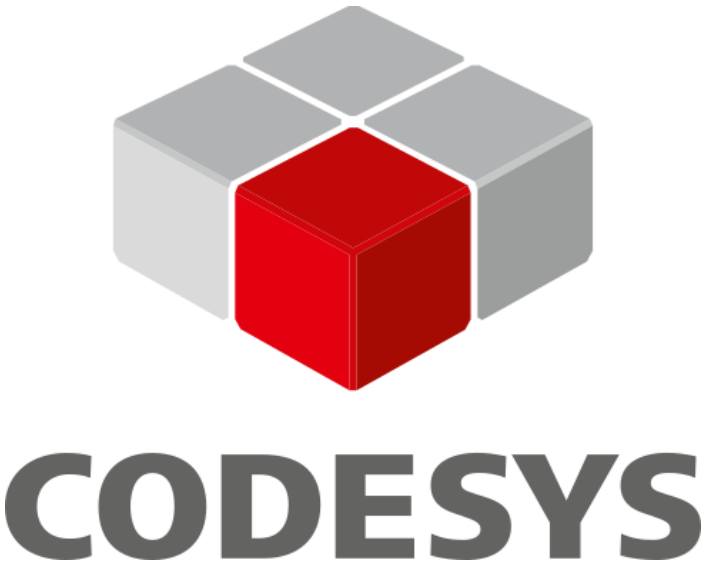
История библиотеки OwenVisuDialogs



27.06.2022
версия 1.0

https://oscat.ru/

Оглавление

Вступление

Как обычно, сначала был интерес.

В процессе работы над библиотекой OwenVisuDialogs Влад использовал несколько специфических программных решений – и я подумал, что они могут быть интересны и другим пользователям CODESYS. Поэтому появилась мысль написать небольшую статью с обзором некоторых фич, которые Влад применил в библиотеке. Пока я составлял список фич, про которые стоит написать – вспомнил историю разработки этой библиотеки и подумал, что она, возможно, тоже ведь может быть кому-то любопытна. Но перед тем, как перейти к **OwenVisuDialogs**, стоило бы рассказать о библиотеке **VisuDialogs**, из которой она выросла. Понятно, что читателей, которым интересно, что там было со стандартными диалогами визуализации CODESYS много лет назад и как они изменялись на протяжении этого времени, едва ли наберется больше десятка, но главное – что мне хотелось еще раз пройтись по всем этим воспоминаниям и попробовать составить из них цельную историю про то, как на интервале двух десятилетий несколько совершенно разных людей приложили руку к, казалось бы, довольно тривиальной и не заслуживающей особого внимания задаче – разработке библиотек диалогов (*в частности, цифровой (numpad) и символьной (keypad) клавиатур*) для среды CODESYS V3.

Приблизительная дата начала этой истории – 2004 год. В этом году я еще не был знаком с Владом и не знал о существовании CODESYS. Поэтому первые 10 лет я описываю на основании информации из баг-трекера CODESYS и своего опыта работы со старыми версиями этой среды.

Евгений Кислов
май – июнь 2022

2004-2014: VisuDialogs и 3S

Баг-трекер CODESYS позволяет получить довольно много интересной с исторической точки зрения информации. Например, самый первый из существующих тикетов по CODESYS V3 в Jira был создан в последний день мая 2005 года – практически 17 лет назад [1]. Можно заметить, что он имеет номер CDS-161 – но нумерация тикетов не соответствует хронологии их создания (*тикет CDS-1, например, создан в апреле 2006*).

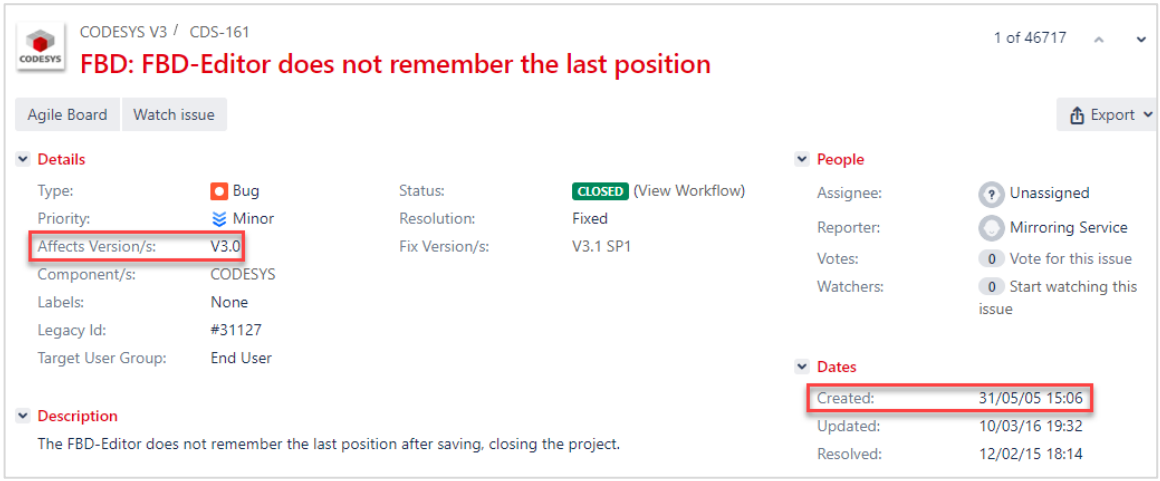


Рис. 1 – Хронологически первый тикет в Jira по CODESYS V3.0

Судя по всему – разработка версии CODESYS V3.0 началась в 2004 году. В некоторых ранних тикетах она помечена как «beta». Я видел презентацию, которую 3S в 2005 году показывали заказчикам, планирующим лицензировать систему исполнения – в ней о версии 3.0 не было ни слова; вероятно, в то время она существовала лишь на уровне прототипа. Но уже в версии 3.0 присутствовала визуализация – об этом, опять же, можно узнать по тикетам в баг-трекере.

[1] Я пишу эти строки 16 мая 2022 года.

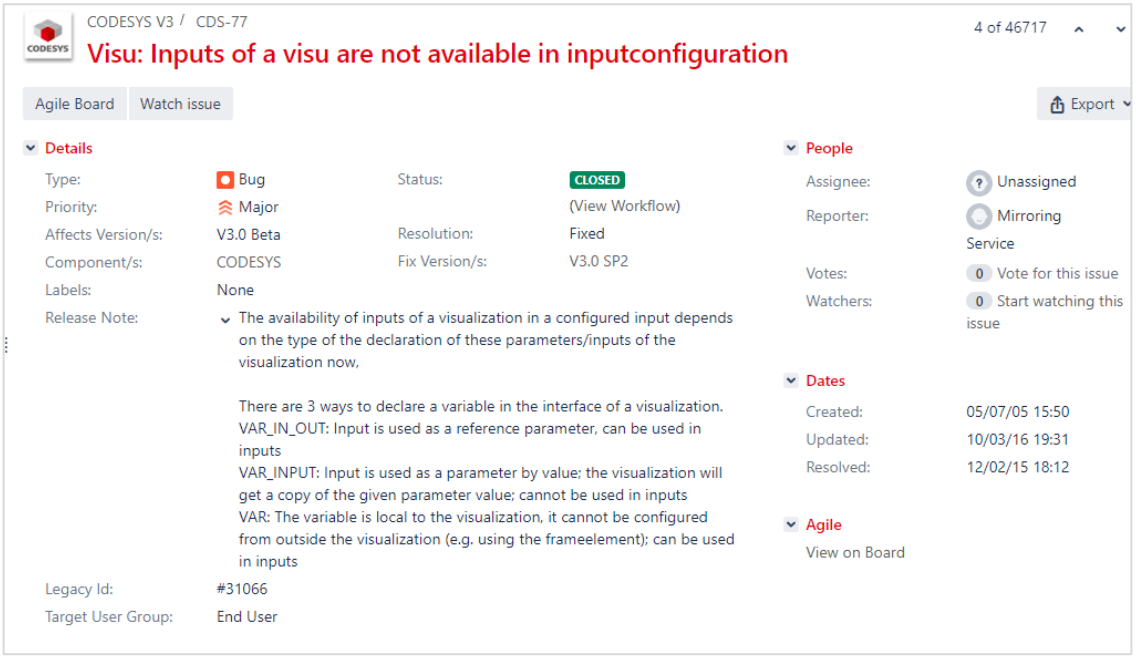


Рис. 2 – Упоминания о визуализации в тикетах CODESYS V3.0

Практически все системы визуализации позволяют оператору вводить какие-либо данные в систему управления – значения уставок и других настроек, названия рецептов и т. д. Средством ввода может являться аппаратная клавиатура ПК или панельного контроллера. Но если экран устройства является сенсорным – то ввод осуществляется с помощью «виртуальной» клавиатуры, реализованной как часть ПО устройства. Обычно среда разработки содержит уже готовые диалоги клавиатур.

Рис. 3 – Панельный контроллер Berghof DC1000 с аппаратной цифровой клавиатурой

В CODESYS эти клавиатуры являются частью библиотеки **VisuDialogs**. Баг-трекер подсказывает (см. рис. 4), что эта библиотека уже входила в состав CODESYS в 2006 году. Вероятно, она присутствовала в составе среды изначально, с самой первой сборки версии 3.0, но первая версия, которую можно увидеть собственными глазами (скачав архив репозитория) – это **3.1.3.0**.

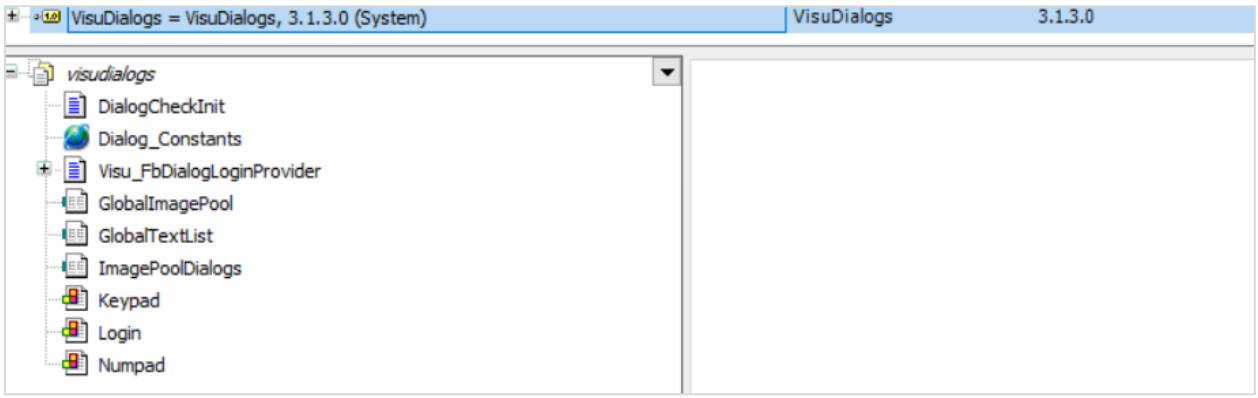


Рис. 4 – Состав библиотеки VisuDialogs 3.1.3.0

В этой версии в библиотеке было всего 3 диалога:

- **Numpad** – диалог цифровой клавиатуры;
- **Keypad** – диалог символьной клавиатуры;
- **Login** – диалог ввода имени пользователя и пароля для подключения к серверу данных контроллера, если для него настроены логин и пароль (в ранних версиях CODESYS сервером данных являлся компонент **DataServer**, а начиная с CODESYS V3.5 SP10 его заменил **DataSource Manager**). Изначально этот диалог не имел никакого отношения к управлению пользователями визуализации (**Visu User Management**), поскольку оно появилось только в версии CODESYS V3.5 SP2, и только вместе с релизом **DataSource Manager** доступ к серверу данных завязали на управление пользователями.

Внешний вид диалогов был точно таким же, как и сейчас:

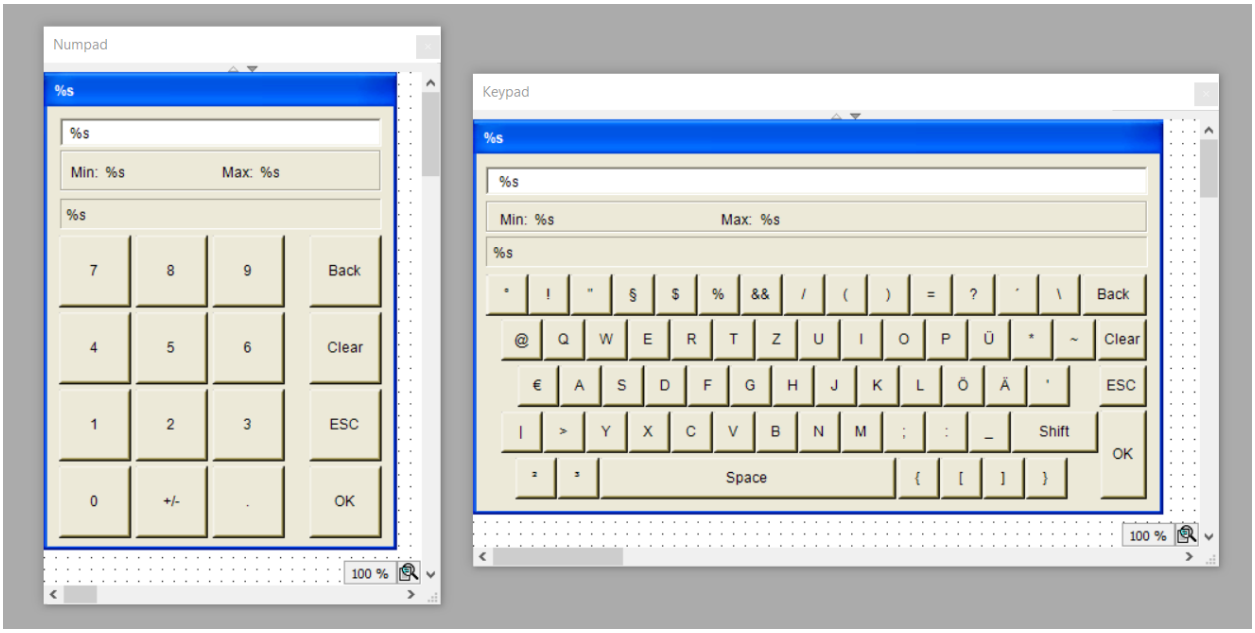


Рис. 5 – Внешний вид диалогов Numpad и Keypad VisuDialogs 3.1.3.0

В дальнейшем в библиотеку добавлялись новые диалоги:

- в версии **3.4.0.0** был добавлен диалог выбора файлов **FileOpenSave** (аналог «Проводника» Windows);
- в версии **3.5.4.0** появился диалог **TextinputWithLimits** – он представлял собой поле для ввода значения с помощью аппаратной клавиатуры с отображением допустимого минимума и максимума для данного значения;
- в версии **3.5.8.0** добавили диалог **NumpadExtended** – версию Numpad'а с возможностью ввода значений в шестнадцатеричной системе счисления (HEX);
- в версии **3.5.11.0** появился диалог **MessageBox** для отображения сообщений (сообщений об ошибках, предупреждений и т. д. – аналог соответствующего окна Windows).

С тех пор вплоть до данного момента[2] состав библиотеки не изменялся.

По мнению многих пользователей, ранние версии библиотеки имели два фундаментальных недостатка:

- аскетичный (иногда это описывается как «стиль Windows 95») дизайн диалогов, слишком мелкий шрифт для подписей клавиш (особенно в диалоге Numpad);
- отсутствие поддержки переключения языков в диалоге Keypad. Более того, этот диалог реализовывает не «английскую», а «немецкую» (QWERTZ) раскладку клавиатуры с умляутами.

С течением времени разработчики CODESYS не решили эти проблемы – но создали для пользователей возможность решить их самостоятельно.

Начиная с определенной[3] версии данная библиотека распространялась в исходниках – ее файл (.library) доступен в директории установки CODESYS по пути **…\CODESYS\Projects\Visu\Dialogs**. Таким образом, дизайн стандартных диалогов не являлся серьезной проблемой – при желании пользователь мог полностью переделать его под свои требования.

Гораздо серьезнее обстояли дела с поддержкой других языков символьной клавиатуры – изначально диалог Keypad был рассчитан на использование переменных типа STRING (с ASCII-based кодировкой), а существенное число панельных контроллеров с CODESYS не поддерживает отображение в визуализации строк в таких кодировках на языках, отличных от английского. Поэтому требовалась поддержка типа WSTRING в диалоге Keypad. Соответствующий тикет был открыт в баг-трекере в январе 2007, а закрыт только с релизом версии V3.5 SP6 в декабре 2014 года.

[2] Когда я пишу эти строки – актуальной версией CODESYS является вышедшая в апреле 2020 V3.5 SP18.
[3] Я не знаю, с какой именно – но точно не позже 3.5.5.0.

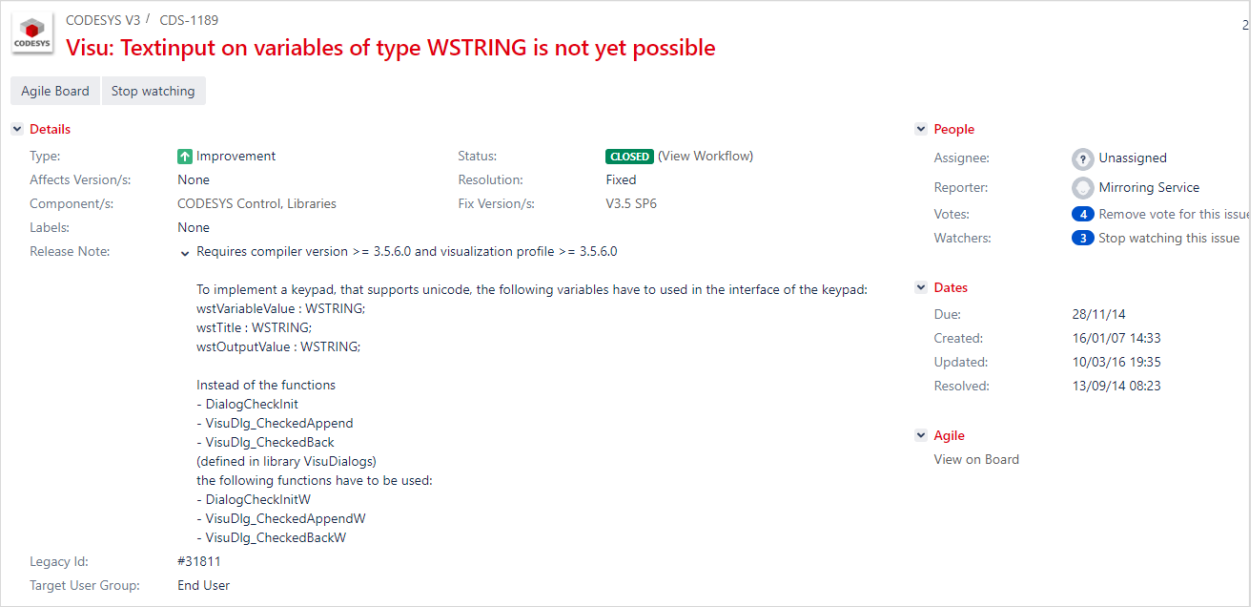


Рис. 6 – Тикет о необходимости поддержки типа WSTRING в диалоге Keypad

Поддержка типа WSTRING была добавлена – но теперь требовалось создать клавиатуру с кириллицей. Пользователи пробовали делать это и до релиза V3.5 SP6 – см., например, эту тему на форуме ОВЕН. В первые секунды прочтения она может вызвать некий диссонанс – уже во втором посте выложено видео с «русскоязычной» клавиатурой – при этом пост датирован **мартом** 2014 года, когда V3.5 SP6 еще не вышел. Казалось бы – о какой проблеме тогда вообще идет речь? Но дело в том, что в видео ввод строки происходит в переменную типа STRING – то есть отобразить ее корректно на экране панельного контроллера не получилось бы. Об этом справедливо замечает пользователь **murdemon**:

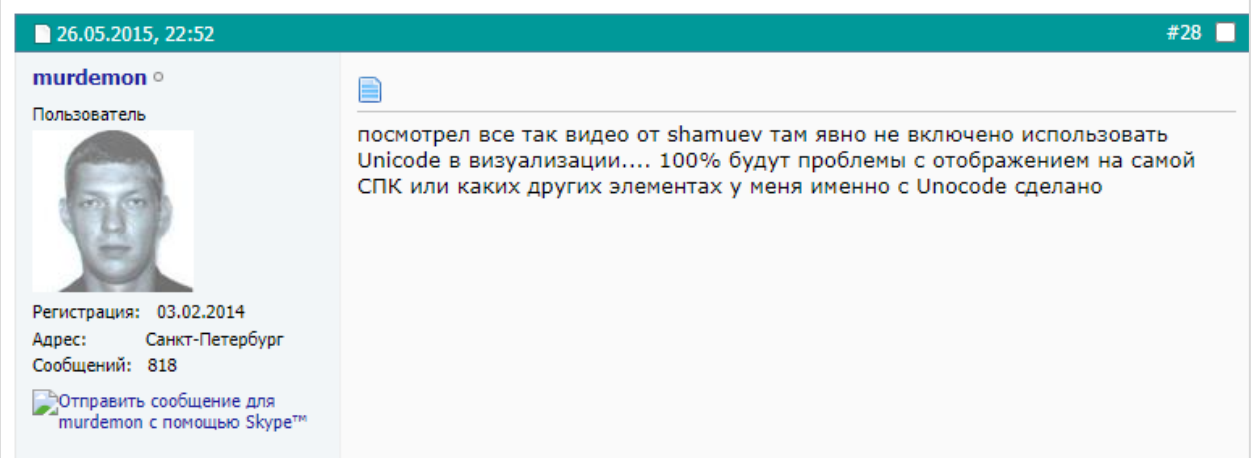


Рис. 7 – Пост пользователя murdemon

Таким образом, именно в версии V3.5 SP6 появилась возможность создания русскоязычной клавиатуры. Теперь кто-то должен был заняться ее реализацией.

2014-2019: VisuKeyboard_En-Ru и Максим Сироткин

Наши клиенты достаточно регулярно напоминали нам, что им не хватает русскоязычной клавиатуры в CODESYS. Мы в свою очередь передавали эту информацию в ПК Пролог (дистрибьютор и системный партнер CODESYS в России), а они – разработчикам CODESYS. Насколько я понимаю – в какой-то момент Пролог озвучил приблизительные сроки по закрытию тикета с рисунка 6, потому что в феврале 2014 года на вебинаре, посвященном старту продаж панельного контроллера ОВЕН СПК105, был показан такой слайд:

Рис. 8 – Анонс русскоязычной клавиатуры в феврале 2014 года

Как мы уже знаем – поддержка типа WSTRING в итоге появилась не в V3.5 SP5, а в V3.5 SP6, который вышел в декабре 2014. Но клиентов интересовала не поддержка WSTRING, а уже готовая русскоязычная клавиатура. Эту клавиатуру разработал **Максим Сироткин**, сотрудник компании ПК Пролог. Она распространялась в виде библиотеки **VisuKeyboard_En-ru**, релиз которой состоялся в конце мая 2016 года. В июне вышел CODESYS V3.5 SP9, в котором в диалогах ввода появилась поддержка курсора (это, например, позволяет выделить и удалить часть строки), поэтому библиотека была обновлена до версии **1.1.0.0**. В июле эта версия библиотеки была выложена на сайте и форуме ОВЕН (вот ссылка на пост с информацией о релизе библиотеки на форум ОВЕН).

Change History

Version	Description	Editor	Date
1.0	Release	SMM	30.05.2016
1.1	Update keyboard with support of enhanced text input with cursor (SP9).	SMM	17.06.2016
1.2	Minimized quantity of visual elements in the dialog. By E.Kislov	SMM	18.02.2019
1.3	Added big version of the keyboard. Visual polishing.	SMM	24.04.2019

Рис. 9 – Историй версий библиотеки VisuKeyboard_En-Ru (ссылка)

Рис. 10 – Отображение курсора в диалоге Keypad

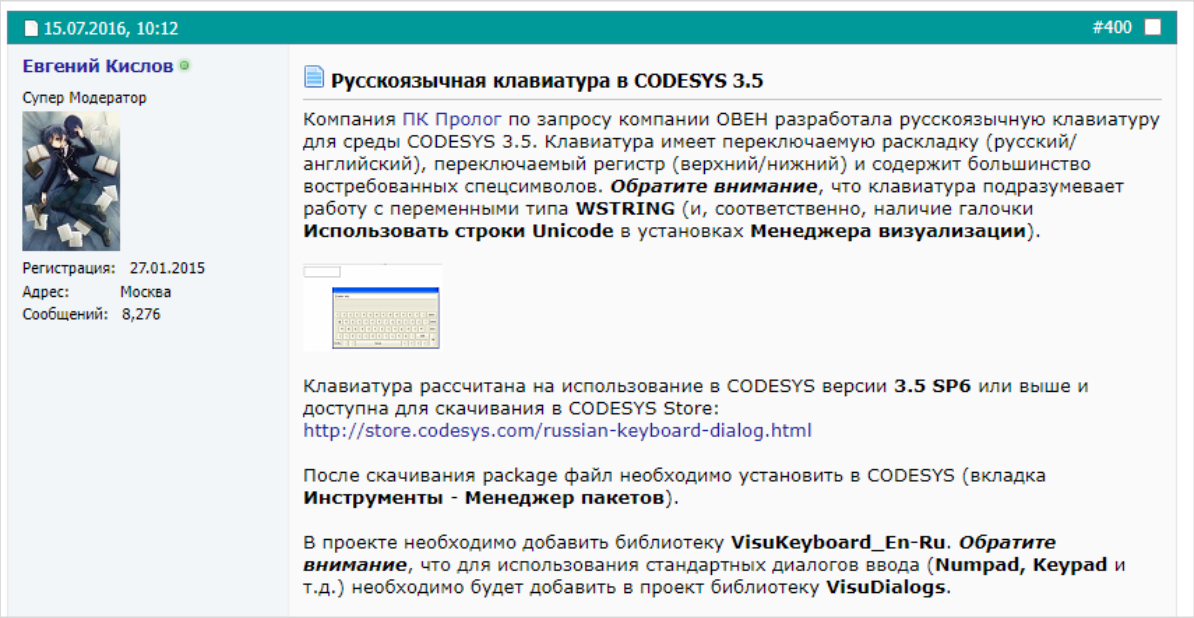
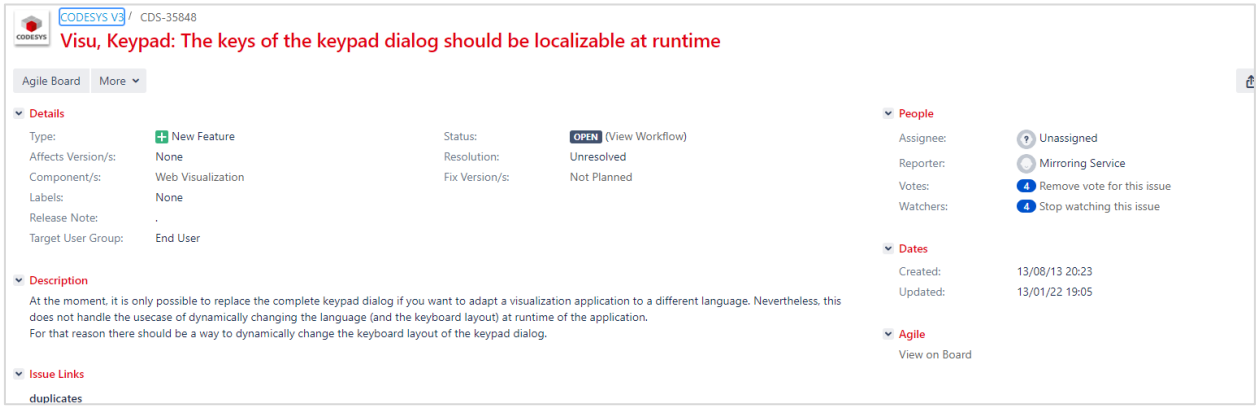


Рис. 11 – Пост с информацией о библиотеке VisuKeyboard_En-Ru на форуме ОВЕН (ссылка)

Резонный вопрос – почему от поддержки WSTRING до релиза клавиатуры прошло почти 1.5 года? Основных причины было две:

- для нас это была задача низкого приоритета, ей не уделялось слишком много времени;
- долгое время не было понимания, кто должен разработать эту клавиатуру – 3S, Пролог или ОВЕН. В баг-трекере с 2013 года висит тикет о необходимости поддержки различных раскладок экранной клавиатуры на уровне системы исполнения (в 2016 появился еще один аналогичный тикет), и, кажется, в 2015 еще было ожидание, что это будет вот-вот сделано – а зачем тогда тратить на это ресурсы? В итоге эти тикеты по состоянию на май 2022 года так и не закрыты.



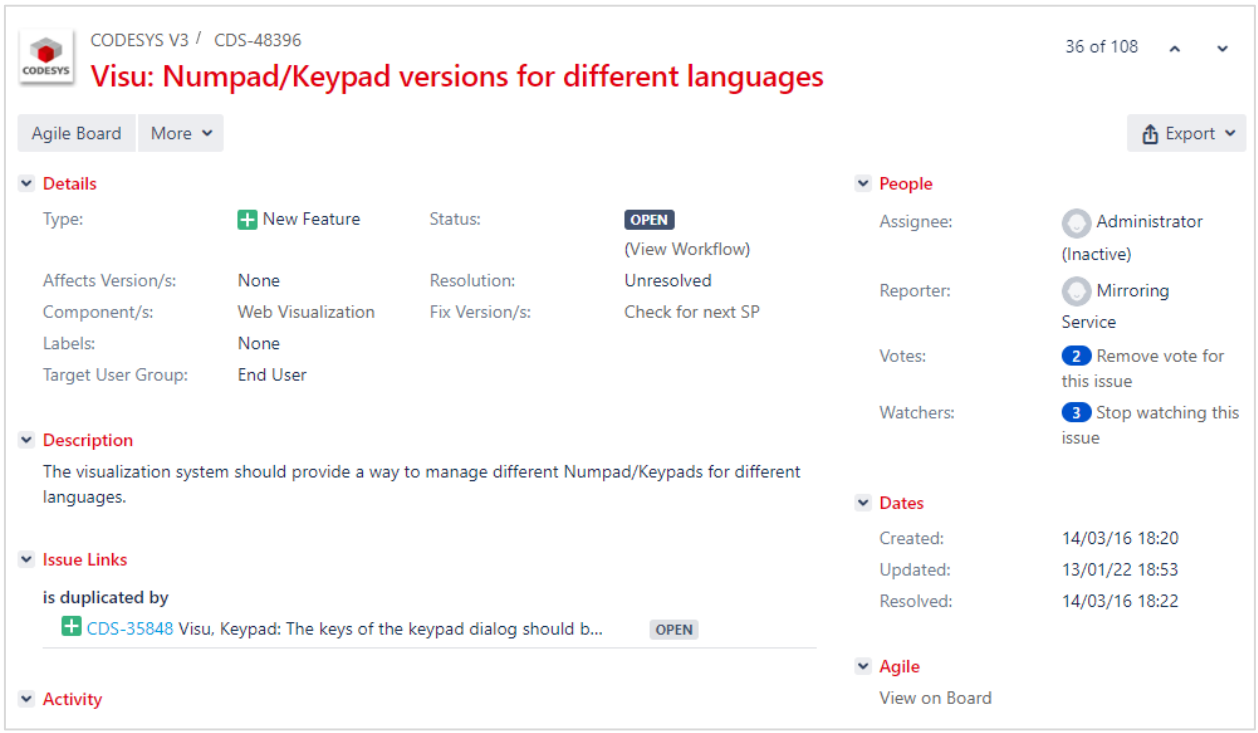


Рис. 12 – Тикеты с запросом на встроенную поддержку клавиатур с различными языками раскладок

Библиотека **VisuKeyboard_En-Ru** была выложена в CODESYS Store – она доступна там и сейчас. На протяжении следующих нескольких лет все клиенты, которым требовалась русскоязычная клавиатура, использовали библиотеку Максима. Выглядела она вот так:

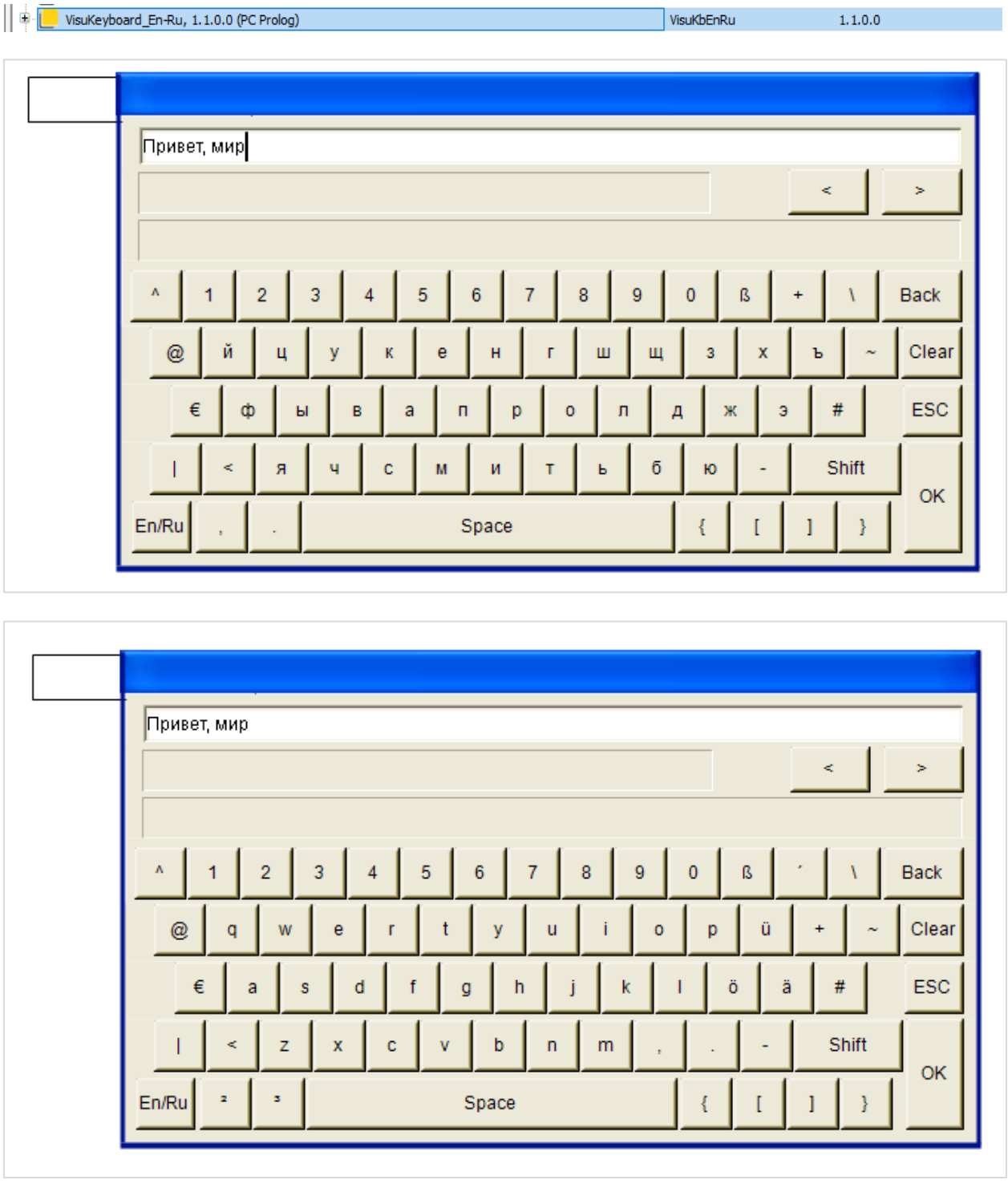


Рис. 13 – Внешний вид клавиатуры из библиотеки VisuKeyboard_En-Ru

В феврале 2019 года ко мне обратился один из клиентов, который нашел в библиотеке баг – при конкретной последовательности действий (*насколько я помню – нужно было несколько раз переключить язык и регистр символов*) после закрытия клавиатуры на экране оставались графические артефакты – несколько прямоугольников цвета фона клавиатуры.

Я изучил эту проблему и выяснил, что она связана с реализацией клавиатуры – каждый язык и регистр был представлен отдельным слоем кнопок. Фактически клавиатура представляла собой 4 наложенные друг на друга клавиатуры (англоязычная с верхним регистром/англоязычная с нижним регистром/русскоязычная с верхним регистром/русскоязычная с нижним регистром), и при изменении языка или регистра происходило переключение видимости больших групп элементов. Когда я убрал с клавиатуры примерно половину кнопок – то проблема перестала проявляться.

Я связался с Максимом, рассказал ему о баге и своих опытах. Максим ответил, что, к сожалению, в ближайшие пару месяцев не сможет заняться его исправлением. Поскольку клиент был заинтересован в оперативном решении проблемы, то я решил исправить ее своими силами. Поскольку проблема была связана с переключением невидимости большого числа элементов – я предпочел вообще отказаться от невидимости и обойтись «однослойной» клавиатурой. Для каждой кнопки я создал список текстов с 4 записями (EN/RU, верхний/нижний регистр) и добавил в библиотеку перечисление с этими вариантами раскладок. После этого оставалось модифицировать код каждой кнопки, добавив в него обработку экземпляра этого перечисления, а к кнопкам En/Ru и Shift привязать код для изменения значений этого экземпляра.

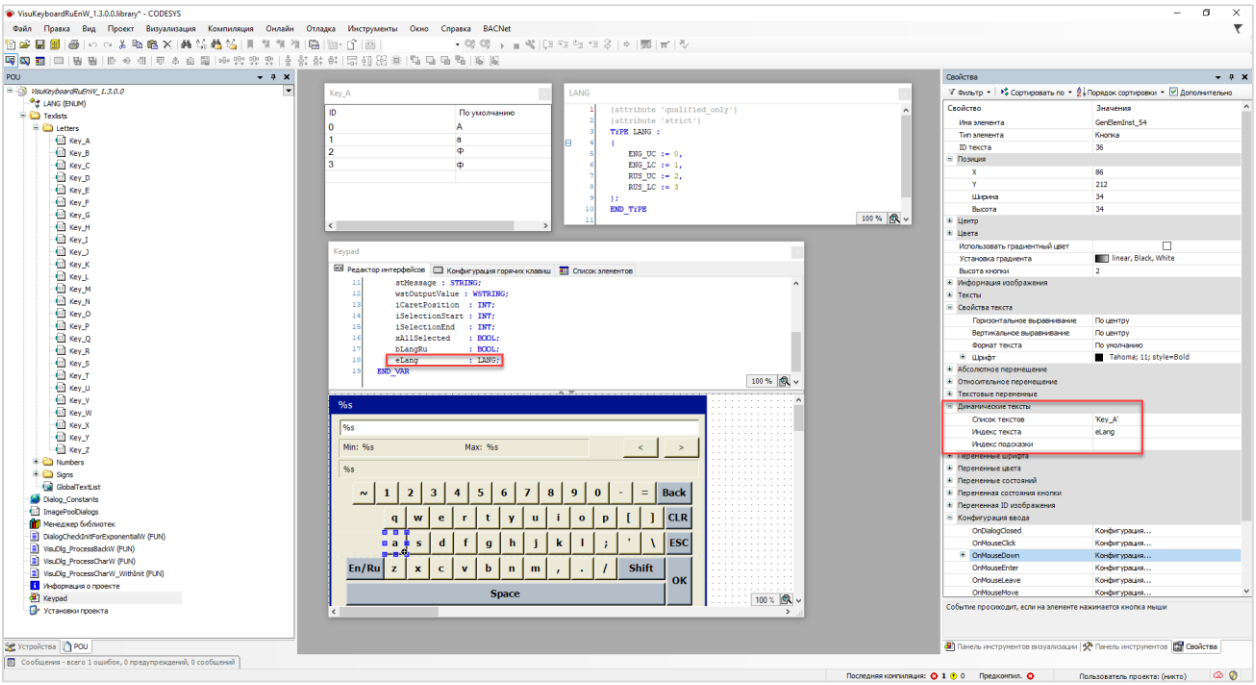


Рис. 14 – Переключение раскладки клавиатуры с помощью списков текстов

Это решило проблему с артефактами. Из-за того, что в библиотеку добавилось ≈50 списков текстов, то проект с этой библиотекой стал загружаться в контроллер секунд на 5 медленнее – но это не кажется мне серьезным недостатком. Еще я, держа в памяти обратную связь от клиентов, увеличил размер подписей клавиш, а также выделил «функциональные» клавиши своим цветом.

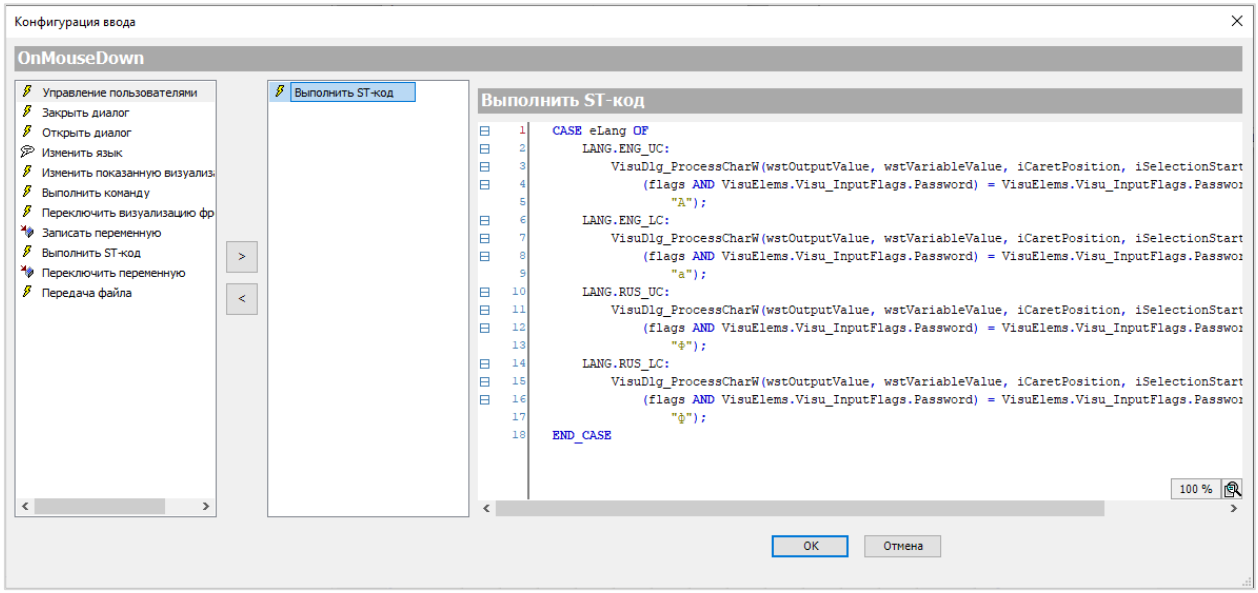


Рис. 15 – Код, привязанный к кнопке клавиатуры

18 февраля 2019 я отправил Максиму мою версию с исправлениями. Он сразу предупредил, что обновить ее в CODESYS Store оперативно не получится, поэтому в те же дни я выложил ее на сайте ОВЕН. Я сохранил библиотеку под версией **1.3.0.0**, потому что **1.2.0.0** использовал для своих тестовых сборок и решил таким образом избежать возможной путаницы.

По истории версий (см. рис. 9) может сложиться впечатление, что «моя» версия была выложена в CODESYS Store под версией **1.2.0.0**, но это не так. В реальности произошло следующее – Максим добавил в мою версию диалог клавиатуры с увеличенным размером («KeypadBig») и выложил ее в CODESYS Store в конце апреля.

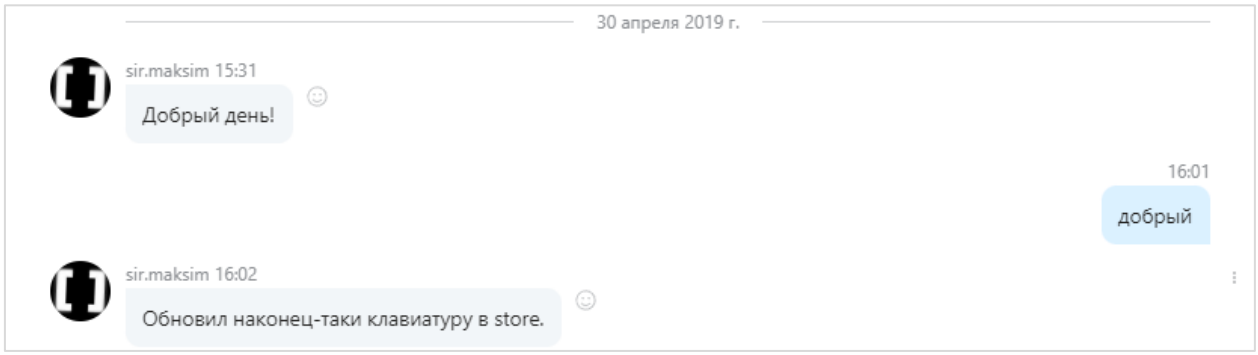


Рис. 16 – Сообщение от Максима о релизе версии библиотеки 1.3.0.0 в CODESYS Store

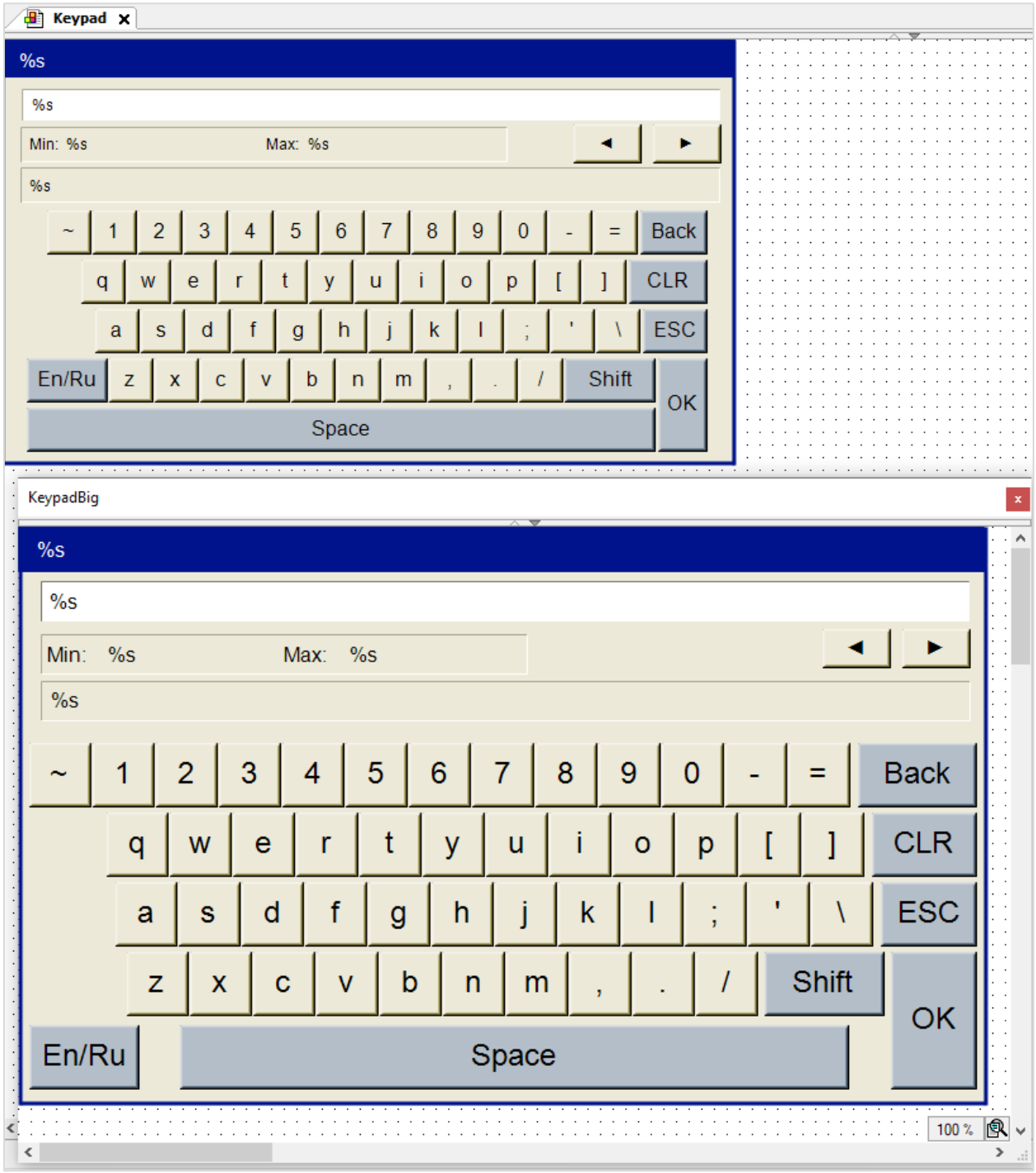


Рис. 17 – Внешний вид диалогов Keypad и KeypadBig из библиотеки VisuKeyboardEnRu версии 1.3.0.0

На этом моменте развитие библиотеки **VisuKeyboardEnRu** закончилось. Следующий этап истории начался спустя почти год – весной 2020. Но сначала хотелось бы чуть-чуть отвлечься и рассказать о диалогах управления пользователями.

2015: VisuUsersMgmtDialogs и 3S

В CODESYS V3.5 SP2 было добавлено управление пользователями визуализации. Этот механизм позволяет разграничить права операторов на доступ к графическим элементам – в случае недостаточного уровня прав элемент будет невидимым или видимым, но неактивным. Для авторизации и настройки пользователей были созданы 3 диалога, которые вошли в состав библиотеки **VisuUserManagement**:

- **VUM_Login** – диалог авторизации;
- **VUM_ChangePassword** – диалог изменения пароля;
- **VUM_UserManagement** – диалог управления пользователями (просмотр списка пользователей, добавление новых пользователей, удаление или блокировка существующих и т.д.).

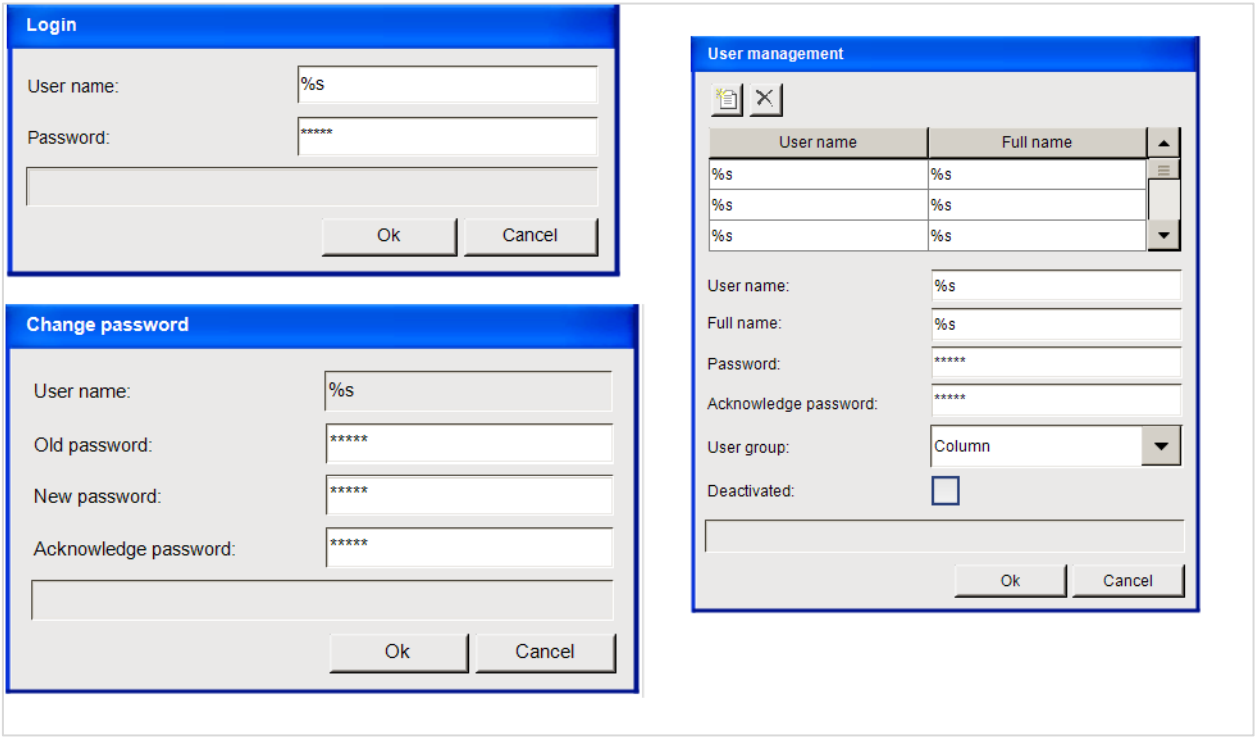


Рис. 18 – Диалоги управления пользователями

Внешний вид диалогов был таким же аскетичным, как и у диалогов библиотеки **VisuDialogs**, и, кроме того, для них отсутствовала поддержка мультиязычности. Поэтому летом 2013 года, вскоре после выхода CODESYS V3.5 SP3, в баг-трекер был добавлен тикет, поясняющий необходимость возможность редактирования этих диалогов пользователями:

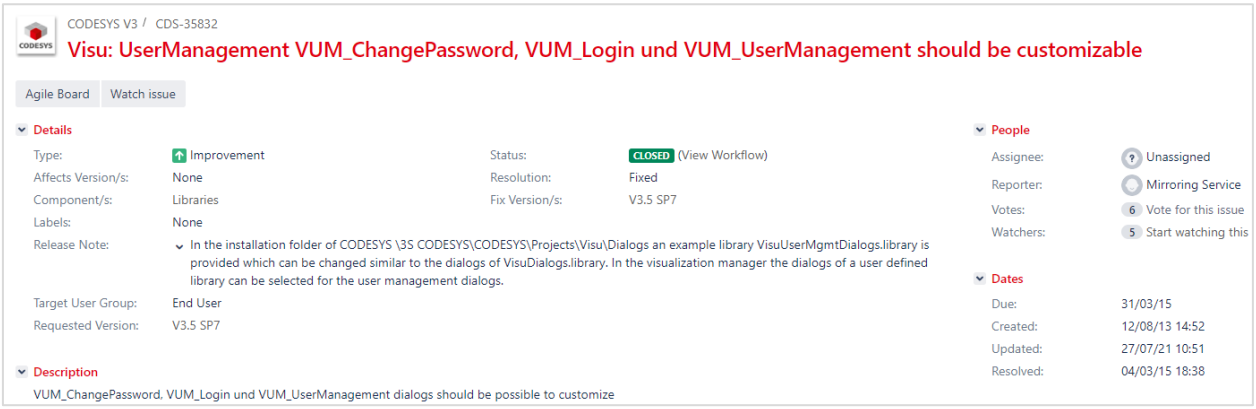


Рис. 19 – Тикет с просьбой добавления возможности редактирования диалогов управления пользователями

Тикет был закрыт весной 2015 года в процессе работы над версией CODESYS V3.5 SP7 (выход которой состоялся тем же летом). Диалоги управления пользователями были выделены в отдельную библиотеку **VisuUserMgmtDialogs**, которая, как и **VisuDialogs**, имела открытые исходники: ее файл (.library) доступен в директории установки CODESYS по пути **…\CODESYS\Projects\Visu\Dialogs**. Это предоставило возможность пользователям изменять внешний вид данных диалогов.

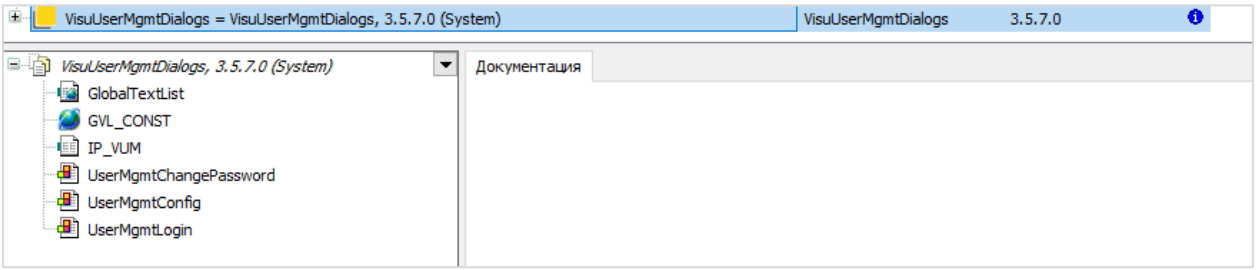


Рис. 20 – Состав библиотеки VisuUserMgmtDialogs

2019-2022: OwenVisuDialogs и Владислав Зинько

В первый раз я общался с Владом в июле 2017 года – у него был вопрос по панелям оператора СП3хх. История нашего чата в Skype показывает, что в следующий раз мы разговаривали только через год – в июле 2018, а более-менее регулярное общение у нас началось зимой 2019 года, когда Влад занимался тестированием шаблонов модулей Mx210 для CODESYS V3.5.

Весной 2019 Влад уволился, а осенью вернулся в компанию, сменив при этом должность и отдел. Теперь он занимался разработкой готовых проектов в CODESYS V3.5 для заказчиков индивидуальных решений на базе нашего оборудования, а поскольку я в основном отвечал как раз за устройства с CODESYS V3.5 на борту – то мы стали переписываться довольно регулярно.

5 марта 2020 в нашем диалоге мелькает такое сообщение:

> v.zinko 05.03.2020, 20:11:24
> я, к слову, сейчас клаву перерисовываю как в новом конфигураторе нампад давненько нарисовал, выглядит более футуристично)

Под «новым конфигуратором» имеется в виду экранный конфигуратор панельного контроллера СПК1хх [M01], появившийся в прошивке **1.2.xxxx.xxxx** (первой прошивке с OpenWRT). Клавиатуры в экранном конфигураторе выглядели вот так:

Рис. 21 – Внешний вид клавиатур экранного конфигуратора СПК

Я сразу подумал о том, что такие клавиатуры были бы интересны значительному числу наших клиентов. Практически сразу возникает идея оформить их как отдельную библиотеку. Уже в конце апреля в наших с Владом диалогах появляются мысли насчет перерисовки в таком же стиле диалогов управления пользователями и диалога выбора файлов. К этому моменту уже сложился основной концепт библиотеки:

- библиотека будет универсальной – она будет включать в себя все диалоги из **VisuDialogs**, **VisuKeyboard_En-Ru** и **VisuUserMgmtDialogs**, перерисованные в стилистике экранного конфигуратора СПК;
- все диалоги будут переведены на русский язык;
- все диалоги из стандартных библиотек также будут включены в новую библиотеку, чтобы позволить пользователю не добавлять их отдельно, если по каким-то причинам ему требуется их использовать (например, для обратной совместимости с уже существующими проектами);
- диалоги должны поддерживать возможность изменения цветовой палитры для возможности переключения между «светлой» и «темной» темой и т.д.;
- как и стандартные библиотеки диалогов – новая библиотека будет доступна в исходниках, чтобы пользователь мог адаптировать ее под себя.

Названия у библиотеки еще не было – в переписке мы называли ее просто «библиотека диалогов».

Еще один момент, который вспоминается в процессе обсуждения библиотеки – извечная проблема с вводом в визуализации переменных типа TIME. Долгое время их можно было вводить только через Keypad с помощью МЭК-синтаксиса в стиле T#22h12m30s. Это, естественно, было крайне неудобным для операторов. В версии V3.5 SP12 появилась поддержка ввода значений времени в понятном для людей формате (22:12:30):

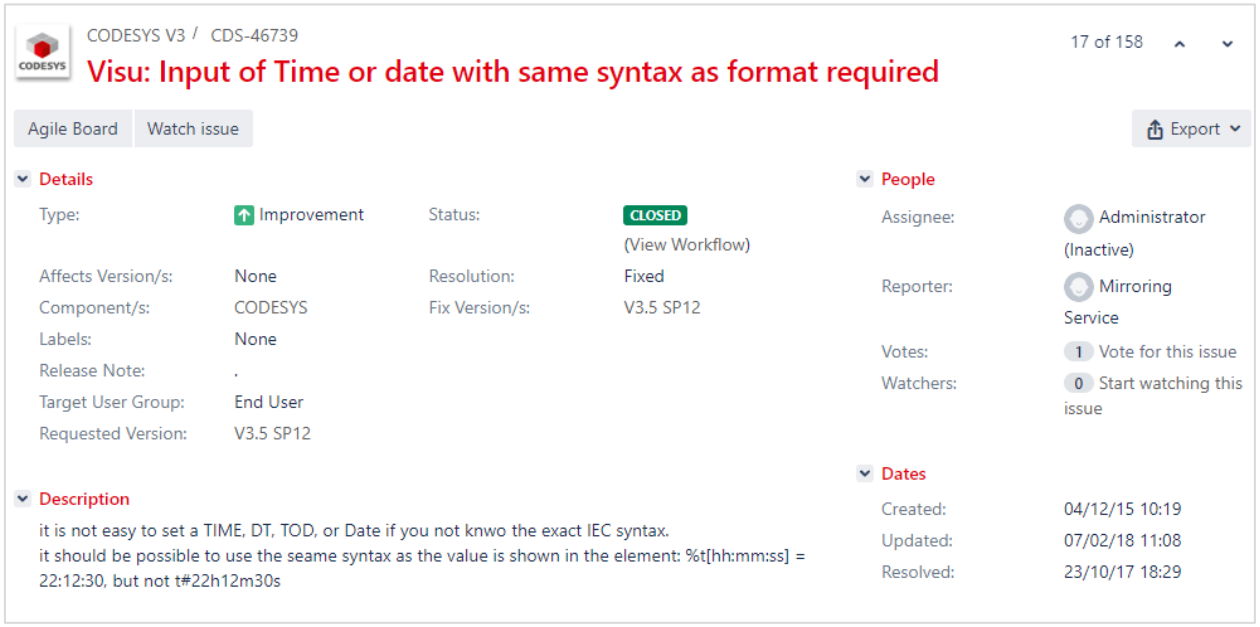


Рис. 22 – Тикет о поддержке «человеческого» синтаксиса для ввода времени в визуализации

Но проблема в том, что в стандартном диалоге Numpad нет символа двоеточия, а открывать каждый раз Keypad для ввода такого значения – довольно неудобно. Это тоже отражено в баг-трекере, но соответствующий тикет до сих пор не выполнен.

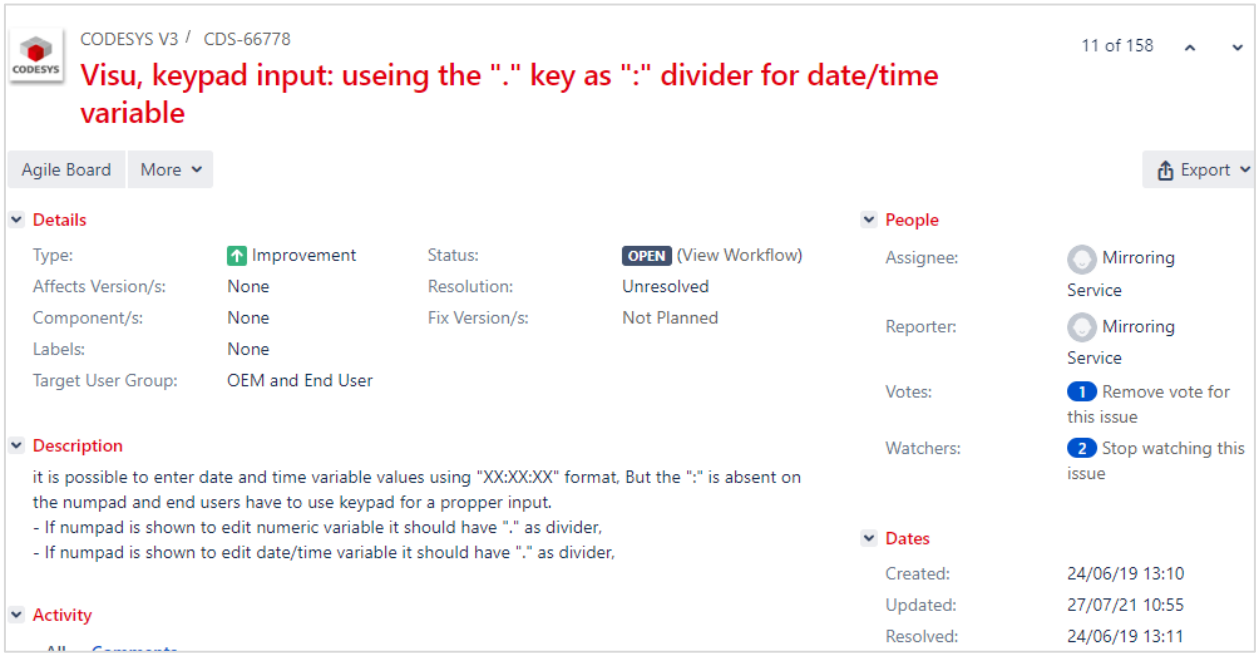


Рис. 23 – Тикет о добавлении символа двоеточия в Numpad

В итоге мы с Владом решили, что в его версии Numpad'а двоеточие обязано присутствовать. В версии V3.5 SP17 добавили достаточно специфический синтаксис для ввода значений времени с помощью точки и новых спецификаторов времени – например, спецификатор %t[HH2:mm2] с введенным значением 20.25 будет соответствовать T#20h15m (.25 – это «четверть часа»). Мне до сих пор кажется, что вариант с добавлением двоеточия гораздо разумнее этой странной доработки.

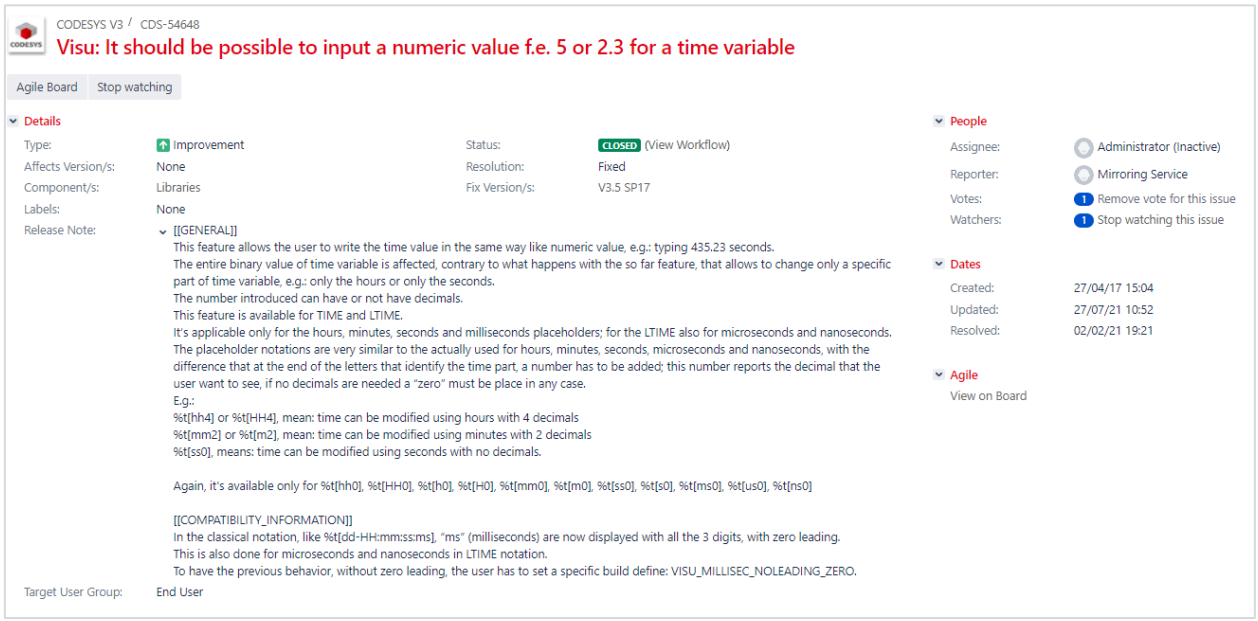


Рис. 24 – Тикет о добавлении новых спецификаторов времени для ввода значений типа TIME/LTIME с использованием точки

Для Влада разработка библиотеки диалогов была фоновой задачей, и на нее постоянно не оставалось времени из-за его загруженности основными проектами. В итоге за весь 2020 год вся запланированная по библиотеке работа так и не была завершена. В декабре 2020 Влад перешел в отдел, в котором работал я – и первой задачей, которой я предложил ему заняться, было завершение работ по библиотеке. В этот момент появляется необходимость выбрать библиотеке «официальное» название – и, недолго думая, мы называем ее **OwenVisuDialogs**.

В течение января 2021 Влад ударными темпами закончил библиотеку, а также создал тестовое ПО и пример для пользователей. Первая версия библиотеки получила обозначение **3.5.14.1**, так как была создана в CODESYS V3.5 SP14 Patch 3 (*на тот момент именно в этой версии среды программировались наши контроллеры*).

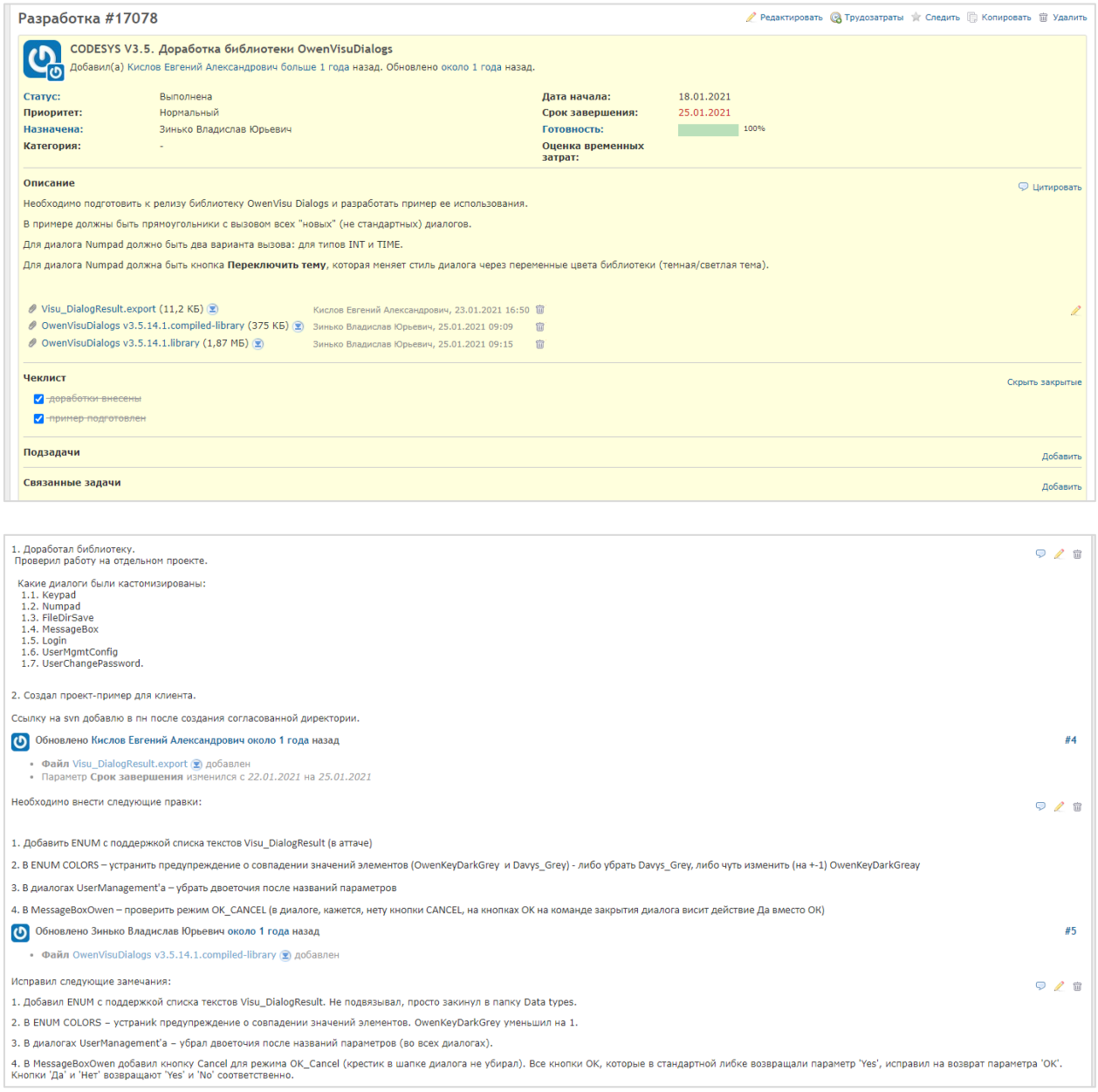


Рис. 25 – Задача в Redmine по релизу первой версии библиотеки OwenVisuDialogs

27 января я создал тему о библиотеке на форуме ОВЕН. Примерно в те же дни она появилась на сайте (про нее даже была короткая новость), а 4 февраля – была опубликована новость в группе VK.

Вот так выглядели диалоги, присутствовавшие в первой версии библиотеки:

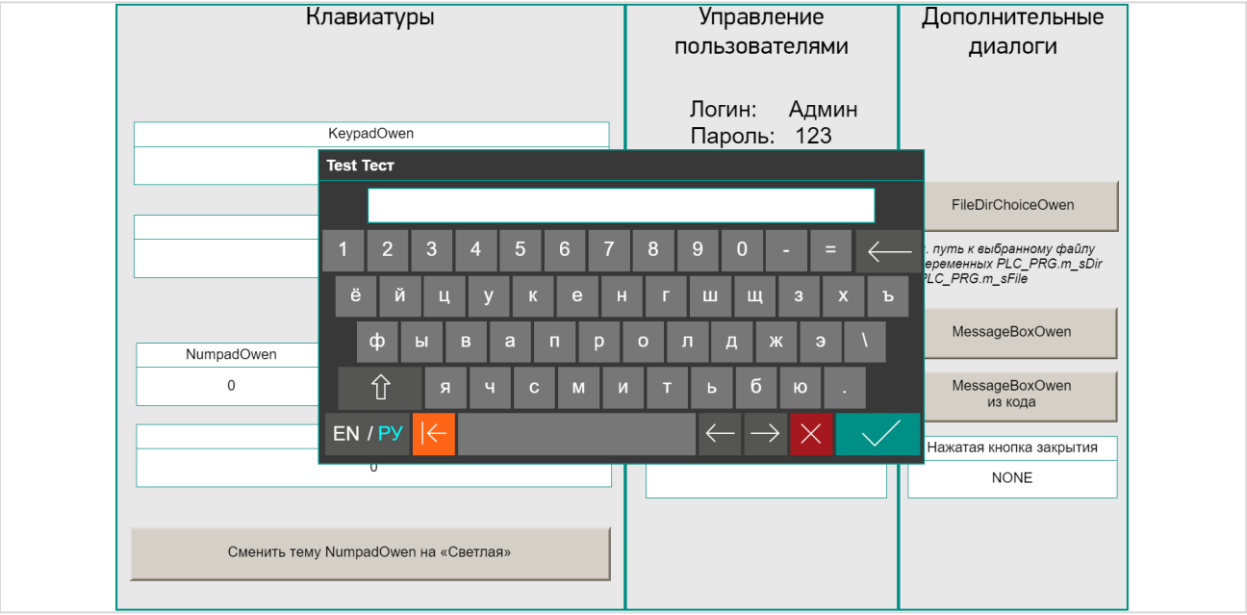


Рис. 26 – Внешний вид диалога KeypadOwen

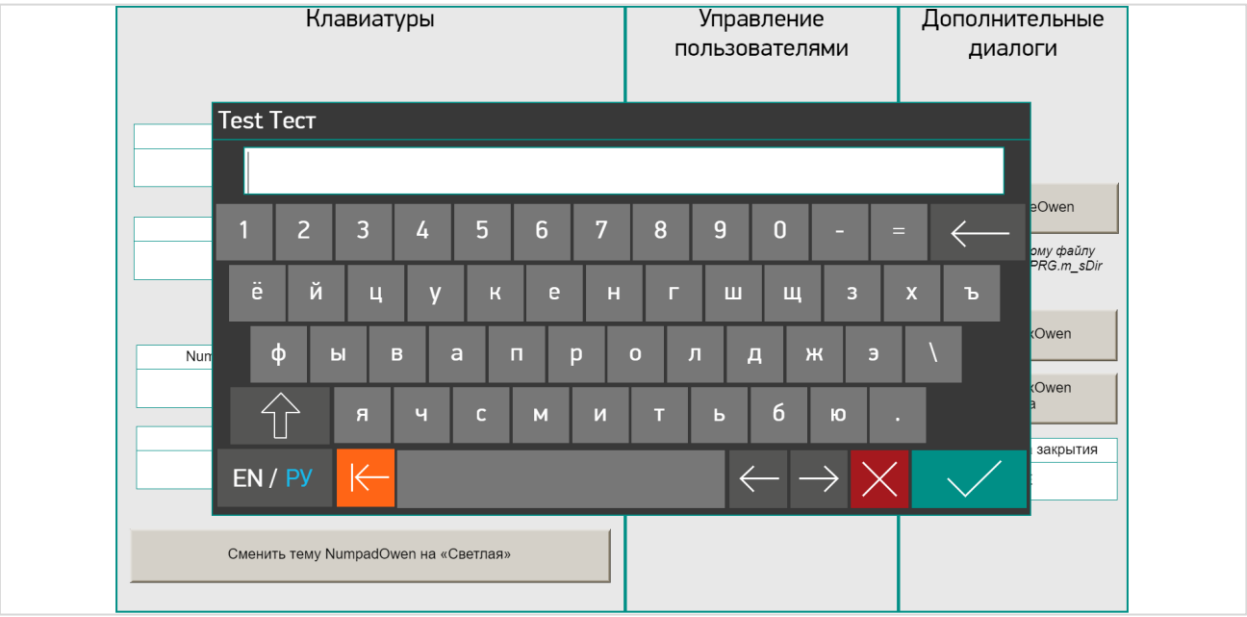


Рис. 27 – Внешний вид диалога KeypadOwenBig

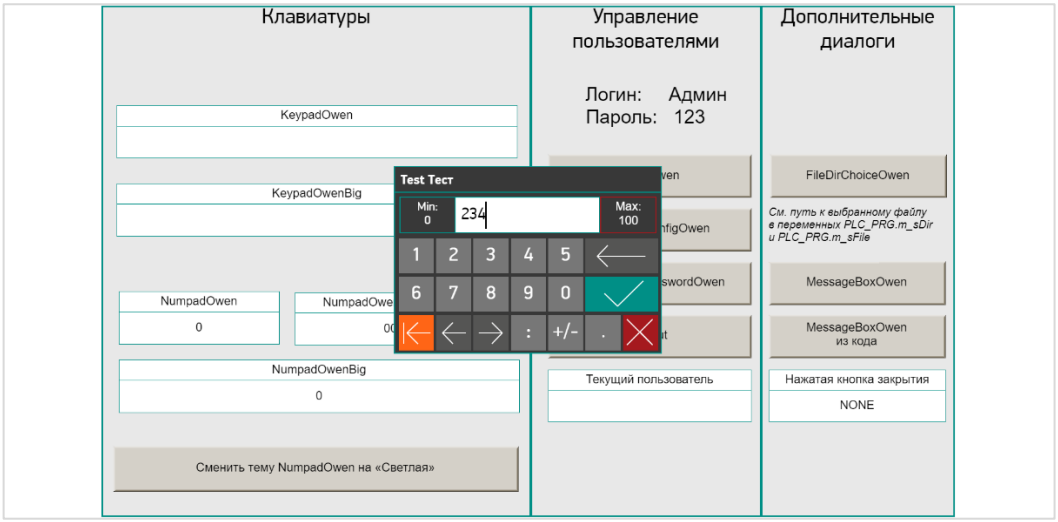


Рис. 28 – Внешний вид диалога NumpadOwen

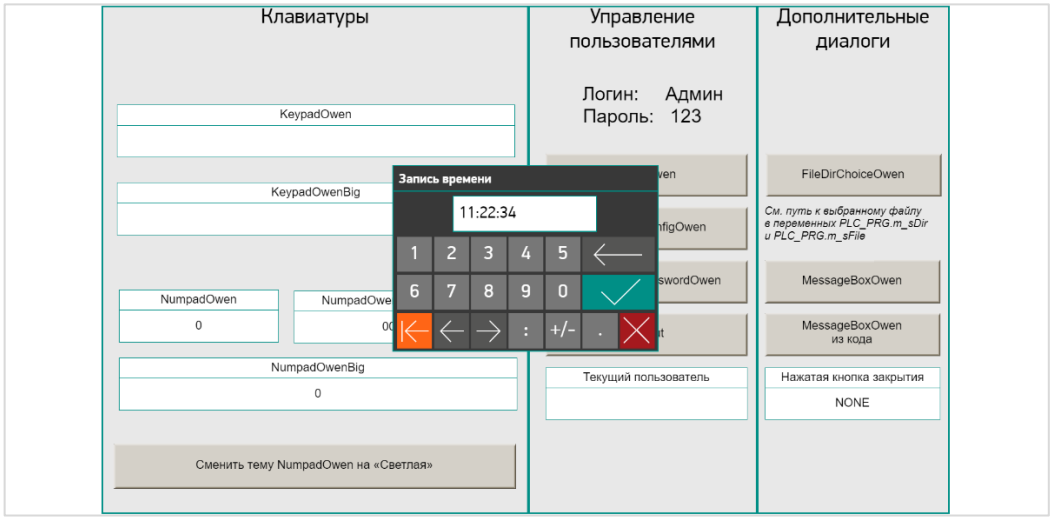


Рис. 29 – Внешний вид диалога NumpadOwen с примером ввода времени

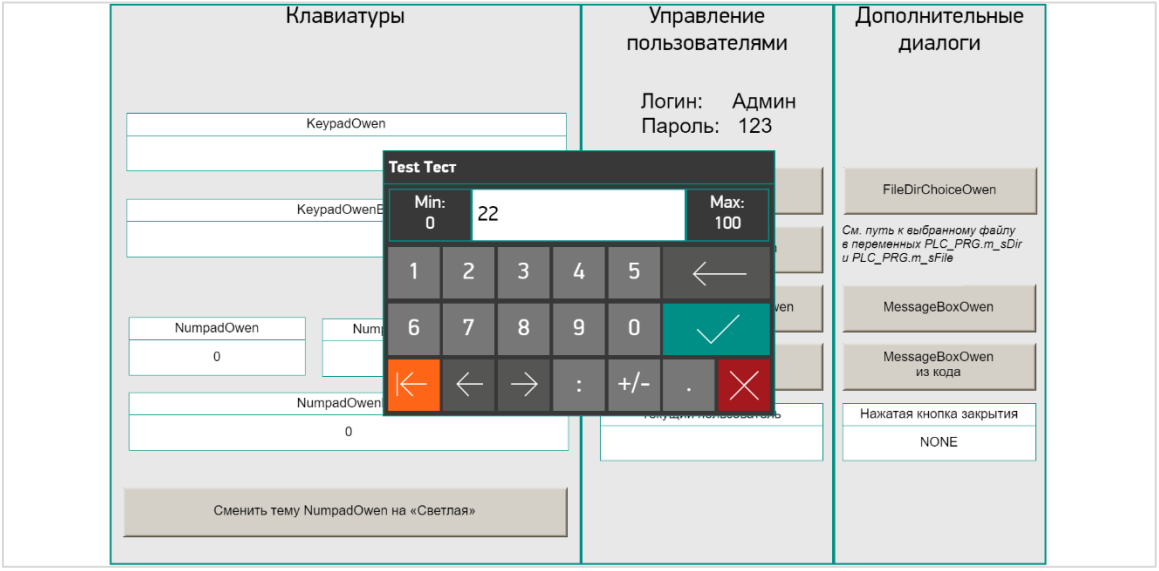


Рис. 30 – Внешний вид диалога NumpadOwenBig

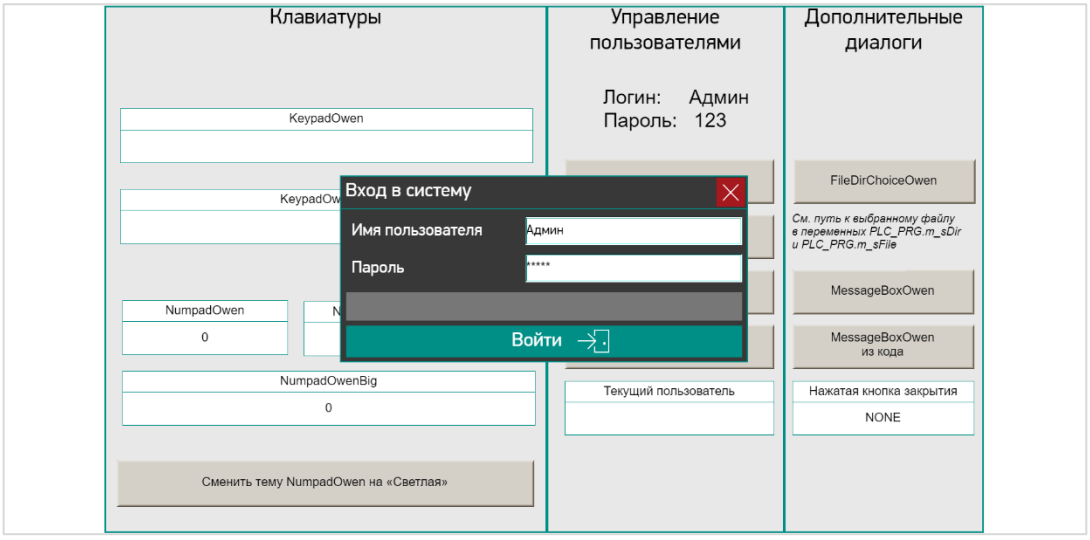


Рис. 31 – Внешний вид диалога LoginOwen

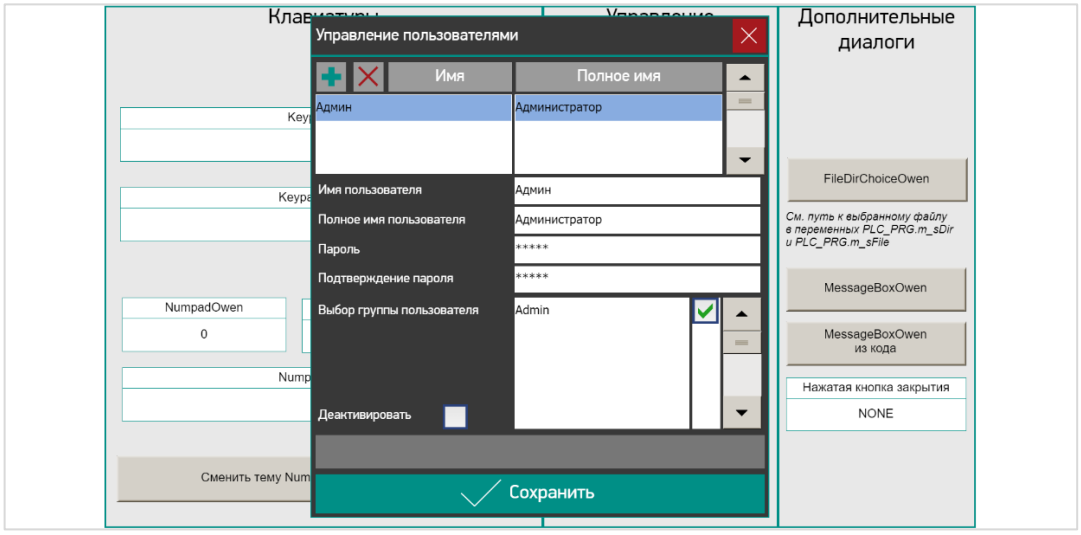


Рис. 32 – Внешний вид диалога UserChangePasswordOwen

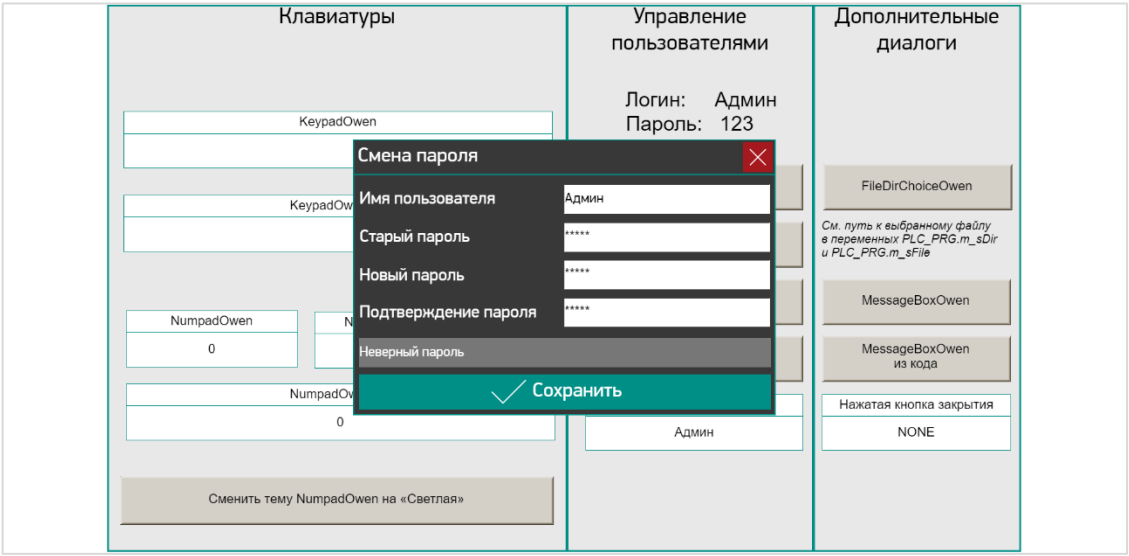


Рис. 33 – Внешний вид диалога UserMgmtConfigOwen

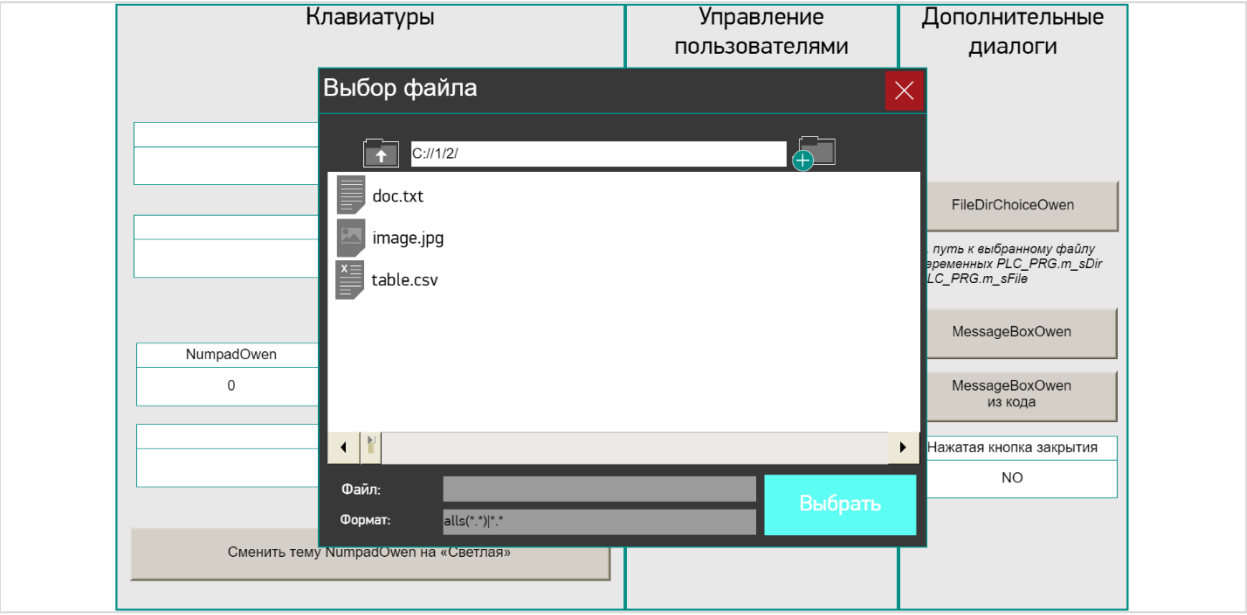


Рис. 34 – Внешний вид диалога FileDirChoiceOwen

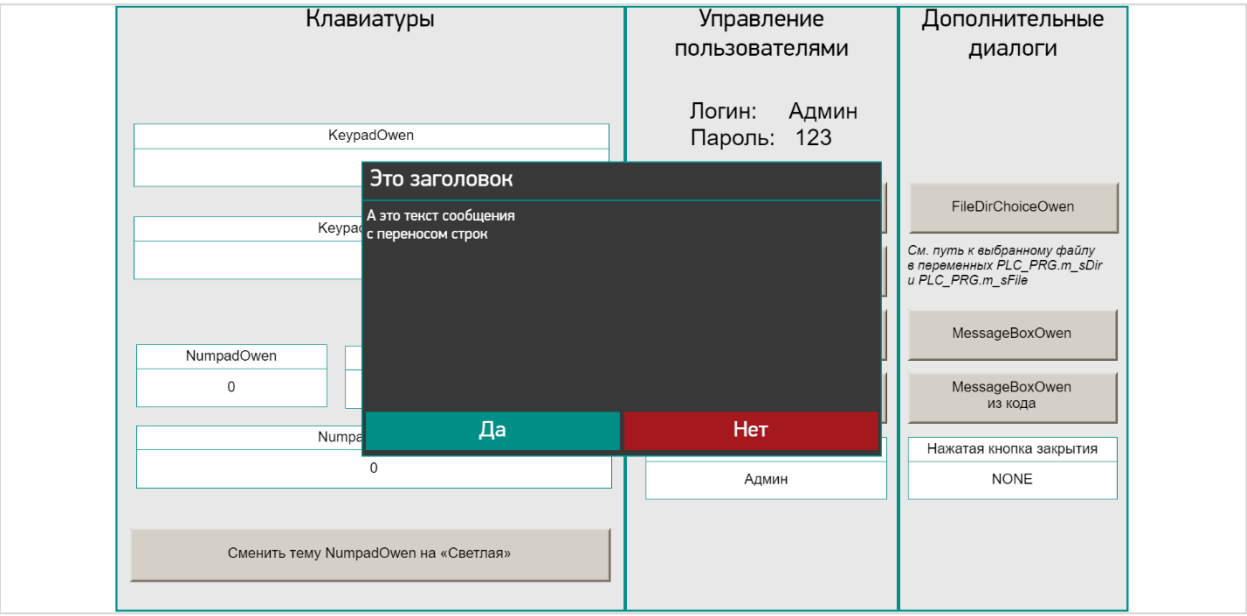


Рис. 35 – Внешний вид диалога MessageBoxOwen

Пользователи сразу начинают применять библиотеку в своих проектах, и мы получаем от них несколько пожеланий и замечаний. В конце марта 2021 я формирую список доработок, и Влад приступает к работе над версией **3.5.14.2**.

Самой крупной доработкой являлось создание нового диалога для задания даты и времени. Я часто замечал, что пользователи в своих проектах делают отдельный экран для настроек системного времени – примерно в таком стиле:

Рис. 36 – Пример пользовательского экрана установки системного времени

Соответственно, у нас была возможность облегчить им жизнь, создав уже готовый диалог для этой задачи. Изменение системного времени в наших контроллерах производится с помощью узла **OwenRTC** в дереве проекта – и если проект создается на основе шаблона, то к каналам этого узла уже привязаны переменные. Пользователю остается только указать их в вызове нового диалога – и он получает готовый графический интерфейс для изменения системного времени.

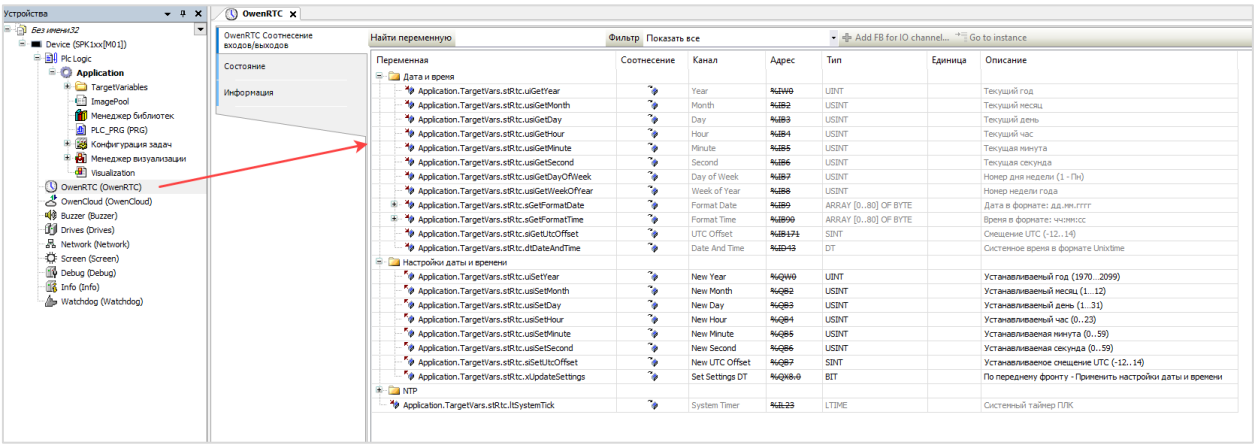


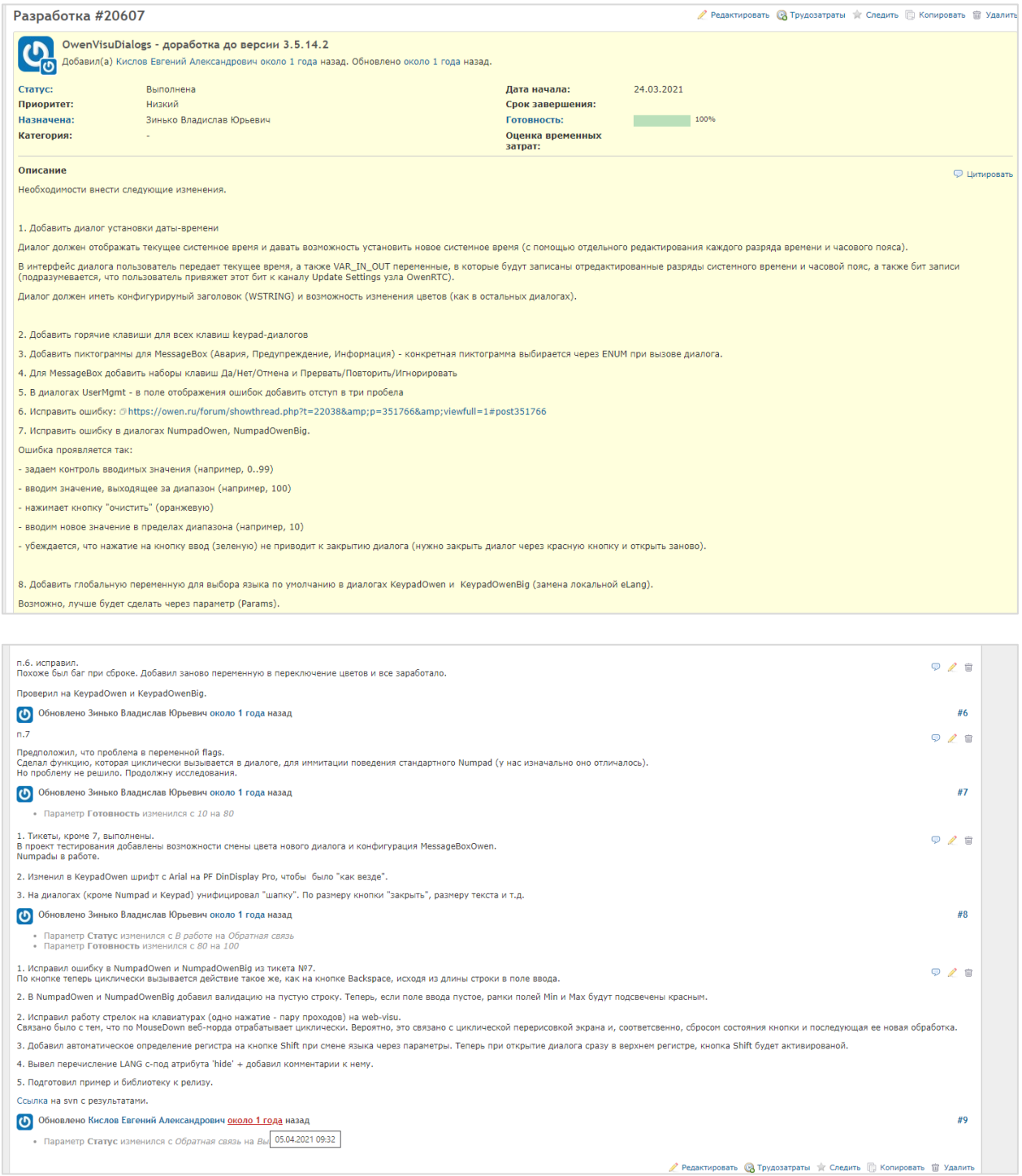
Рис. 36 – Узел OwenRTC в дереве проекта

У меня также было навязчивое желание прикрутить ко всем клавиатурам «горячие клавиши» (в CODESYS так называется привязка аппаратной клавиши к какому-либо элементу на экране визуализации) – не то чтобы клиенты часто ими пользовались, но мне хотелось по максимуму использовать встроенный функционал; да и всё же иногда вопросы по их поводу мы получали – так что кому-то они точно были нужны.

Еще очень хотелось прикрутить к MessageBox иконки (как в Windows) – чтобы сразу можно было отличить сообщение о тревоге от информационной подсказки.

Ряд пожеланий высказали и сами пользователи – например, им требовалась возможность выбрать язык, с которым по умолчанию будет открываться диалог Keypad.

В начале апреля 2022 работы над версией **3.5.14.2** завершаются. 5 апреля я обновляю тему на форуме.



Разработка #20607

Редактировать | Трудозатраты | Следить | Копировать | Удалить

OwenVisuDialogs - доработка до версии 3.5.14.2

Добавил(а) Кислов Евгений Александрович около 1 года назад. Обновлено около 1 года назад.

Статус:	Выполнена	**Дата начала:**	24.03.2021
Приоритет:	Низкий	**Срок завершения:**	
Назначена:	Зинько Владислав Юрьевич	**Готовность:**	100%
Категория:	-	**Оценка временных затрат:**	

Описание — Цитировать

Необходимости внести следующие изменения.

1. Добавить диалог установки даты-времени

Диалог должен отображать текущее системное время и давать возможность установить новое системное время (с помощью отдельного редактирования каждого разряда времени и часового пояса).

В интерфейс диалога пользователь передает текущее время, а также VAR_IN_OUT переменные, в которые будут записаны отредактированные разряды системного времени и часовой пояс, а также бит записи (подразумевается, что пользователь привяжет этот бит к каналу Update Settings узла OwenRTC).

Диалог должен иметь конфигурируемый заголовок (WSTRING) и возможность изменения цветов (как в остальных диалогах).

2. Добавить горячие клавиши для всех клавиш keypad-диалогов

3. Добавить пиктограммы для MessageBox (Авария, Предупреждение, Информация) - конкретная пиктограмма выбирается через ENUM при вызове диалога.

4. Для MessageBox добавить наборы клавиш Да/Нет/Отмена и Прервать/Повторить/Игнорировать

5. В диалогах UserMgmt - в поле отображения ошибок добавить отступ в три пробела

6. Исправить ошибку: https://owen.ru/forum/showthread.php?t=22038&p=351766&viewfull=1#post351766

7. Исправить ошибку в диалогах NumpadOwen, NumpadOwenBig.

Ошибка проявляется так:

- задаем контроль вводимых значения (например, 0..99)
- вводим значение, выходящее за диапазон (например, 100)
- нажимает кнопку "очистить" (оранжевую)
- вводим новое значение в пределах диапазона (например, 10)
- убеждается, что нажатие на кнопку ввод (зеленую) не приводит к закрытию диалога (нужно закрыть диалог через красную кнопку и открыть заново).

8. Добавить глобальную переменную для выбора языка по умолчанию в диалогах KeypadOwen и KeypadOwenBig (замена локальной eLang).

Возможно, лучше будет сделать через параметр (Params).

п.6. исправил.
Похоже был баг при сброке. Добавил заново переменную в переключение цветов и все заработало.

Проверил на KeypadOwen и KeypadOwenBig.

Обновлено Зинько Владислав Юрьевич около 1 года назад #6

п.7

Предположил, что проблема в переменной flags.
Сделал функцию, которая циклически вызывается в диалоге, для иммитации поведения стандартного Numpad (у нас изначально оно отличалось).
Но проблему не решило. Продолжну исследования.

Обновлено Зинько Владислав Юрьевич около 1 года назад #7

- Параметр **Готовность** изменился с *10* на *80*

1. Тикеты, кроме 7, выполнены.
В проект тестирования добавлены возможности смены цвета нового диалога и конфигурация MessageBoxOwen.
Numpady в работе.

2. Изменил в KeypadOwen шрифт с Arial на PF DinDisplay Pro, чтобы было "как везде".

3. На диалогах (кроме Numpad и Keypad) унифицировал "шапку". По размеру кнопки "закрыть", размеру текста и т.д.

Обновлено Зинько Владислав Юрьевич около 1 года назад #8

- Параметр **Статус** изменился с *В работе* на *Обратная связь*
- Параметр **Готовность** изменился с *80* на *100*

1. Исправил ошибку в NumpadOwen и NumpadOwenBig из тикета №7.
По кнопке теперь циклически вызывается действие такое же, как на кнопке Backspace, исходя из длины строки в поле ввода.

2. В NumpadOwen и NumpadOwenBig добавил валидацию на пустую строку. Теперь, если поле ввода пустое, рамки полей Min и Max будут подсвечены красным.

2. Исправил работу стрелок на клавиатурах (одно нажатие - пару проходов) на web-visu.
Связано было с тем, что по MouseDown веб-морда отрабатывает циклически. Вероятно, это связано с циклической перерисовкой экрана и, соответственно, сбросом состояния кнопки и последующая ее новая обработка.

3. Добавил автоматическое определение регистра на кнопке Shift при смене языка через параметры. Теперь при открытие диалога сразу в верхнем регистре, кнопка Shift будет активированой.

4. Вывел перечисление LANG с-под атрибута 'hide' + добавил комментарии к нему.

5. Подготовил пример и библиотеку к релизу.

Ссылка на svn с результатами.

Обновлено Кислов Евгений Александрович около 1 года назад #9

- Параметр **Статус** изменился с *Обратная связь* на *Вы* 05.04.2021 09:32

Редактировать | Трудозатраты | Следить | Копировать | Удалить

Рис. 37 – Задача в Redmine по релизу версии 3.5.14.2 библиотеки OwenVisuDialogs

Вот так выглядит диалог для изменения системного времени, добавленный в этой версии библиотеки:

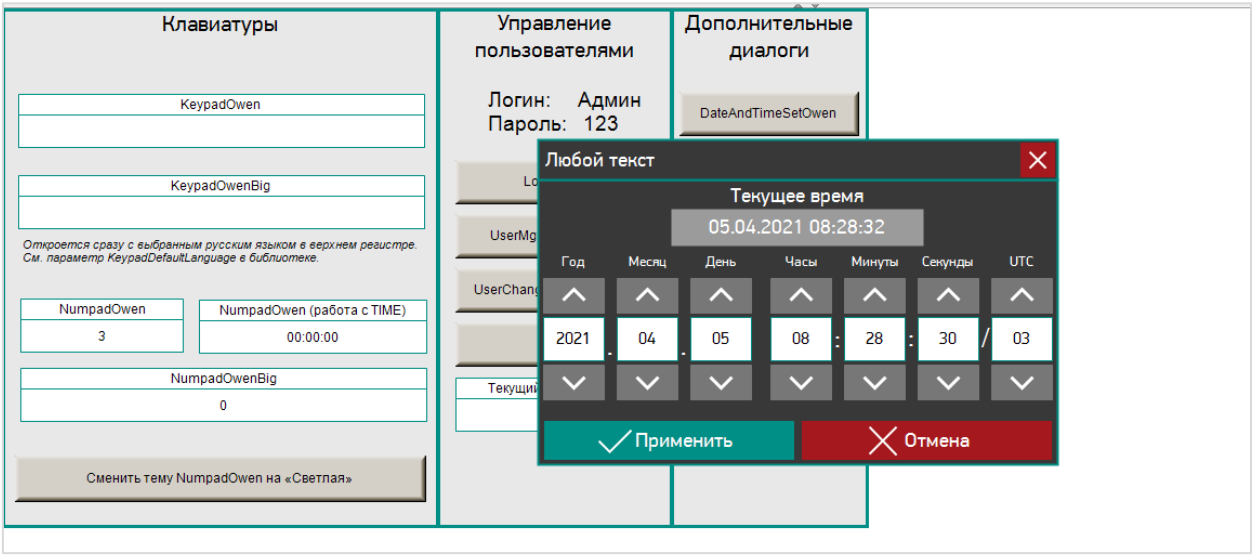


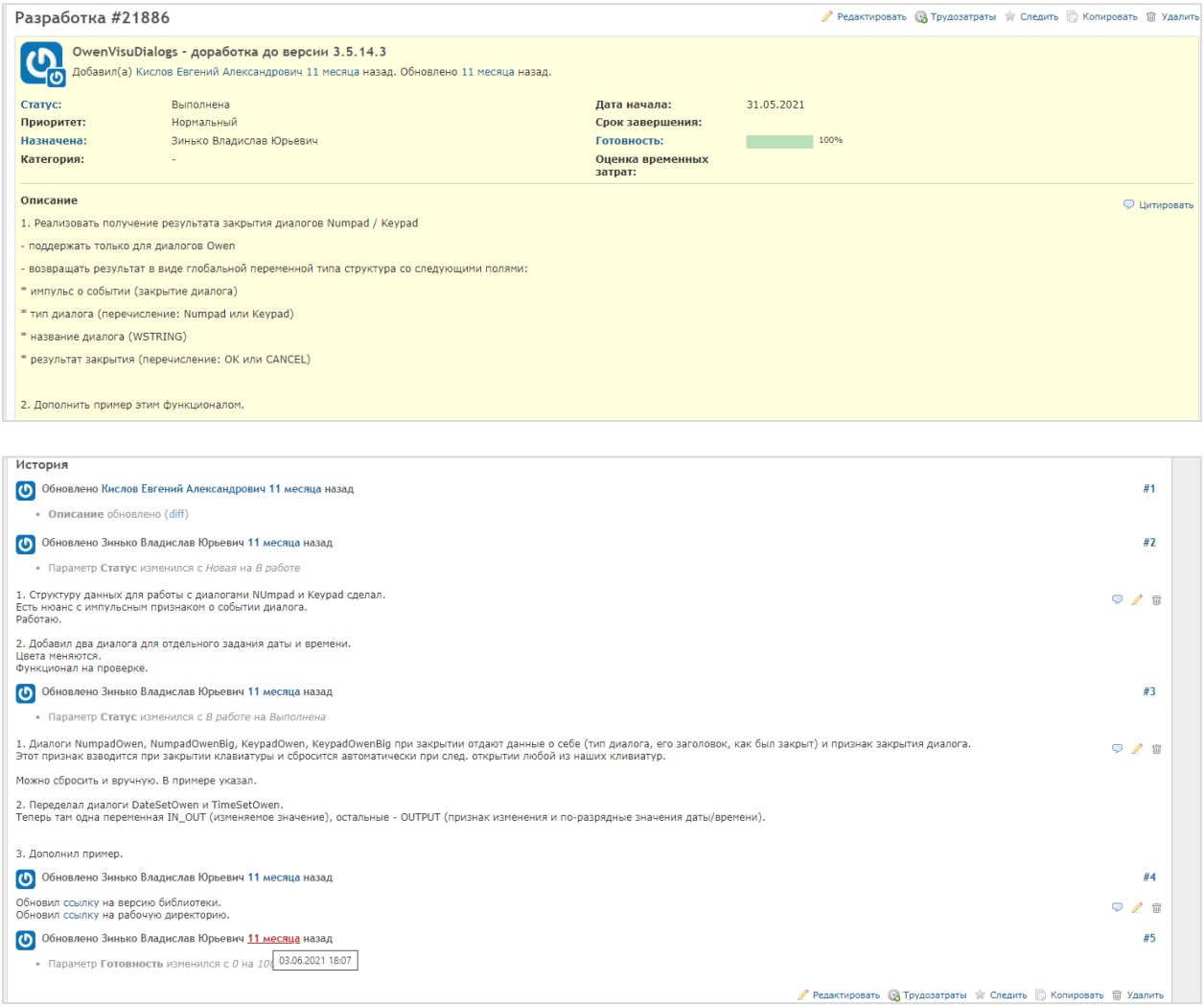
Рис. 38 – Внешний вид диалога DateTimeSetOwen

28 мая 2021 пользователь форума ОВЕН Votrj1 озвучивает вопрос, который, в принципе, к тому моменту я уже слышу не в первый раз: как в CODESYS определить, что диалог ввода (Numpad или Keypad) был закрыт с определенным результатом? (*например, если Numpad был закрыт по нажатию на кнопку ОК, что означает ввод нового значения, то нужно в коде программы выполнить какие-то операции*)

В тот же день мы начинаем обсуждать с Владом, как реализовать это в нашей библиотеке. Параллельно находится способ делать это стандартными средствами CODESYS – но этот способ, мягко говоря, является не очень удобным (требуется создать ФБ, реализующий библиотечный интерфейс, переопределить его методы, обеспечить его инициализацию и разобраться с тем, в каком именно виде он возвращает информацию о закрытии диалога). К этому моменту мы уже решаем с Владом добавить в глобальные переменные **OwenVisuDialogs** структуру с информацией о последнем закрытом диалоге (она включает в себя флаг закрытия, название диалога (заголовок), тип диалога (Numpad или Keypad) и результат – то есть кнопку, по которой диалог был закрыт (OK или CANCEL)).

31 мая я создаю задачу в Redmine, а уже 2 июня Влад заканчивает по ней работу. В процессе внезапно выясняется, что в библиотеку добавились еще два диалога для ввода переменных типа DATE и TIME (в сущности – это специальным образом обрезанные версии диалога ввода системного времени из версии **3.5.14.2**). Кажется, мы получили обратную связь от пользователей, что такие диалоги были бы им полезны (например, для ввода дат и интервалов времени работы оборудования при формировании суточного расписания). Кроме того, удобного способа для ввода в визуализации значений типа DATE в CODESYS вообще не было.

2 июня я выкладываю версию библиотеки 3.5.14.3 на форум и сайт.



Разработка #21886

Редактировать | Трудозатраты | Следить | Копировать | Удалить

OwenVisuDialogs - доработка до версии 3.5.14.3

Добавил(а) Кислов Евгений Александрович 11 месяца назад. Обновлено 11 месяца назад.

Статус:	Выполнена	**Дата начала:**	31.05.2021
Приоритет:	Нормальный	**Срок завершения:**	
Назначена:	Зинько Владислав Юрьевич	**Готовность:**	100%
Категория:	-	**Оценка временных затрат:**	

Описание — Цитировать

1. Реализовать получение результата закрытия диалогов Numpad / Keypad

- поддержать только для диалогов Owen

- возвращать результат в виде глобальной переменной типа структура со следующими полями:

* импульс о событии (закрытие диалога)

* тип диалога (перечисление: Numpad или Keypad)

* название диалога (WSTRING)

* результат закрытия (перечисление: OK или CANCEL)

2. Дополнить пример этим функционалом.

История

Обновлено Кислов Евгений Александрович 11 месяца назад #1

- **Описание** обновлено (diff)

Обновлено Зинько Владислав Юрьевич 11 месяца назад #2

- Параметр **Статус** изменился с *Новая* на *В работе*

1. Структуру данных для работы с диалогами NUmpad и Keypad сделал.
Есть нюанс с импульсным признаком о событии диалога.
Работаю.

2. Добавил два диалога для отдельного задания даты и времени.
Цвета меняются.
Функционал на проверке.

Обновлено Зинько Владислав Юрьевич 11 месяца назад #3

- Параметр **Статус** изменился с *В работе* на *Выполнена*

1. Диалоги NumpadOwen, NumpadOwenBig, KeypadOwen, KeypadOwenBig при закрытии отдают данные о себе (тип диалога, его заголовок, как был закрыт) и признак закрытия диалога.
Этот признак взводится при закрытии клавиатуры и сбросится автоматически при след. открытии любой из наших кливиатур.

Можно сбросить и вручную. В примере указал.

2. Переделал диалоги DateSetOwen и TimeSetOwen.
Теперь там одна переменная IN_OUT (изменяемое значение), остальные - OUTPUT (признак изменения и по-разрядные значения даты/времени).

3. Дополнил пример.

Обновлено Зинько Владислав Юрьевич 11 месяца назад #4

Обновил ссылку на версию библиотеки.
Обновил ссылку на рабочую директорию.

Обновлено Зинько Владислав Юрьевич 11 месяца назад #5

- Параметр **Готовность** изменился с *0* на *10(* 03.06.2021 18:07

Редактировать | Трудозатраты | Следить | Копировать | Удалить

Рис. 38 – Задача в Redmine по релизу версии 3.5.14.3 библиотеки OwenVisuDialogs

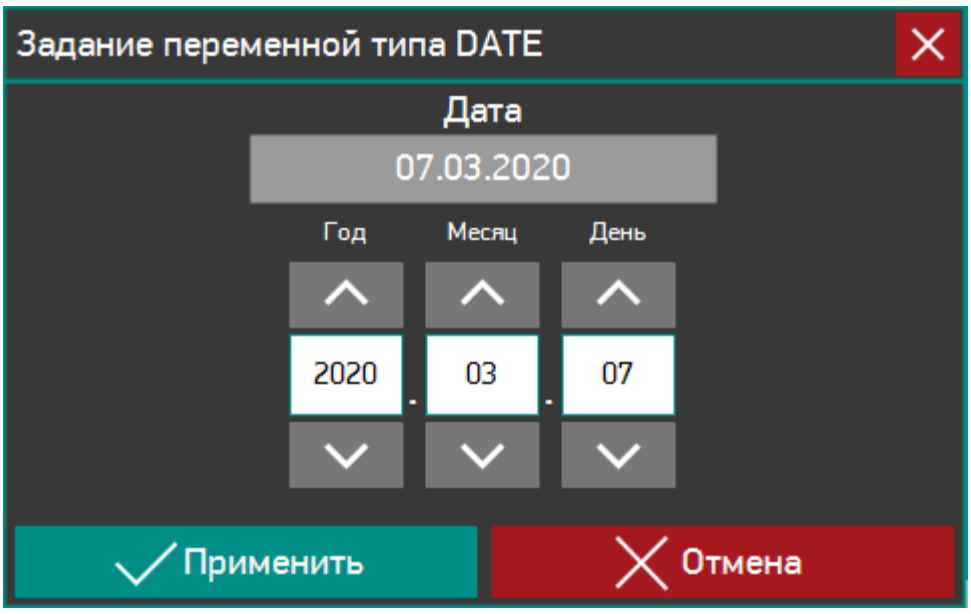


Рис. 39 – Внешний вид диалога DateSetOwen

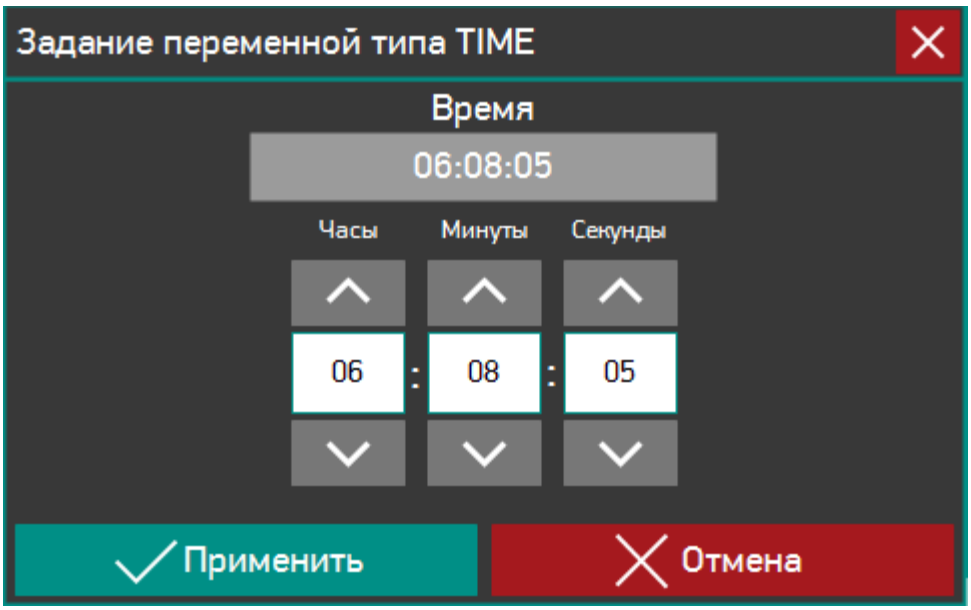


Рис. 40 – Внешний вид диалога TimeSetOwen

Итак, за три месяца после выпуска версии **3.5.14.1** мы два раза обновляли библиотеку, добавляя в нее новый функционал – вполне неплохой результат.

Летом 2021 года мы выпускаем прошивки для наших контроллеров с системой исполнения CODESYS V3.5 SP16 Patch 3. В сентябре один из клиентов замечает, что в этой версии CODESYS не удается использовать диалоги управления пользователями из **OwenVisuDialogs** – возникают ошибки компиляции:

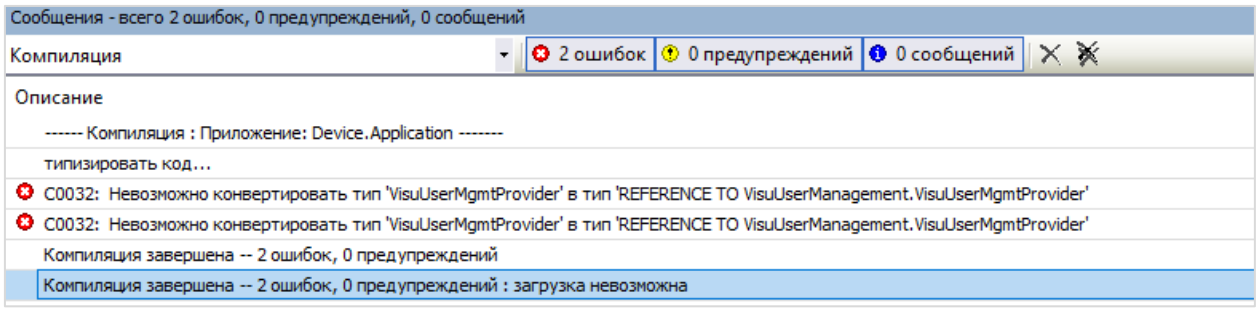


Рис. 41 – Ошибки компиляции библиотеки OwenVisuDialogs 3.5.14.x в CODESYS V3.5 SP16 Patch 3

Проблема обнаружилась довольно быстро – в новой версии CODESYS произошли какие-то изменения в системной библиотеке **VisuUserMgmt**, блоки которой как раз используются для реализации диалогов управления пользователями, поэтому пришлось пересобрать **OwenVisuDialogs** в CODESYS V3.5 SP16 Patch 3, использовав самые свежие версии вложенных библиотек. Новая сборка получила версию **3.5.16.3** (*чтобы показать, что по функционалу она соответствует версии библиотеки **3.5.14.3**; в тот момент были планы далее обновлять версии библиотеки синхронно для обоих SP – через какое-то время выпустить **3.5.14.4** и **3.5.16.4**, и т.д.*). В конце сентября 2021 эта версия выкладывается на нашем форуме и сайте.

В январе 2022 Влад завершил работу над очередной (и последней из запланированных на тот момент) версией библиотеки OwenVendorProtocols (*эта библиотека включает в себя блоки опроса тепло- и электросчетчиков по нестандартным протоколам обмена*). В процессе обсуждений следующих проектов мы решаем для начала выпустить очередную версию **OwenVisuDialogs**, для которой уже придуманы две значительные фичи:

- поддержка мультиязычности;
- добавление диалогов для работы с узлами таргет-файла (Drives, Network и т.д.).

Предложение о мультиязычности диалогов озвучил Влад – в прошлом он сталкивался с ситуациями, когда некоторые клиенты настаивали на такой опции для своих проектов. Идея насчет таргет-диалогов возникла естественным путем – в версии **3.5.14.2** мы добавили диалог **DateTimeSetOwen**, который по нашей задумке должен был использоваться совместно с каналами узла таргет-файла **OwenRTC**. Но ведь у нас есть и другие узлы, для которых пригодился бы готовый графический интерфейс – например, **Drives** (просмотр информации о памяти контроллера и подключенных накопителей), **Network** (отображение и изменение сетевых настроек контроллеров СПК) и т.д. – почему бы не добавить диалоги и для них?

Камнем преткновения становится вопрос передачи нужных переменных в диалог. Шаблоны проектов для наших контроллеров включают в себя структуры для узлов таргет-файлов; в списке глобальных переменных объявлены экземпляры этих структур, а их поля привязаны к соответствующим каналам этих узлов. Самый удобный вариант, который приходит в голову – передавать при вызове блока экземпляр этой структуры, но тут мы столкнулись с проблемой – если одна и та же структура (с одним и тем же названием и набором переменных) объявлена в проекте и библиотеке, то CODESYS всё равно считает их разными объектами и запрещает присваивать их экземпляры между собой. В итоге сначала мы решили, что пользователь будет передавать в диалог указатель на структуру, объявленную в шаблоне проекта – это не потребует от него каких-то изменений в проекте; правда, если он как-то отредактирует структуру из шаблона – то имеет серьезные шансы словить access violation. Этот вариант нам не нравился, но ничего лучше мы в тот момент не придумали.

Работа над новой версией библиотеки по ряду причин сильно затягивается, и в какой-то момент становится понятно, что не за горами релиз прошивок с системой исполнения CODESYS V3.5 SP17 Patch 3 – а значит, разумно готовить новую версию библиотеки сразу под неё. У нас появляется карт-бланш для любых изменений, и я переделываю для новой версии шаблоны проектов – оформляю все структуры, используемые в шаблонах проектов для узлов таргет-файлов, в виде отдельной библиотеки **OwenTypes**. В эту же библиотеку переносятся перечисления из других библиотек (например, у нас была библиотека **OwenWatchdog**, которая включала в себя только одно перечисление для узла **Watchdog**; после того, как это перечисление было перенесено в **OwenTypes**, то необходимость в библиотеке пропала и у нас стало на одну сущность меньше). Библиотека **OwenTypes** добавляется в обновленный пакет таргет-файлов.

После этого Влад добавляет **OwenTypes** в **OwenVisuDialogs** и необходимость в указателях исчезает – потому что и в шаблоне, и в диалогах нашей библиотеки отныне используются одни и те же структуры.

В «стандартной» библиотеке **VisuDialogs** в новой версии CODESYS тоже произошли кое-какие изменения:

- появилась возможность перетаскивать диалоги по экрану. Для этого нужно навести курсором на их заголовок, зажать левую кнопку мыши и перетащить диалог на новое место;
- в диалоге просмотра содержимого каталогов (**FileOpenSave**) объекты каталога теперь сортируются (раньше они отображались в произвольном порядке).

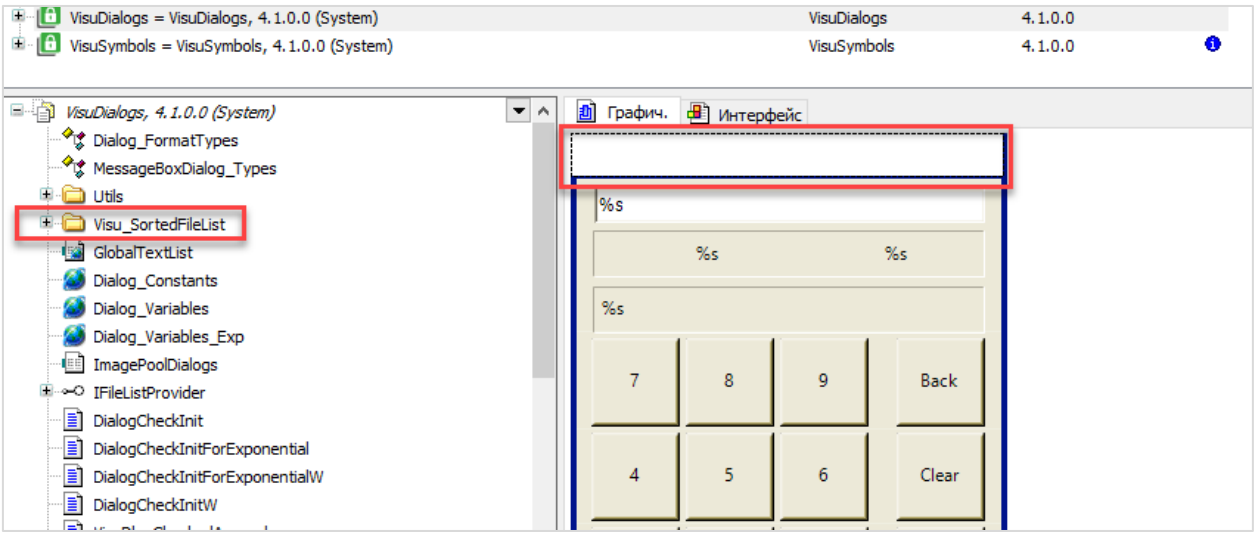


Рис. 42 – Новый функционал в библиотеке VisuDialogs 4.1.0.0

Второй момент уже давно раздражал Влада, и он предлагал решить его своими силами. Поскольку я считал, что это недостаток на стороне CODESYS, который рано или поздно будет решен, то отговаривал его от этого – и, видимо, правильно делал.

При попытке перенести эти улучшения в **OwenVisuDialogs** мы сталкиваемся с проблемой – версия **4.1.0.0** библиотеки **VisuDialogs** недоступна в исходниках. На форуме CODESYS один из разработчиков подтверждает, что исходники больше не устанавливаются вместе с обновлениями среды – вместо этого в будущем их можно загрузить из CODESYS Forge со страницы примеров (*в итоге исходники новых версий библиотеки выложили в июне 2022 года вместе с релизом версии плагина визуализации **4.2.0.0** – к этому моменту мы уже выпустили обновление **OwenVisuDialogs***).

В результате «перетаскиваемость» диалогов удается очень легко реализовать своими силами. Как оказалось, в CODESYS V3.5 SP17 Patch 3 это стало штатным функционалом – достаточно разместить в диалоге элемент **Невидимый ввод** с установленной галочкой **Used As Pointed Area**. Сортировку файлов, соответственно, мы в очередной раз откладываем до лучших времен.

В начале 2022 года Влад периодически возвращается к работе над библиотекой. В релиз она уходит 24 мая – одновременно с выпуском обновленной прошивки с CODESYS V3.5 SP17 Patch 3 для нашего контроллера ПЛК200. Версия релизной библиотеки – **3.5.17.2**, потому что **3.5.17.1** мы

активно использовали во время отладки, создав с десяток ее различных вариаций. Поэтому чтобы самим на запутаться – увеличили релизную версию на единичку.

Кроме добавления мультиязычности и таргет-диалогов в новой версии библиотеки нами было исправлено два бага CODESYS.

Первый из них был связан с тем, что в версии CODESYS V3.5 SP17 Patch 3 перестал корректно работать диалог логина – если в менеджере визуализации не установлена галочка **Support client animations and overlay native elements**, то закрыть этот диалог просто не получалось. Проблема была в том, что в настройках поля ввода пароля установлена галочка **Поле пароля** (*в этом случае введенное значение по мере ввода скрывается «звездочками»*), которая в новой версии CODESYS стала обрабатываться некорректно. Подробнее можно почитать в этой и этой темах на форуме CODESYS. Казалось бы, проблема решается элементарно – достаточно установить галочку **Support client animations and overlay native elements**. Но дело в том, что для наших панельных контроллеров СПК такой вариант не подходит – поскольку они не поддерживают технологию overlay для таргет-визуализации, то эта галочка вообще не отображается в менеджере визуализации. Поэтому мы создали диалог **LoginOwen2**, в котором сняли галочку **Поле пароля** с элемента ввода пароля – с закрытием такого диалога никаких проблем не возникает. Надо, кстати, отметить, что снятие галочки не означает, что пароль теперь отображается в явном виде – он всё также скрыт звездочками после ввода. Дело в том, что галочка влияет на появление звездочки сразу после ввода очередного символа пароля. Без нее символы пароля видны, но после завершения ввода он заменяется звездочками сразу весь целиком.

Второй баг связан с диалогом просмотра содержимого каталогов. Этот диалог позволяет удалять каталоги – но только в том случае, если в нем нет вложенных каталогов и файлов (*это ограничение используемой для удаления функции **SysDirDelete** из библиотеки **SysDir***). Влад давно хотел это исправить – и в новой версии библиотеки для «нашего» диалога заменил вызов этой функции на ФБ **DirRemove** из библиотеки **CAA File**, который поддерживает рекурсивное удаление каталога. Кстати, пока я писал эти строки, то еще раз посмотрел на эти библиотеки и обнаружил, что в библиотеке **SysDir** есть функция **SysDirDelete2** для рекурсивного удаления – почему-то мы не догадались ее поискать и пошли чуть более сложным путем. Что ж, бывает.

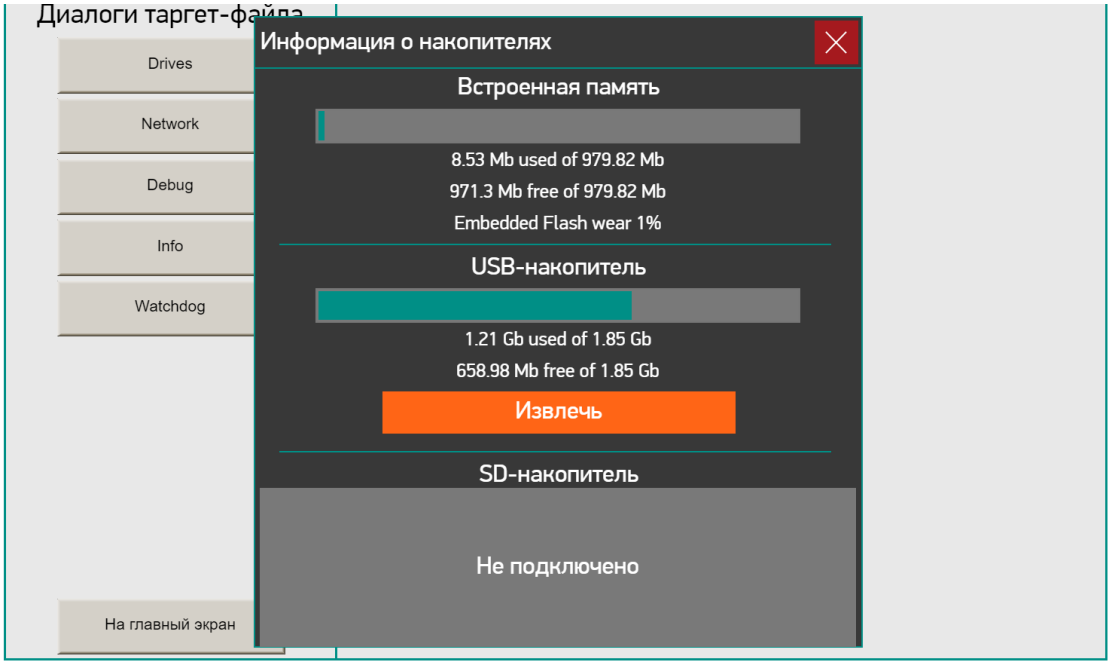


Рис. 43 – Внешний вид диалога DrivesOwen

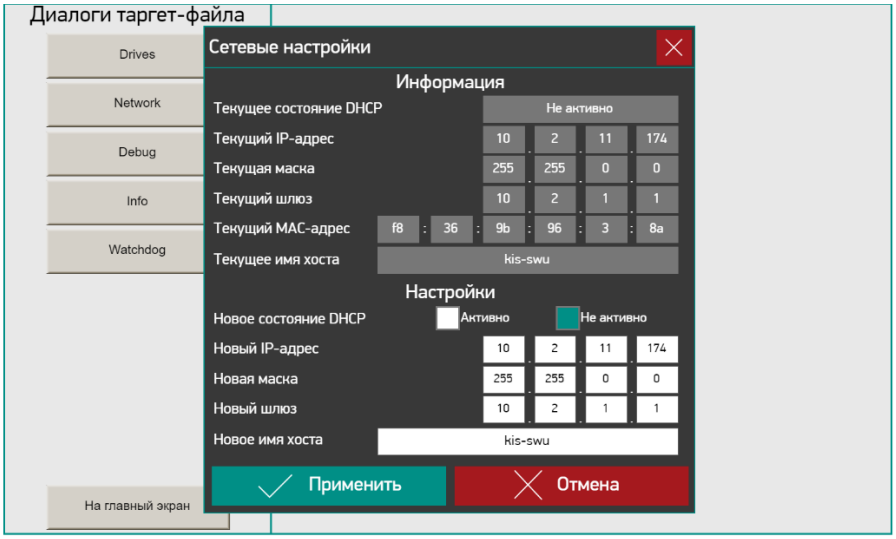


Рис. 44 – Внешний вид диалога NetworkOwen

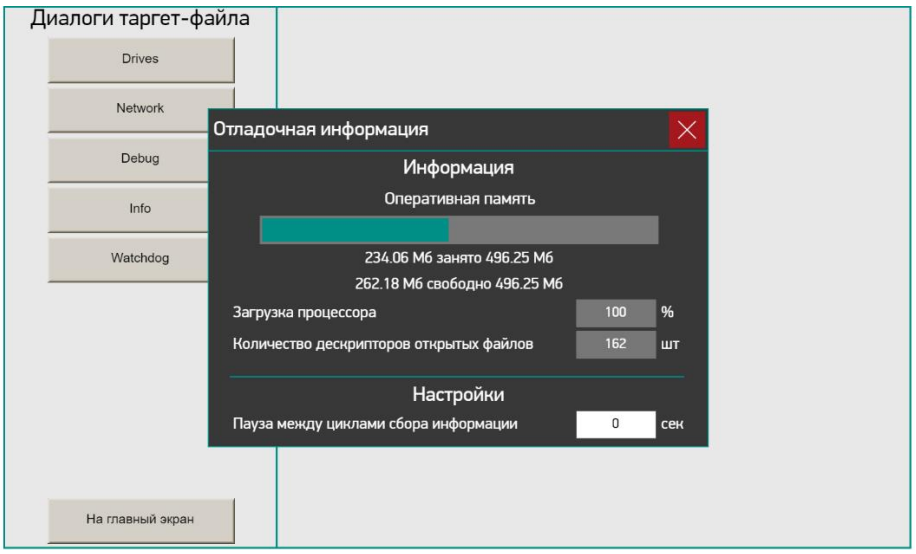


Рис. 45 – Внешний вид диалога DebugOwen

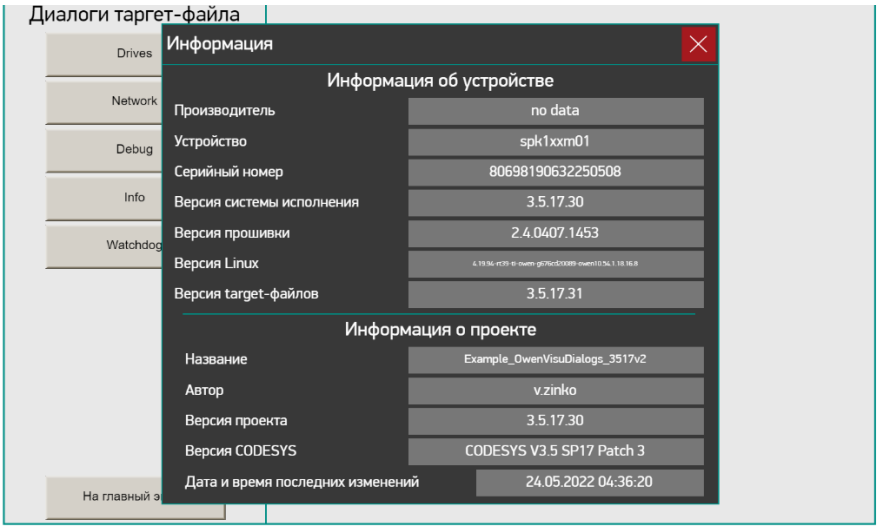


Рис. 46 – Внешний вид диалога InfoOwen

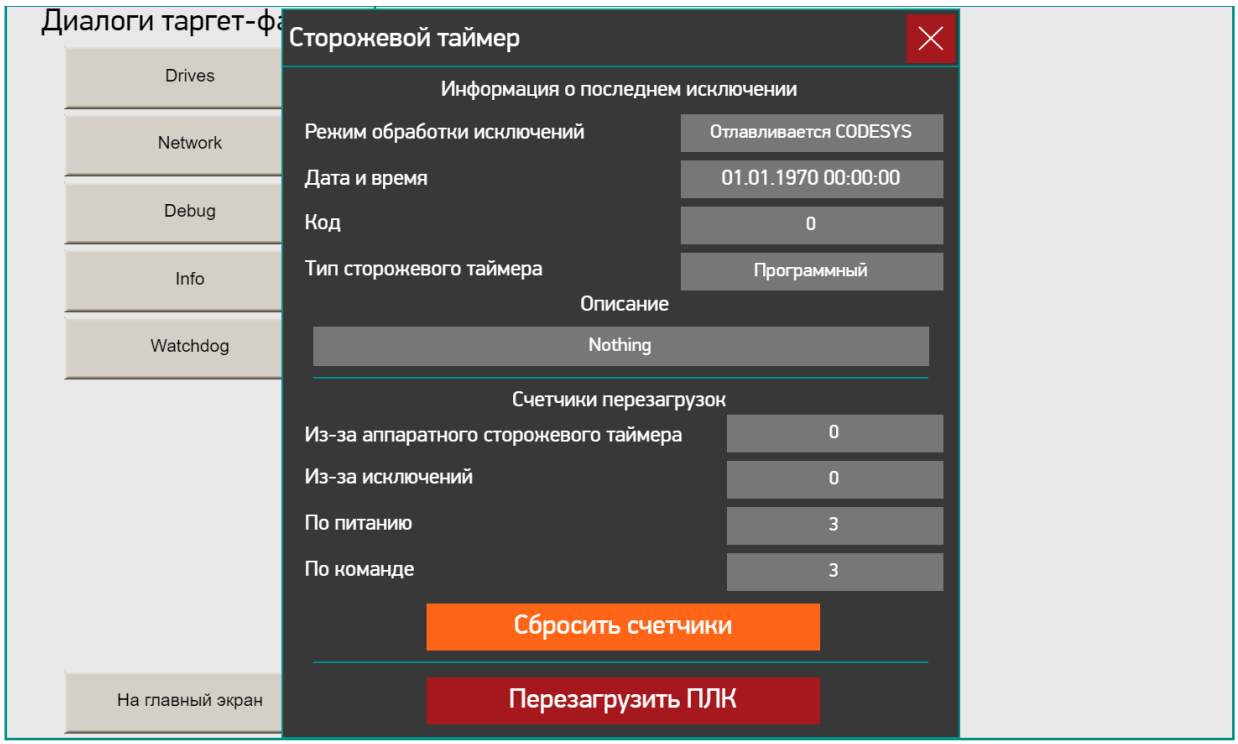


Рис. 47 – Внешний вид диалога WatchdogOwen

На текущий момент (июнь 2022) мне больше нечего рассказать об истории разработки библиотеки **OwenVisuDialogs**. Наверняка в будущем мы еще обновим ее – например, наконец-то перетащим механизм сортировки файлов в диалоге просмотра каталогов, раз уж свежую версию **VisuDialogs** наконец-то выложили в исходниках. А теперь пора перейти к той теме, ради которой эта статья изначально задумывалась – к обзору нескольких интересных фич, использованных в процессе реализации функционала библиотеки.

Фичи и нюансы OwenVisuDialogs

1. Циклический вызов функций в диалогах

Достаточно часто при работе с диалогами возникают ситуации, в которых было бы удобно иметь возможность циклически выполнять определенный фрагмент кода, пока этот диалог открыт – например, «на лету» проверять корректность введенного пользователем значения. Иногда было бы достаточно даже не циклического, а событийного выполнения – например, в момент открытия диалога проинициализировать его переменные на основании входных значений, переданных пользователем при открытии диалога. Но, к сожалению, внутри диалога невозможно явным образом детектировать момент его открытия.

Влад однажды (еще до начала работ над **OwenVisuDialogs**) рассказал мне о способе, который позволяет реализовать такое поведение. Для этого к какому-либо параметру элемента визуализации, размещенного в диалоге, привязывается вызов функции. Эта функция будет циклически (в контексте задачи **VISU_TASK**) вызываться всё время, пока этот диалог открыт.

Влад обычно привязывал вызовы функций к фоновому прямоугольнику диалога – к параметрам **Невидимый** и **Отключение ввода** (соответственно, функция, привязанная к параметру невидимости, всегда возвращала **FALSE**, чтобы прямоугольник оставался видимым).

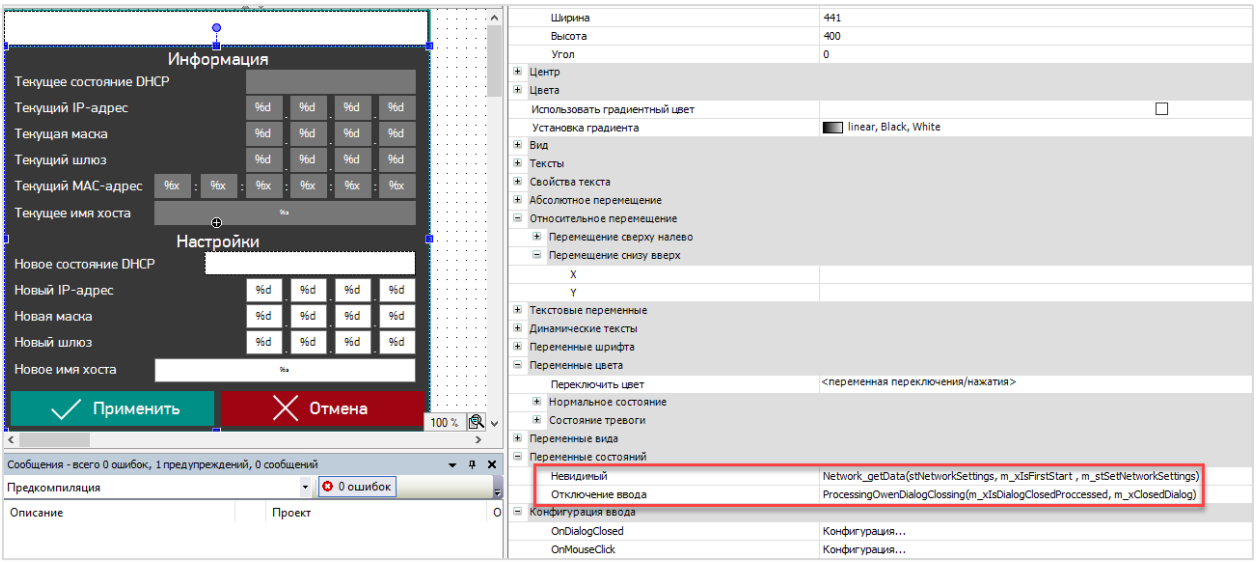


Рис. 48 – Привязка функций к параметрам элементов диалога

Такие функции могут взаимодействовать с интерфейсными переменными диалога (входами и входами-выходами) и глобальными переменными библиотеки. Например, функция **ProcessingOwenDialogClosing**, которую видно на скриншоте ниже, сбрасывает в **FALSE** глобальную переменную **DialogsSettings.g_stClosedDialogInfo.xIsDialogClosed** (признак закрытия любого кастомизированного диалога библиотеки) в момент открытия нового диалога. Функция имеет два входа-выхода – на первый из них передается флаг открытия диалога, а на второй – флаг закрытия. Флаг закрытия формируется в коде, который привязан к кнопке **Отмена** (закрыть диалог можно только с ее помощью). Флаг открытия обрабатывается самой функцией – ему присваивается

значение **TRUE** в момент открытия диалога и значение **FALSE** перед его закрытием. Поскольку функция не имеет собственной памяти, то для хранения данных используются переменные диалога, которые передаются на входы-выходы функции.

2. Циклический вызов ФБ в диалогах

В диалоге ввода **DateTimeSetOwen** менять значения разрядов времени можно не только с помощью клавиатуры (она появляется при нажатии на значение разряда), но и с помощью кнопок «вверх» и «вниз». При этом чем дольше зажата кнопка – тем на большую величину изменяется значение разряда за каждую секунду удержания. Соответственно, чтобы определить время удержания – нужно циклически вызывать функциональный блок таймера. В п. 1 был описан способ циклического вызова функции в диалоге. Но если внутри функции вызывать ФБ, то возникает вопрос – где этот ФБ должен быть объявлен? Явно не в локальных переменных функции – ведь функция не имеет памяти. Поэтому экземпляры таймеров объявлены в локальных переменных самого диалога – под них память выделяется. Для того, чтобы функция могла вызывать эти таймеры – массив экземпляров таймеров передается в функцию через ее вход-выход.

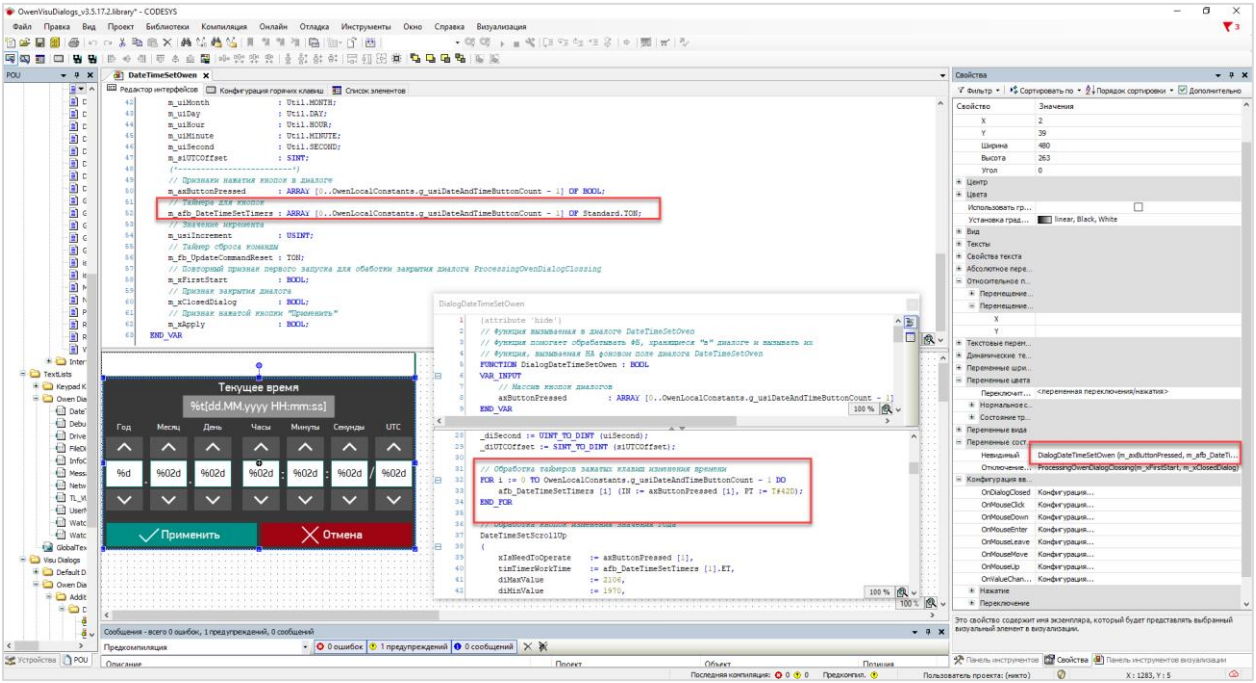


Рис. 48 – Код функции DialogDateTimeSetOwen

3. Диалог FileDirChoiceOwen

Диалог **FileDirChoiceOwen** является кастомизированной версией диалога **FileOpenSave**. По сравнению со «стандартным» диалогом **FileOpenSave** Влад внес множество изменений.

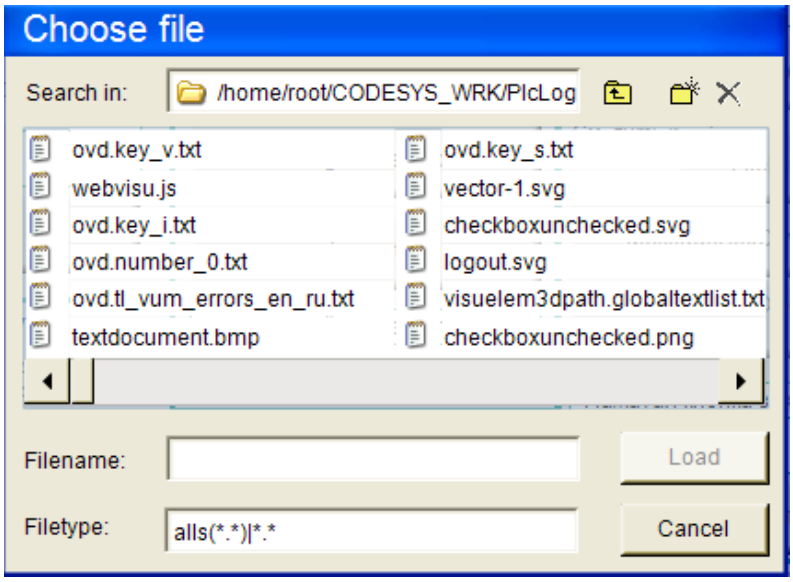


Рис. 50 – Внешний вид диалога FileOpenSave

1. Заголовок диалога в виде переменной типа **WSTRING** и переменная для текста кнопки «выбора»

В стандартном диалоге заголовок представлял собой переменную типа **STRING** – соответственно, отобразить к нем текст на кириллице не получалось. Поэтому в нашем диалоге тип переменной изменился на **WSTRING**; эта переменная передается в метод **Initialize** ФБ **Visu_OwenFbFileListProvider**. Этот метод необходимо вызвать перед открытием диалога. Кроме того, Влад добавил в этот метод возможность задать текст для кнопки «выбора» (на рис. 50 этот текст – «Load»). Мы рассуждали так: в зависимости от контекста задачи диалог может использоваться для разных целей – например, открытия файла, выбора файла, его выгрузки на ПК и т.д. Поэтому разумно дать пользователю возможность самому задавать текст этой кнопки («Открыть», «Выбрать») и т.п.

2. Механизм прокрутки

В стандартном диалоге механизм прокрутки довольно своеобразный – в методе **Initialize** пользователь задает значение переменной **iRowCount**. По перемещению ползунка (или по нажатию стрелок рядом с ним) происходил «сдвиг» на соответствующее число отображаемых элементов. Проще объяснить на примере: предположим, что **iRowCount** = **4**. Тогда по нажатию (*на самом деле, из-за особенностей реализации нажать придется несколько раз*) на стрелку «вправо» на рис. 50 диалог будет выглядеть вот так:

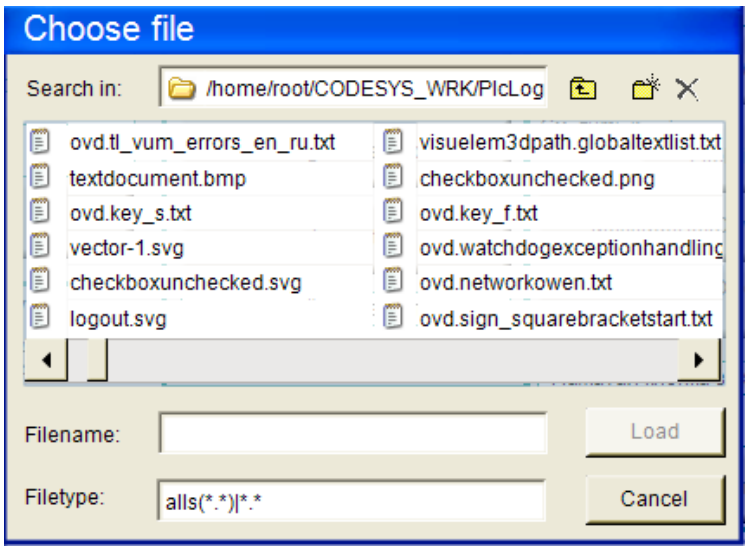


Рис. 51 – Внешний вид диалога FileOpenSave после одной прокрутки вправо

То есть произойдет сдвиг на 4 элемента – первые 4 элемента исчезнут за левой границей экрана, а их место займут элементы, ранее имевшие номера 5-8 и т.д.

Влад изменил это поведение – переменная **iRowCount** удалена из метода, а прокрутка таблицы теперь происходит путем смещения целых столбцов (для этого были внесены соответствующие изменения в код ФБ **Visu_OwenFbFileListProvider**).

3. Иконки форматов

В стандартном диалоге для всех форматов файлов отображается одна и та же иконка (см. скриншоты выше).

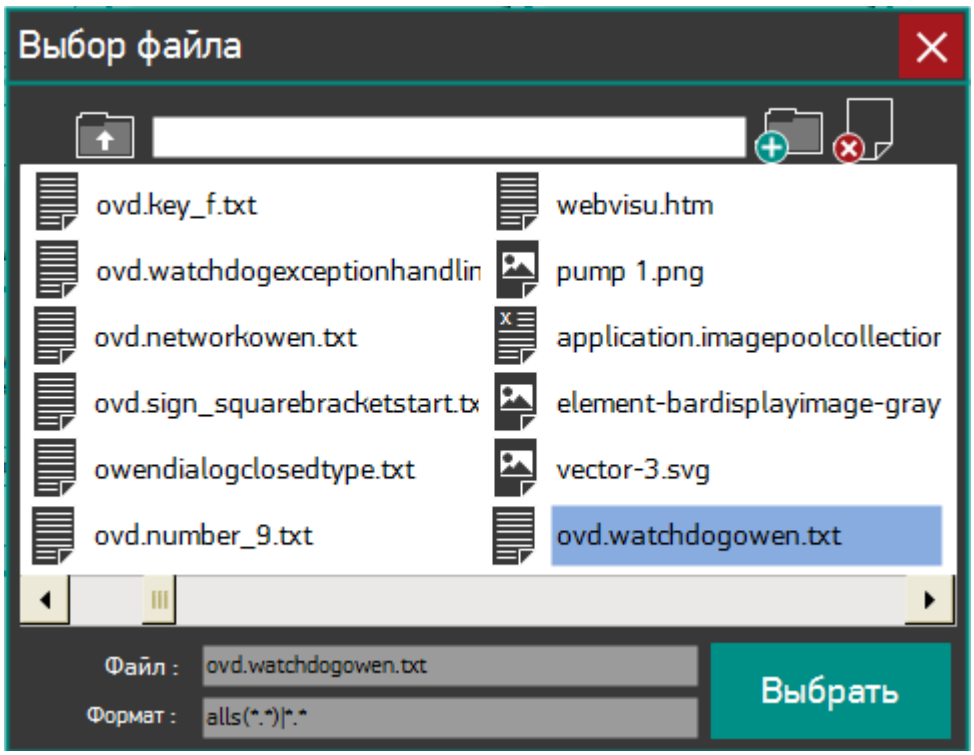


Рис. 52 – Иконки форматов диалога FileDirChoiceOwen

В нашем диалоге Влад сделал три типа иконок:

- для табличных файлов (.csv, .xls, .xltx, .ods, .ots);
- для графических файлов (.jpg, .jpeg, .tif, .tiff, .png, .gif, .bmp, .dib, .svg);
- для всех остальных файлов.

Все эти иконки видны на рис. 52.

При этом иконки не являются изображениями – они отрисованы вручную с помощью стандартных элементов CODESYS (Прямоугольник, Полигон и т.д.). Это позволяет менять их цвета прямо в коде программы (см. п. 4). Кстати, иконки форматов – один из единичных случаев использования группировки элементов в библиотеке. В основном использование группировки в ней намеренно избегалось, потому что во многих версиях CODESYS это сопровождается различными графическим артефактами (хотя, кажется, в свежих версиях подобных проблем стало меньше).

Для определения формата Влад создал функции **isImageDoucment** и **isTabularDoucment**. Кстати, я только что заметил опечатку в их названиях – поправим в следующей версии библиотеки.

Напоследок отмечу, что в баг-трекере CODESYS есть тикет по этому вопросу, но он до сих пор на закрыт.

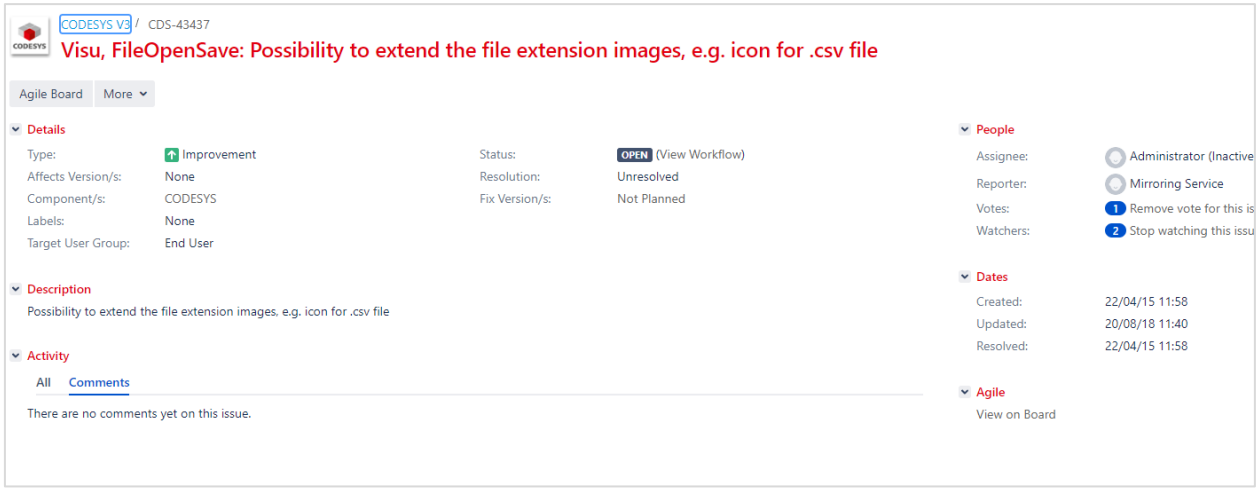


Рис. 53 – Тикет по поддержке иконок форматов файлов в диалоге FileOpenSave

4. Дополнительные окна при создании и удалении директории/файлов

В стандартном диалоге создание директорий и удаление директорий/файлов происходит сразу после нажатия на соответствующую кнопку (на рис. 51 их видно в верхнем правом углу). Влад добавил для этих случаев отдельные информационные окна (они реализованы в виде наложенных поверх диалога элементов, которые становятся видимыми после нажатия на соответствующую кнопку).

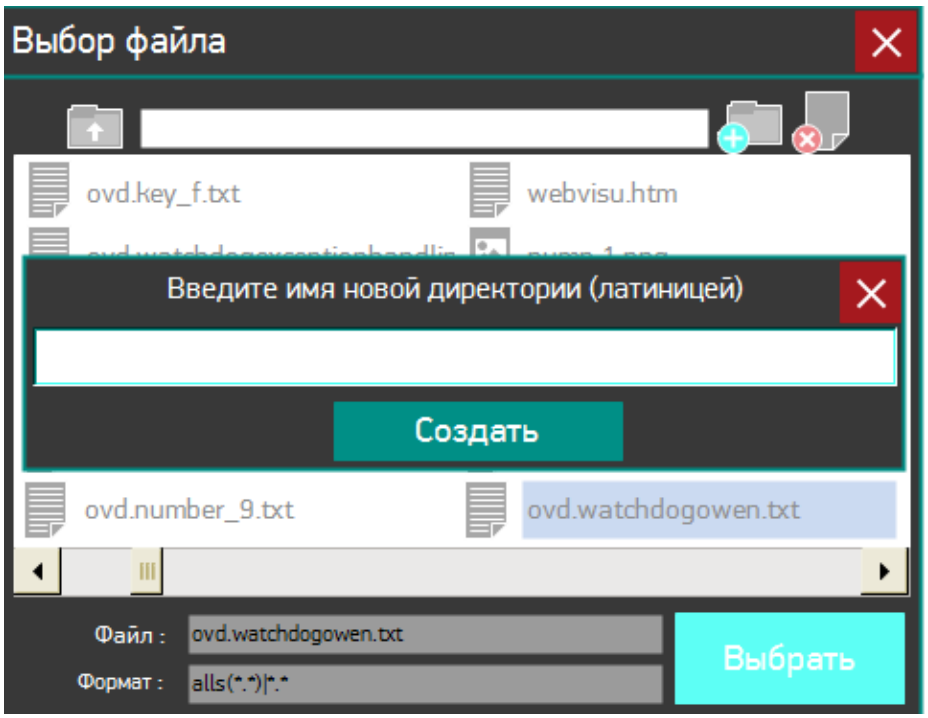


Рис. 54 – Всплывающее окно создания новой директории

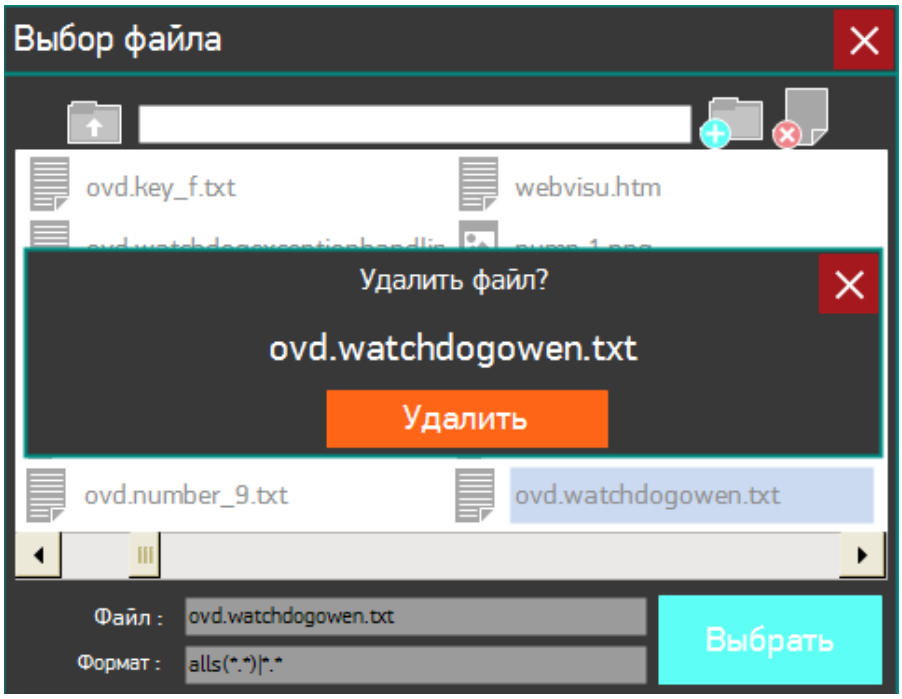


Рис. 55 – Всплывающее окно удаления файла/директории

Кстати, насчет удаления директорий – как уже упоминал ранее, в стандартном диалоге их можно было удалить только в том случае, если они были пустыми. Влад добавил возможность удаления директорий, внутри которых есть вложенные каталоги и файлы. Подробнее об этом см. на странице 33.

4. Изменение цветов диалогов

Еще при планировании работ по самой первой версии библиотеки мы часто обсуждали вопрос кастомизации внешнего вида диалогов с точки зрения пользователей. Мы выбрали определенный визуальный стиль, но что, если он не понравится каким-то клиентам? Понятно, что они могут открыть библиотеку (ведь мы распространяем ее в исходниках) и отредактировать ее под себя – но это всё же не самый удобный способ. Кроме того, мы периодически выпускаем обновления библиотеки – и пользователям будет непросто синхронизировать их со своей версией. В общем, нужен был более простой способ.

В итоге мы пришли к мысли сделать цвета диалогов полностью настраиваемыми через глобальные переменные библиотеки. Это позволяет менять цвета прямо из кода программы и, более того, в процессе ее выполнения – например реализовать переключение между «темной» и «светлой» темой в зависимости от времени суток.

В списке глобальных переменных **DialogSettings** объявлены экземпляры структур цветов каждого диалога. Цвет представляет собой переменную типа **DWORD** (формат ARGB). В библиотеке есть перечисление **COLORS**, содержащее набор готовых цветов, и функции **RGB_TO_DWORD_COLOR** / **DWORD_COLOR_TO_RGB**, позволяющие «собирать» и «разбирать» цвета из отдельных оттенков.

DialogsSettings | FileDirChoiceOwenColors

```
{attribute 'qualified_only'}
VAR_GLOBAL
    // Структура настроек диалога KeypadOwen
    g_stKeypadOwenColors            : KeypadOwenColors;
    // Структура настроек диалога KeypadOwenBig
    g_stKeypadOwenBigColors         : KeypadOwenColors;
    // Структура настроек диалога NumpadOwen
    g_stNumpadOwenColors            : NumpadOwenColors;
    // Структура настроек диалога NumpadOwenBig
    g_stNumpadOwenBigColors         : NumpadOwenColors;
    // Структура настроек диалога loginOwen
    g_stLoginOwenColors             : LoginOwenColors;
    // Структура настроек диалога UserChangePswdOwen
    g_stUserChangePswdOwenColors    : UserChangePasswordOwenColors;
    // Структура настроек диалога UserMgmtOwen
    g_stUserMgmtOwenColors          : UserMgmtOwenColors;
    // Структура настроек диалога FileDirChoiseOwen
    g_stFileDirChoiseColors         : FileDirChoiceOwenColors;
    // Структура настроек диалога MessageBoxOwen
    g_stMessageBoxOwenColors        : MessageBoxOwenColors;
    // Структура настроек диалога DateTimeSetOwen
    g_stDateTimeSetColors           : DateTimeSetOwenColors;
    // Структура настроек диалога DateSetOwen
    g_stDateSetColors               : DateTimeSetOwenColors;
    // Структура настроек диалога TimeSetOwen
    g_stTimeSetColors               : DateTimeSetOwenColors;
```

Рис. 56 – Список глобальных переменных DialogSettings

DialogsSettings | FileDirChoiceOwenColors

```
// Структура настроек цветов диалога FileDirChoice
TYPE FileDirChoiceOwenColors :
STRUCT
    // Цвет фона основного окна
    dwDialogBackground          : DWORD                     := COLORS.OwenDlgBackground;
    // Цвет рамки диалога
    dwDialogFrame               : DWORD                     := COLORS.OwenLogoGreen;
    // Цвет фона заголовка
    dwTitleBackground           : DWORD                     := COLORS.OwenDlgBackground;
    // Цвет рамки заголовка
    dwTitleFrame                : DWORD                     := COLORS.OwenLogoGreen;
    // Цвет текста заголовка
    dwTitleText                 : DWORD                     := COLORS.White;
    // Цвет рамки поля наименования директории
    dwDirFrame                  : DWORD                     := COLORS.White;
    // Цвет фона поля наименования директории
    dwDirBackground             : DWORD                     := COLORS.White;
    // Цвет текста поля наименования директории
    dwDirText                   : DWORD                     := COLORS.White;
    // Цвет слова "Файл"
    dwFileText                  : DWORD                     := COLORS.White;
```

Рис. 57 – Структура цветов диалога FireDirChoiceOwen

RGB_TO_DWORD_COLOR

```
// Функция возвращает общую переменную типа DWORD для управления цветом элемента на визуализации
FUNCTION RGB_TO_DWORD_COLOR : DWORD
VAR_INPUT
    // Прозрачность цвета
    byAlpha         : BYTE;
    // Глубина красного оттенка
    byRed           : BYTE;
    // Глубина зеленого оттенка
    byGreen         : BYTE;
    // Глубина синего оттенка
    byBlue          : BYTE;
END_VAR
```

```
RGB_TO_DWORD_COLOR := BYTE_TO_DWORD (byBlue) +
                      SHL (BYTE_TO_DWORD (byGreen), 8) +
                      SHL (BYTE_TO_DWORD (byRed), 16) +
                      SHL (BYTE_TO_DWORD (byAlpha), 24);
```

Рис. 58 – Код функции RGB_TO_DWORD_COLOR

5. Адаптация размера шрифта под длину строки

В некоторых диалогах отображаются строк, длина которых заранее неизвестна. Например, сетевое имя контроллера (hostname) может включать в себя 10 символов, а может – 60 (максимальное ограничение – 80). При этом размер прямоугольника, в котором этот текст отображается – фиксированный. То есть если закладываться на отображение строки максимально возможной длины – то размер шрифта пришлось бы выбрать достаточно маленький, что не очень здорово при отображении коротких строк. Поэтому Влад наложил друг на друга несколько прямоугольников с разным размером шрифта и одной и той же текстовой переменной, и в зависимости от длины строки переключал их видимость. Он рассматривал вариант изменения размера шрифта (его тоже можно менять динамически, и тогда наложения прямоугольников можно было бы избежать), но в итоге отказался от него, чтобы не добавлять в библиотеку еще одну функцию.

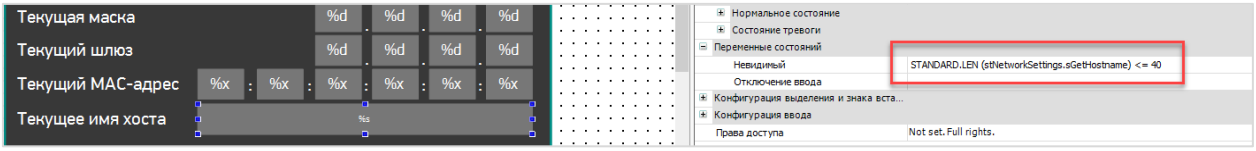


Рис. 59 – Переключение видимости элементов в зависимости от длины строки

Честно говоря, не уверен, что в этом была серьезная необходимость – пользователь может просто поставить в менеджере визуализации галочку для автоматической адаптации размера шрифта под размер элемента в зависимости от длины отображаемого значения. С другой стороны, он может и не знать о ее существовании – так что текущая реализация имеет определенный смысл.

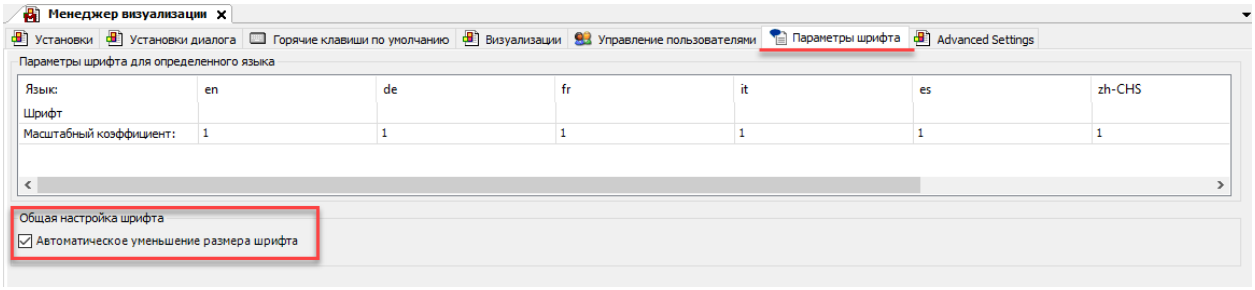


Рис. 60 – Настройка автоматического изменения размера шрифта в менеджере визуализации

6. Валидация значений в диалоге Numpad

При настройке вызова диалога **Numpad** можно указать минимальное и максимальное допустимое значение.

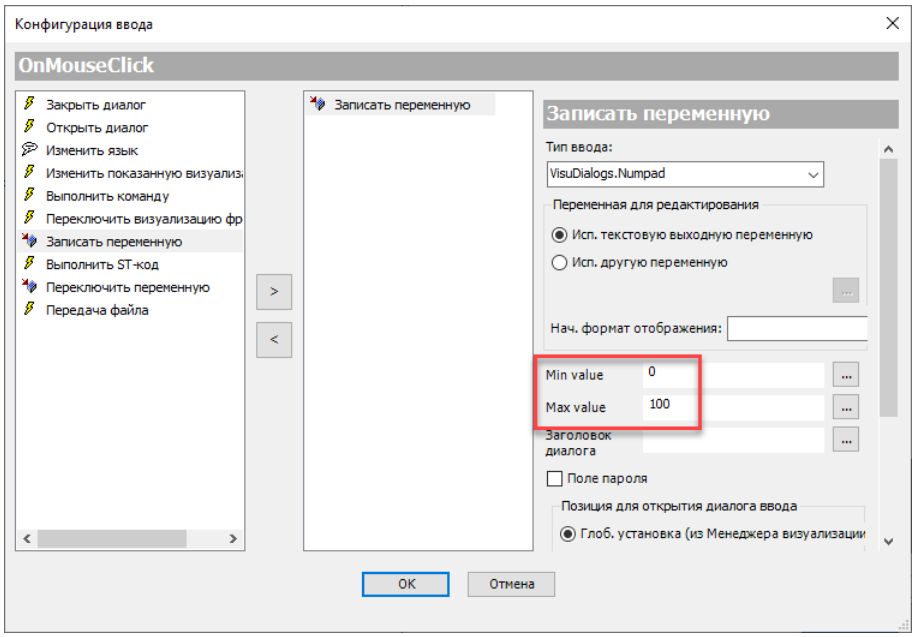


Рис. 61 – Выбор диапазона допустимых значения для диалога Numpad

Если введенное пользователем значение выходит за пределы этого диапазона – то после нажатия на кнопку ОК диалог не будет закрыт, а надпись **Min value** или **Max value** будет подсвечена красным (в зависимости от того, меньше ли введенное значение нижней границы или больше верхней).

Рис. 62 – Индикация выхода введенного значения за допустимый диапазон

Переключение цвета реализовано следующим образом:

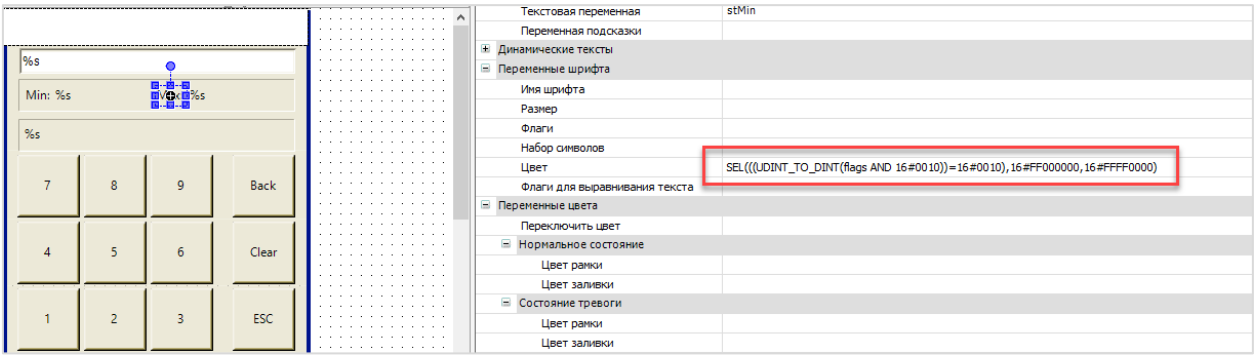


Рис. 63 – Код переключения цвета

Переменная **flags**, по всей видимости, изменяется в неявно создаваемом для каждого диалога ввода (см. рис. 64) функциональном блоке (пользователю этот блок недоступен). Этот же блок запрещает закрытие диалога при определенном значении переменной **flags** (хотя к кнопке **OK** и привязано действие **Закрыть диалог** без каких-либо условий, но фактически пока введенное значение выходит за допустимый диапазон – диалог нельзя будет закрыть нажатием на эту кнопку).

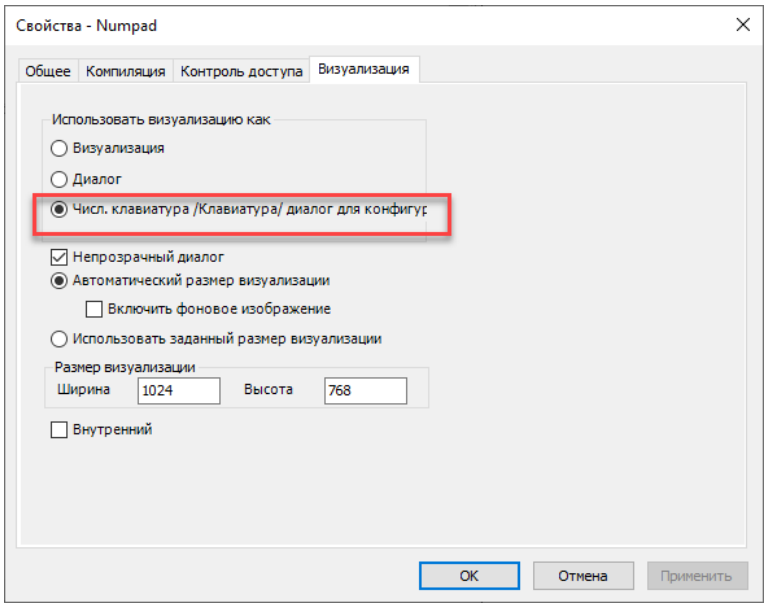


Рис. 64 – Выбор типа диалога в его настройках

В процессе разработки библиотеки **OwenVisuDialogs** выяснился интересный момент – если скопировать диалог **Numpad**, то функционал индикации выхода значения за диапазон перестает работать. Этот баг был исправлен только в версии V3.5 SP17.

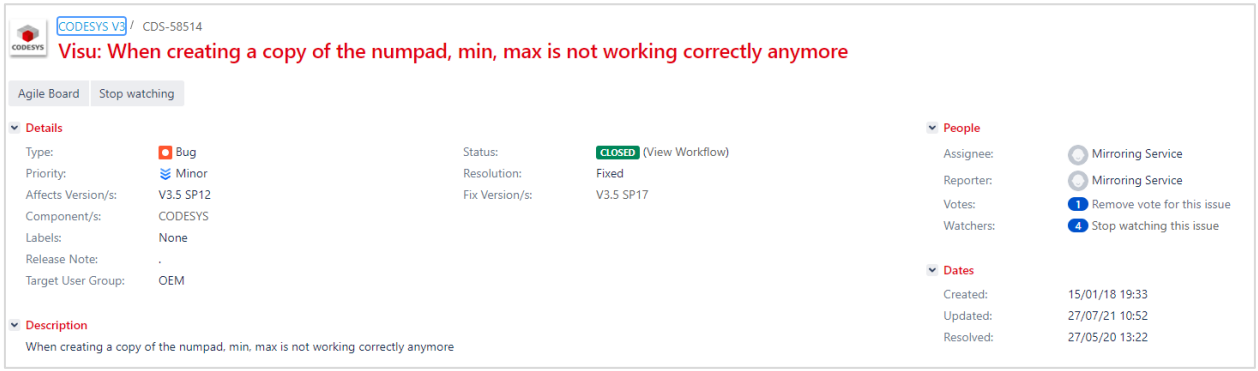


Рис. 65 – Тикет о некорректной работе индикации выхода значения за диапазон при копировании диалога

Поэтому Влад реализовал свой механизм валидации, который работал «на лету» – то есть индикация о выходе за диапазон срабатывает еще до нажатия кнопки ОК.

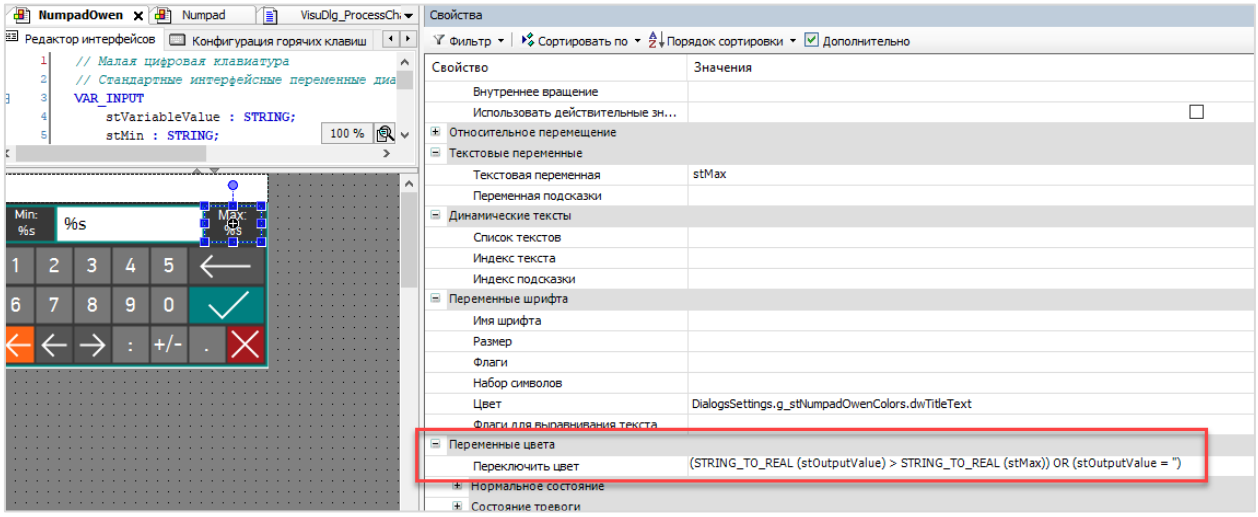


Рис. 66 – Код переключения цвета для диалогов ОВЕН

В ST-коде кнопки **ОК**, кстати, пришлось добавить дополнительную проверку на принадлежность введенного значения диапазону – чтобы в случае выхода за его границы не возвращать в глобальные переменные информацию о закрытии диалога (поскольку в данном случае он не будет закрыт за счет обработки переменной **flags** в неявно созданном ФБ – см. информацию на предыдущей странице).

7. Нюанс диалогов управления пользователями

Если вы решите отредактировать диалоги управления пользователями, то для вас может оказаться важной одна особенность – в этих диалогах не стоит добавлять интерфейсные переменные (например, локальные). Дело в том, что в случае добавления таких переменных – этот диалог нельзя будет указать в менеджере визуализации на вкладке **Установки диалога** в качестве диалога управления пользователями. Поэтому все дополнительные переменные, используемые в этих диалогах (например, переменная переключения цвета кнопки при ее нажатии с рис. 67) вынесены в глобальные переменные.

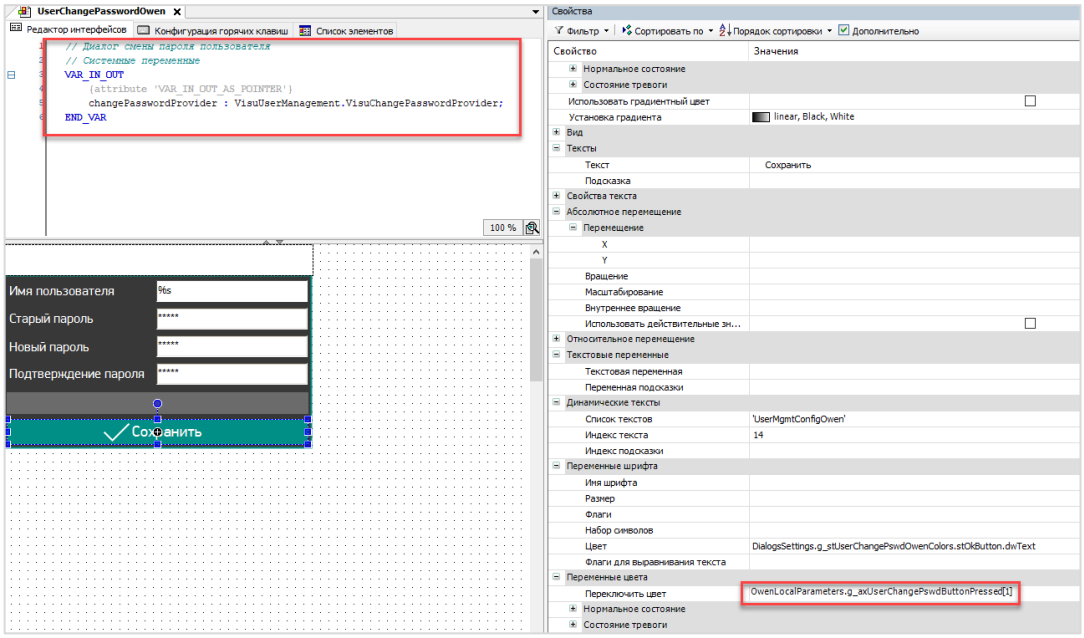


Рис. 67 – Использование глобальных переменных в диалогах управления пользователями

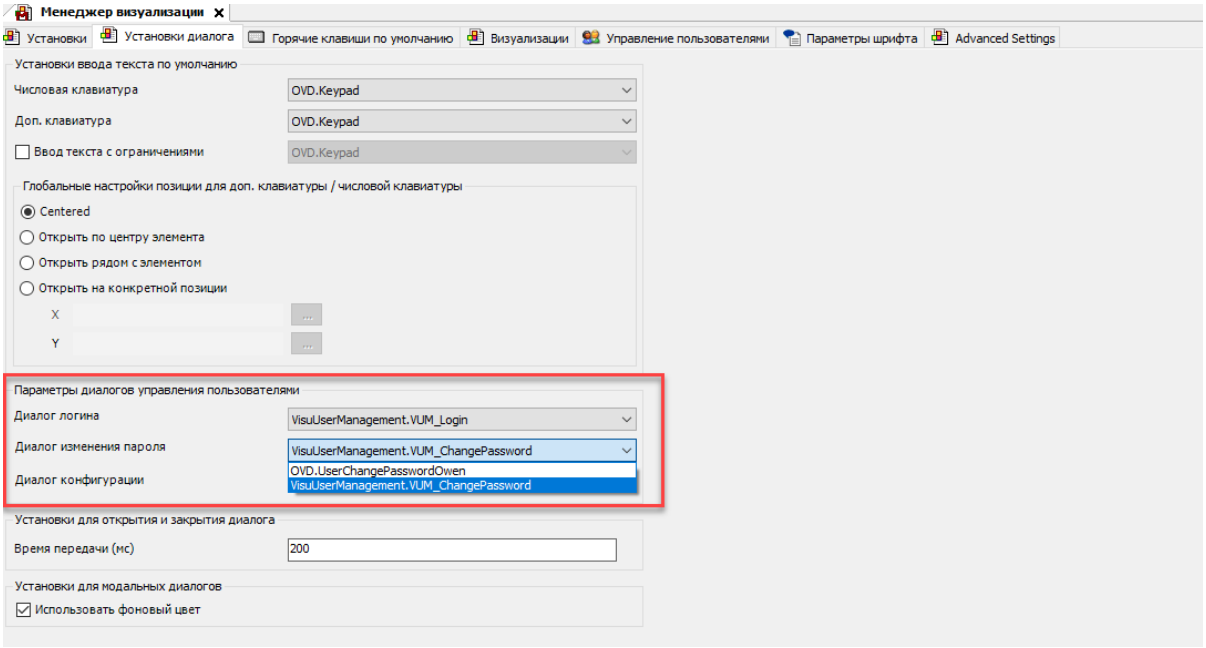


Рис. 68 – Выбор диалога библиотеки в качестве диалога управления пользователями

8. Получение информации о закрытии диалогов

Одной из уже ранее упоминавшихся особенностей библиотеки является предоставление информации о закрытии диалогов. Это реализовано через ST-код, привязанный к кнопкам закрытия диалогов. Например, в диалоге **NumpadOwen** к кнопке **OK** привязан вот такой код:

Рис. 69 – Код кнопки закрытия диалога NumpadOwen с копированием информации о закрытии диалога в глобальные переменные

Принцип проверки принадлежности значения допустимому диапазону уже был описан в п. 6. В первой строке кода происходит сброс флага нажатия кнопки для изменения ее цвета. Код обработки закрытия диалога на рисунке выделен красным. В нем выполняется наполнение структуры **g_stClosedDialogInfo** списка глобальных переменных **DialogsSettings**. Эта структура содержит информацию о последнем закрытом диалоге:

- **xIsDialogClosed** (тип BOOL) – признак того, что один из диалогов библиотеки был закрыт;
- **eDialogClosingResult** (тип OwenDialogClosed_RESULT) – результат закрытия диалога (название кнопки, по нажатию на которую он был закрыт);
- **eDialogType** (тип OwenDialog_TYPE) – тип закрытого диалога;
- **wsTitle** (тип WSTRING) – заголовок закрытого диалога (чтобы, например, можно было отличить закрытие Numpad'а, привязанного к разным элементам визуализации).

Флаг **xIsDialogClosed** автоматически сбрасывается при открытии нового библиотечного диалога. Это реализовано с помощью функции **ProcessingOwenDialogClosing**, которая циклически вызывается в каждом кастомизированном диалоге (см. п. 1).

При закрытии диалога переменной **xClosedDialog** присваивается значение **TRUE** (см. рис. 69), что приводит к сбросу флага **xFirstStart** (см. рис. 70). В результате при следующем открытии диалога произойдет сброс переменной **xIsDialogClosed**. Кроме того, пользователь может сбросить эту переменную из кода своей программы.

NumpadOwen | ProcessingOwenDialogClossing

```
{attribute 'hide'}
// Функция обработки закрытия ОВЕН-диалогов
// Через эту функцию сбрасывается признак первого запуска диалога и глобальная переменная, указывающая на
// Иными словами, через эту функцию мы сбрасываем DialogsSettings.g_stClosedDialogInfo.xIsDialogClosed
// Вызывается во всех ОВЕН-диалогах на фоновом прямоугольнике в поле "Deactivate input"
FUNCTION ProcessingOwenDialogClossing : BOOL
VAR_IN_OUT
    // Признак того, что диалог уже отбработал один цикл. По умолчанию должен быть FALSE
    xFirstStart             : BOOL;
    // Признак того, что конкретный диалог был закрыт. Устанавливается по кнопке закрытия диалога
    xClosedDialog           : BOOL;
END_VAR
```

```
ProcessingOwenDialogClossing := FALSE;

IF NOT xFirstStart THEN
    xFirstStart := TRUE;
    DialogsSettings.g_stClosedDialogInfo.xIsDialogClosed := FALSE;
END_IF

IF xClosedDialog THEN
    xFirstStart := FALSE;
    xClosedDialog := FALSE;
END_IF
```

Рис. 70 – Код функции ProcessingOwenDialogClosing