)</b>	Путь к последнему файлу автосохранения
<b>\$(TEMP)</b>	Путь к папке временных файлов
<b>\$(SystemFontsPath)</b>	Путь к папке шрифтов ОС (пока только win, возвращает SHGetSpecialFolderPath(CSIDL_FONTS))
<b>\$(UserFontsPath)</b>	Путь к папке шрифтов пользователя (пока только win, возвращает SHGetSpecialFolderPath(CSIDL_LOCAL_APPDATA)\Microsoft\Windows\Fonts)

### 4.1.2. Графика

На данной вкладке сгруппированы настройки графической подсистемы зкад

- **Устройство отображения** - графическая библиотека используемая для рендера чертежа. Доступны варианты: OpenGL - предпочтительный вариант (требует наличия драйвера производителя видеокарты), GDI - вариант для случая когда первый вариант не работает, LCLCanvas - данный вариант не предназначен для работы с чертежом, но если не работают на **OpenGL**, ни **GDI** можно попробовать его. Настройка работает только для вновь открытых/созданных чертежей, уже открытые чертежи по прежнему будут использовать устройство отображения выбранное на момент их создания
- **Параметры текущего устройства отображения** - здесь собраны некоторые настройки и отладочная информация устройства отображения используемого текущим чертежом
- **Версия GLU** - версия библиотеки GLU используемая программой (используется для триангуляции и построения сплайнов, ZCAD не может работать с реализацией GLU от Microsoft, в комплекте поставляется MESA GLU) (доступен только для чтения)
- **Расширения GLU** - список расширений предоставляемый GLU (доступен только для чтения)
- **Время рендера** - время в миллисекундах затраченное на последнее полное отображение чертежа (доступен только для чтения)
- **Время обновления** - время в миллисекундах затраченное на последнее обновление чертежа. Для экономии ресурсов ZCAD старается по возможности не перечерчивать весь чертеж, а только обновлять его измененные фрагменты (доступен только для чтения)
- **Максимальное время одного прохода рендера** - время в миллисекундах доступное на рендер чертежа, в случае его превышения рендер будет остановлен и продолжен в следующем цикле обработки событий. 0 - возможность не используется (доступен

только для чтения)

- **Ухудшение изображения** - параметры управления детализацией отрисовки чертежа. При включении и превышении порога времени рендера, детализация чертежа будет уменьшаться для уменьшения времени отрисовки
- **Деградация при перетаскивании** - При включении понижает детализацию чертежа при выполнении операций зумирования и паранамирования, для повышения отзывчивости. при завершении данных операций чертеж отрисовывается с нормальной детализацией
- **Максимально шаблонов типа в линии** - Максимально доступное количество штрихов\точек\символов в примитивах, при превышении примитив будет отрисован сплошной линией

### 4.1.3. Отображение

- **Системная геометрия** - отображение некоторой вспомогательной информации, например габаритов примитивов
- **Цвет вспомогательной геометрии** - цвет для вывода информации из предыдущего пункта
- **Масштаб колеса мыши** - коэффициент масштабирования чертежа при вращении колеса
- **Размер апертуры привязки** - размер в пикселях зоны "притягивания" курсора к точкам привязки
- **Размер прицела** - размер в процентах от размера видимой области графического курсора ZCAD
- **Убирать системный курсор в области отрисовки** - опция отключает системный курсор при черчении, оставляя только курсор ZCAD
- **Размер ручек** - размер "ручек" редактирования примитивов
- **Цвет фона** - RGB цвет фона чертежа
- **Цвет не выбранных ручек** - индексный цвет "ручек" примитивов
- **Цвет выбранных ручек** - индексный цвет выбранных "ручек" примитивов
- **Цвет горячих ручек** - индексный цвет "ручек" находящихся под курсором
- **Масштаб отображения толщин линий** - число от 2 до 20 характеризующее толщину отображения веса линий на чертеже, больше - жирнее
- **Толщина линий по умолчанию** - толщина линий принятая для отображения веса линий Default

### 4.1.4. Система

- **Версия программы** - версия сборки в формате git describe --tags (доступен только для чтения)
- **Информация о сборке** - разная информация: целевая платформа, версия компилятора и т.п. (доступен только для чтения)

- **Время работы** - длительность текущей сессии (доступен только для чтения)
- **Один экземпляр** - контроль повторного запуска программы. При установке данного параметра возможен запуск только одной сессии программы, попытки запустить следующую сессию только активируют уже запущенную
- **Не показывать заставку** - отключение сплэш скрина при запуске программы
- **Не загружать раскладку окон** - не загружает файл раскладки окон программы при старте, программа запускается в "не пристыкованном" режиме (доступен только для чтения, устанавливается ключом командной строки !!NEEDLINK!!)
- **Обновления РО файлов** - режим контроля и обновления файлов локализации программы, используется совместно с командой **UpdatePO** !!NEEDLINK!! (доступен только для чтения, устанавливается ключом командной строки !!NEEDLINK!!)

#### 4.1.5. Сохранение

- **Автосохранение** - включает работу автосохранения
- **Время до автосохранения** - время в секундах оставшееся до очередного автосохранения (доступен только для чтения)
- **Время между автосохранениями** - настройка времени между сохранениями в секундах
- **Файл автосохранения** - путь и имя файла автосохранения

#### 4.1.6. Черчение

- **Отображать вес линий** - Включение отображения веса линий
- **Режим привязки** - Битовая маска настроек привязок. в данном месте просто для информации, не используется
- **Режим полярной трассировки** - Включение трассировки
- **Текущий слой** - отображает и позволяет редактировать текущий слой
- **Текущий вес линии** - отображает и позволяет редактировать текущий вес линии
- **Текущий вес цвет** - отображает и позволяет редактировать текущий цвет
- **Масштаб типов линий чертежа** - отображает и позволяет редактировать глобальный масштаб типов линий чертежа
- **Текущий масштаб типов линий примитивов** - отображает и позволяет редактировать текущий масштаб типов линий
- **Стиль размеров** - отображает и позволяет редактировать текущий стиль размеров
- **Поворачивать текст в описании линий** - Поворачивает текстовые элементы в стилях линий для более удобного чтения
- **Стиль текста** - отображает и позволяет редактировать текущий стиль текста
- **LUnits (формат линейных единиц)** - аналог DXF переменной LUnits
- **LUPrec (точность линейных единиц)** - аналог DXF переменной LUPrec
- **AUnits (формат угловых единиц)** - аналог DXF переменной AUnits

- **AUPrec (точность угловых единиц)** - аналог DXF переменной AUPrec
- **AngDir (направление положительного угла)** - аналог DXF переменной AngDir
- **AngBase (базовый угол)** - аналог DXF переменной AngBase
- **InsUnits (масштаб вставки блока)** - аналог DXF переменной InsUnits
- **TextSize (размер вновь созданных текстовых примитивов)** - аналог DXF переменной TextSize
- **Настройка шаговой привязки** - настройки привязки к регулярной прямоугольной сетке
- **Шаг сетки** - настройки отображения регулярной прямоугольной сетки
- **Показать сетку** - включение отображения сетки
- **Шаг** - включение шаговой привязки
- **Редактирование составных объектов** - включение отдельного выделения примитивов являющихся частью сложных примитивов !!NEEDLINK!!
- **Вспомогательная геометрия** - отображение вспомогательной геометрии
- **Отображать выбранный объект в инспекторе** - показывать свойства выбранного примитива в инспекторе

## 4.2. Структура директорий

Общая структура директорий программы имеет вид:

```
Failed to generate image: Could not load PlantUML. Either require 'asciidoctor-
diagram-plantuml' or specify the location of the PlantUML JAR(s) using the
'DIAGRAM_PLANTUML_CLASSPATH' environment variable. Alternatively a PlantUML binary can
be provided (plantuml-native in $PATH).
!include styles/filesystem-tree-style.iuml
legend
**cad**
|_ **bin** //бинарные файлы в подкаталогах по платформам//
|_ **i386-win32**
|_ **x86_64-linux**
|_ **x86_64-win64**
|_ **cfg** //конфигурационные файлы программы//
|_ **components** //различные скрипты, конфигурационные файлы//
|_ **configs** //различные конфигурационные файлы//
|_ **navigators** //пресеты навигаторов//
|_ **menu** //конфигурационные файлы элементов пользовательского интерфейса//
|_ **data** //файлы данных программы//
|_ **blocks** //библиотека блоков//
|_ **el**
|_ **ops**
|_ **dictionaries** //словари проверки орфографии//
|_ **examples** //примеры файлов dxf//
|_ **fonts** //шрифты shx и ttf//
|_ **images** //иконки//
```

```

|_ **actions**
|_ **draw**
|_ **snap**
|_ **valec**
|_ **zelectro**
|_ **palettes**
|_ **languages** //PO файлы локализаций//
|_ **preload** //папка для скриптов выполняемых при запуске программы//
|_ **programdb** //встроенная база оборудования//
|_ **rtl** //различные скрипты//
|_ **dwg**
|_ **objcalc**
|_ **objdefunits**
|_ **include**
|_ **styles**
|_ **velec**
|_ **templates** //шаблоны пустых чертежей//
end legend

```

Дистрибутив разбит на 3 директории **bin**, **cfg**, **data**

Директория **bin** содержит исполняемые файлы и может располагаться в любом месте файловой системы

Директория **cfg** может располагаться либо рядом с **bin** либо ее содержимое должно находиться в **C:\ProgramData\zcad\** (на линуксе в **/etc/zcad/**)

Директория **data** может располагаться либо рядом с **bin** либо местоположение ее содержимого должно быть указано в ключе **PreferredDistribPath** в файле **cfg\configs\config.xml**

Отдельно стоит рассмотреть содержимое директории **data/preload**:

```

Failed to generate image: Could not load PlantUML. Either require 'asciidoctor-
diagram-plantuml' or specify the location of the PlantUML JAR(s) using the
'DIAGRAM_PLANTUML_CLASSPATH' environment variable. Alternatively a PlantUML binary can
be provided (plantuml-native in $PATH).
!include styles/filesystem-tree-style.iuml
legend
**preload**
|_ **Различные поддиректории**
|_ **autorun.cmd** //командный скрипт выполняемый на последней стадии запуска
программы//
|_ **stage0.cmd0** //командный скрипт выполняемый на ранней стадии запуска программы//
end legend

```

При запуске программы осуществляется сканирование **data/preload** и вложенных директорий и выполнение найденных скриптов **cmd0** и **cmd** скрипты с расширением **cmd0** выполняются на ранней стадии запуска программы (до создания GUI) и содержат в себе

команды загрузки описаний экшенов, меню и тулбаров. Файлы с расширением **cmd** выполняются при завершении инициализации, после создания GUI и содержат в себе команды загрузки файлов определений блоков и устройств.

При сканировании директорий первым будет выполнен скрипт **stage0.cmd** далее будут выполнены все файлы с расширением **cmd** лежащие в данной директории и ниже по иерархии файловой системы. Для каждой вложенной директории также первым выполняется **autorun.cmd** и далее другие скрипты **\\*.cmd** лежащие в данной директории

## 4.3. Переключатели командной строки

При запуске ZCAD из командной строки можно задать опции командной строки корректирующие его поведение. Некоторые опции работают как флаги, некоторые опции требуют указания аргументов

Таблица 4. Переключатели командной строки

<b>nosplash</b>	Не показывать сплэшскрин
<b>updatepo</b>	Обновить файлы локализации (используется совместно с командой UpdatePO) ведет учет всех запросов на локализацию, помечает не используемые и добавляет вновь встреченные строки. При использовании ключа фактически перевод строк не производится, интерфейс остается английским
<b>noloadlayout</b>	Не загружать файл раскладки окон
<b>debugui</b>	Включить "отладочный" интерфейс - некоторые элементы интерфейса для разработчика
<b>experimentalfeatures</b>	Включить "экспериментальные" фишки - то что разрабатывается на данный момент не закончено и не нужно пользователю
<b>notcheckuniqueinstance</b>	Не проверять повторный запуск, с данным ключом может быть запущено несколько копий программы
<b>logfile</b>	Принудительно задать путь к лог файлу. Требуется аргумент - путь и имя файла
<b>leam</b>	Принудительно разрешить все модули логирования
<b>lem</b>	Принудительно разрешить указанный модуль (модули) логирования. Требуется аргумент(ы) - имя модуля(лей)
<b>ldm</b>	Принудительно запретить указанный модуль (модули) логирования. Требуется аргумент(ы) - имя модуля(лей)
<b>lemm</b>	Принудительно разрешить модули логирования чьи имена соответствуют маске(кам). Требуется аргумент(ы) - маску(ки) имени модуля(лей)



<b>ldmm</b>	Принудительно запретить модули логирования чьи имена соответствуют маске(кам). Требуется аргумент(ы) - маску(ки) имени модуля(лей)
<b>lcl</b>	Установить уровень логирования. Требуется аргумент - уровень логирования
<b>maxstackframecount</b>	Установить максимальную глубину размотки стека при обработке ошибок
<b>runscript</b>	Запустить скрипт(ы) после запуска программы. Требуется аргумент(ы) - Имя файла(ов) скрипта(ов)
<b>memprofiling</b>	Запуск профилировщика памяти на этапе инициализации программы. см. команду dbgMemProfiler !!NEEDLINK!!

Таблица 5. Уровни логирования

<b>LM_Trace</b>	Вывод всего подряд. Максимально подробный и перегруженный лог.
<b>LM_Debug</b>	Журналирование моментов вызова «крупных» операций
<b>LM_Info</b>	Разовые операции, которые повторяются крайне редко, но не регулярно. (загрузка конфига, плагина, запуск бэкапа)
<b>LM_Warning</b>	Неожиданные параметры вызова и т.п. Вообще все, что может свидетельствовать о не штатном использовании.
<b>LM_Error</b>	Вывод сообщений об ошибках.
<b>LM_Fatal</b>	Вывод сообщений об критических ошибках, имхо стоит убрать.
<b>LM_Necessarily</b>	Вывод в любом случае.

При штатном запуске программы используется **LM\_Info** как текущий уровень. В лог попадают сообщения текущего уровня и уровней ниже по таблице, все что выше фильтруется

Список доступных модулей логирования пока не определен, его можно посмотреть в конце лога при штатном запуске

Пример запуска программы:

```
zcad nosplash logfile d:\zcad.log lem translator,shx ldm default lcl lm_trace
```

Не показывать сплэшскрин, переназначить вывод лога в **d:\zcad.log**, разрешить вывод сообщений в лог от системы (модуля) перевода и парсера shx шрифтов, запретить сообщения модуля по умолчанию, установить уровень логирования **LM\_Trace**

## 5. Для разработчика

### 5.1. Сборка программы из исходников

Программа имеет 2 режима сборки **ZCAD** и **ZCADELECTROTECH**, в первом только базовые CAD функции, во втором плюсом чуток электрической специфики. Советую пробовать собрать **ZCADELECTROTECH**, т.к. я его сам всегда использую, соответственно он более стабилен.

Простая компиляция исходников даст вам только файл **zcad.exe**, но для работы нужны некоторые другие файлы, без которых программа не работает (не говоря о **allgeneratedfiles.inc**, **zcadversion.inc** и **buildmode.inc** без которых даже не скомпилируется)

Для автоматизации процесса сборки был написан скрипт на основе системы сборки **make**. Опишу его использование применительно к тридцати двух битным **Lazarus2.2** и **fpc3.2.2** под управлением ОС **Windows**

Итак:

#### 5.1.1. Установка Lazarus

Устанавливаем последний релизный **Lazarus** (2.2 на момент написания текста). Гарантированно **ZCAD** компилируется в транковом **Lazarus** транковом **fpc**, с релизами бывают нюансы, но они решаемы.

Запускаем, проверяем работоспособность **Lazarus** - собираем тестовый пустой проект. тут проблем быть не должно.

Ищем идущую в комплекте **fpc** утилиту **make**, она скорее всего лежит тут **lazarus\fpc\3.2.2\bin\i386-win32\make.exe** но, мало ли. В дальнейшем я считаю что **Lazarus** установлен на диск **C**, и путь для запуска **make** соответственно **C:\lazarus\fpc\3.2.2\bin\i386-win32\make.exe** если что - уточнить по месту)).

Также еще понадобятся такие пути:

- путь к **Lazarus** **C:\lazarus**
- путь к первичному файлу настроек **Lazarus**, по умолчанию он **C:\Users\<ИМЯПОЛЬЗОВАТЕЛЯ>\AppData\Local\lazarus**

Если имя пользователя у вас на кириллице, примите мои поздравления! Требуются дополнительные действия, пользователи с нормальными именами спокойно переходят к пункту 2.

Кириллица в путях не поддерживается утилитой **make**, или я с этим не разобрался. Придется "перенастроить" **Lazarus** чтоб настройки хранились по 'нормальным' путям. Для этого в папке **C:\lazarus** создаем файл **runlazarus.bat** следующего содержания:

```
startlazarus.exe --pcp=C:\lazarus\mylazcfg
```

и далее всегда используем его для запуска Lazarus IDE, все что написано ниже вам следует отредактировать из расчета что путь к настройкам Lazarus будет **C:\lazarus\mylazcfg**

### 5.1.2. Получение ZCAD

Клонируем исходники зкада (или скачиваем архивом, но это плохо, лучше клонировать git ом). Некоторые части исходников оформлены субмодулями **git**, поэтому:



субмодули **git** используемые **ZCAD** нужно инициализировать и обновить.

По описанным выше причинам путь до папки **zcad** не должен содержать не латинские символы Тут будет много файлов\папок, но основные:

- **zcad\cad\_source** - папка с исходниками зкад
- **zcad\environment** - папка с файлами окружения программы и исходник небольшой программы **typeexporter** настраивающей исходники зкад для компиляции
- **zcad\Makefile** - файл с скриптами установки
- **zcad\cad** - данной папки изначально нет, будет создана в пункте 4 и содержит скомпилированный дистрибутив zcad со всеми нужными файлами

### 5.1.3. Установка пакетов от которых зависит ZCAD

Для облегчения я приложил пакеты от которых зависит **ZCAD** в дистрибутив исходников (за исключением идущих в составе **Lazarus**). Открываем командную строку в папке **zcad**, там где лежит файл **Makefile**

Нужно установить в **Lazarus** пакеты требуемые для компиляции **ZCAD**, данный пункт выполняется только один раз, для свежее установленного **Lazarus**, если пакеты установлены, пропускаем данный пункт (но если что, то повторное выполнение ничего страшного не несет). Выполняем:

```
C:\lazarus\fpc\3.2.2\bin\i386-win32\make          installpkgstolaz          LP=C:\lazarus
PCP=C:\Users\<ИМЯПОЛЬЗОВАТЕЛЯ>\AppData\Local\lazarus
```

**installpkgstolaz** это пропишет в конфигах **Lazarus** требуемые пакеты из **zcad\cad\_source\other** и **zcad\cad\_source\components** и пересобирет **Lazarus**. По неясным причинам пересборка в данном пункте иногда завершается ошибкой, но ничего страшного, просто идем дальше, **Lazarus** докомпилирует все нужное в 4.

### 5.1.4. Компиляция ZCAD

Собственно запускаем компиляцию зкад, запустив следующее:

```
[.shell]C:\lazarus\fpc\3.2.2\bin\i386-win32\make          cleanzcadelectrotech          LP=C:\lazarus
PCP=C:\Users\<ИМЯПОЛЬЗОВАТЕЛЯ>\AppData\Local\lazarus#
```

**cleanzcadelectrotech** - данная цель выполнит сборку программы в режиме **ZCADELECTROTECH**, замените на **cleanzcad** - если желаете режим **ZCAD**

Скрипт выполнит следующее:

- **СОТРЕТ** папки `zcad\cad` и `zcad\cad_source\autogenerated` если они присутствует **СО ВСЕМ ИХ СОДЕРЖИМЫМ** ничего не спрашивая
- создаст папки `zcad\cad` и `zcad\cad_source\autogenerated`
- скопирует нужные для работы файлы из `zcad\environment\runtimefiles` в `zcad\cad`
- создаст файл `zcad\cad_source\zcadversion.inc`
- создаст файл `zcad\cad_source\autogenerated\buildmode.inc`
- скомпилирует `zcad\cad_source\cad_source\utils\typeexporter.lpi` и запустит его с нужными параметрами, **typeexporter** в свою очередь наполнит `zcad\cad_source\autogenerated` в том числе создаст `zcad\cad_source\autogenerated\allgeneratedfiles.inc` (только после этого шага зкад может быть собран)
- скомпилирует **ZCAD**

Если все прошло нормально, имеем наполненную как надо папку `zcad\cad`, в том числе свежесозданный запускаемый бинарник `zcad\cad\bin\i386-win32\zcad.exe` В дальнейшем можно просто открыть в Lazarus файл `zcad\cad_source\zcad.lpi` и смотреть-собирать исходники как обычно в IDE



**Lazarus, FPC и ZCAD** развивающиеся проекты, информация устаревает и бывают нюансы. В частности на данный момент из-за бага FPC <https://gitlab.com/freepascal.org/fpc/source/-/issues/39387> в IDE работает только полная пересборка зкада, т.е. в Lazarus если просто нажать **F9** зкад не соберется с вылетом компилятора, при любых изменениях надо всегда выполнять полную пересборку **shift-F9**

## 5.2. Локализация программы

Для локализации **ZCAD** используется система интернационализации на основе **Gettext** и **\*.po** файлов встроенная в **Lazarus**. Локализации подлежат как тексты прописанные в исходниках программы, так и заданные в файлах загружаемых во время работы программы - например файл `zcad\menu\menuscontent.xml` содержит наполнение палитр программы, его содержимое неизвестно в момент компиляции программы, он будет прочитан только при запуске. Обычный подход предполагает разные копии загружаемых в рантайм файлов для каждой локализации. Но в **ZCAD** принят несколько другой подход - перевод содержимого данных файлов в момент их загрузки.

Прежде всего стоит заметить, как описано в ([Структура директорий](#)) файлы локализаций используемые программой лежат в папке `cad/languages`, данная папка создается в процессе ([Сборка программы из исходников](#)) в нее копируются файлы из соответствующих мест `environment`. Поэтому работа с локализацией осуществляется в `cad/languages`, но для коммита файлы нужно скопировать в папку `environment`.

Тексты прописанные в исходниках программы и оформленные как **resourcestring** при компиляции программы автоматически собираются в файле `zcad.po` и разносятся по файлам

локализаций **zcad.XX.po** где **XX** - идентификатор языка. Файлы **zcad.XX.po** можно непосредственно переводить используя предназначенные для этого программы например **Poedit** и подобные. На данный момент **ZCAD** имеет только русскую локализацию **zcad.ru.po**. Добавить локализацию можно просто создав соответствующий файл, например **zcad.de.po** для немецкой.

Также в **cad/languages** лежат файлы перевода пакетов используемых в **ZCAD**:

- **anchordockstr.XX.po** перевод пакета докинга окон [AnchorDocking](#)
- **lclstrconsts.XX.po** перевод LCL

Тексты получаемые из рантайм файлов собираются в файле **rtzcad.po**, переводы находятся в **zcad.XX.po** также как и описано выше новую локализацию можно получить созданием соответствующего файла. Актуализация **rtzcad.po** и **zcad.XX.po** не производится автоматически, для этого нужно выполнить описанные ниже действия.

Запускаем программу с ключом командной строки ([Переключатели командной строки](#)) **updatepo**. При этом локализация программы не будет выполняться, будет работать оригинальный английский интерфейс. Создаем новый чертеж и вводим команду [UpdatePO](#) (**UpdatePO**) будет показано окно с информацией сколько обнаружено неактуальных строк и сколько добавлено новых. При нажатии **Ok** будет перезаписан файл **rtzcad.po**, неактуальные строки будут удалены, новые добавлены.

Далее из **rtzcad.po** строки нужно обновить в файлах локализаций **rtzcad.XX.po**, это можно сделать с помощью команды

```
C:\lazarus\fpc\3.2.2\bin\i386-win32\make          updatelocalizedpofiles          LP=C:\lazarus
PCP=C:\Users\<ИМЯПОЛЬЗОВАТЕЛЯ>\AppData\Local\lazarus
```

Образование путей смотри ([Сборка программы из исходников](#)), сценарий **updatelocalizedpofiles** выполняет обновление файлов локализации.

После этого файлы **rtzcad.XX.po** можно переводить в специализированной программе

## 5.3. Написание документации ZCAD

Документация программы разрабатывается в формате AsciiDoc. Исходники данного руководства лежат в папке **cad\_source/docs/userguide/**. AsciiDoc - простой текстовый формат, но для генерации pdf и html из него потребуются дополнительные манипуляций

### 5.3.1. Установка AsciiDoctor для Windows

Т.к. AsciiDoc написан на языке Ruby, придется установить средства разработки этого языка:

Идем на сайт <https://rubyinstaller.org/downloads/> Скачиваем и устанавливаем последний Ruby+Devkit (на момент написания это 3.2.2-1) В процессе установки нужно будет указать какой вариант MSYS использовать, выбираем базовый 1, затем для завершения установки просто жмем enter

После установки Ruby в командной строке можно будет использовать gem - пакетный менеджер Ruby, с помощью его собственно устанавливает AsciiDoctor и требуемые расширения. Открываем командную строку и вводим эти три команды последовательно:

```
gem install asciidoctor
```

```
gem install asciidoctor-pdf
```

```
gem install asciidoctor-diagram
```

Дополнительно для работы asciidoctor-diagram требуется наличие в системе Java версии не ниже 11, если необходимо идем и устанавливаем по ссылке <https://www.oracle.com/java/technologies/downloads/#jdk21-windows>

### 5.3.2. Установка GraphViz

Также понадобятся средства визуализации графов из состава Graphviz. Идем на <https://graphviz.org/download/> и качаем установщик для Windows на момент написания это graphviz-9.0.0 (64-bit) EXE installer при установке выбираем вариант **Add Graphviz to the system PATH for all users** чтоб графвиз был доступен из командной строки.

### 5.3.3. Генерация руководства пользователя



Описанное выше нужно произвести один раз, это установит на компьютер необходимые программы и их зависимости

После всех установок запускаем новую командную строку (рекомендую Far) чтобы все изменения системной переменной PATH вступили в силу. Идем в корневую директорию zcad (там где лежит makefile) и вводим

```
make documentation
```

Ждем выполнения скриптов, по завершению в папке zcad\cad\_source\docs\userguide\ находим сгенерированные файлы userguide.ru.pdf и userguide.ru.html