

Работа со списками текстов в CODESYS V3



CODESYS



19.12.2025
версия 2.1

Оглавление

Оглавление.....	2
Введение	4
1. Основная информация о списках текстов	5
2. Добавление списка текстов в проект. Настройки списка текстов	6
3. Использование динамических текстов в визуализации	12
4. Создание мультязычной визуализации	13
5. Перечисления с поддержкой текстов. Элемент визуализации «Выпадающий список»	16
6. Библиотека <code>StrDynamicText</code>	19
6.1. Основная информация.....	19
6.2. Функция <code>DynamicTextChangeLanguage</code>	20
6.3. Функция <code>DynamicTextGetLanguage</code>	20
6.4. Функция <code>DynamicTextGetDefaultText</code>	21
6.5. Функция <code>DynamicTextGetDefaultTextW</code>	22
6.6. Функция <code>DynamicTextGetText</code>	23
6.7. Функция <code>DynamicTextGetTextW</code>	25
6.8. Функция <code>DynamicTextLoadDefaultTexts</code>	27
6.9. Функция <code>DynamicTextRegisterFile</code>	27
6.10. Функция <code>DynamicTextRegisterPath</code>	28
6.11. Функция <code>DynamicTextReloadTexts</code>	28
6.12. Функция <code>DynamicTextUnRegisterFile</code>	29
6.13. Функция <code>DynamicTextIterateIndices</code>	29
6.14. Комментарий к пунктам 6.15-6.24	30
6.15. Функция <code>DynamicTextGetLanguageText</code>	31
6.16. Функция <code>DynamicTextGetLanguageTextW</code>	32
6.17. Функция <code>DynamicTextLoadLanguage</code>	33
6.18. Функция <code>DynamicTextUnloadLanguage</code>	34
6.19. Функция <code>DynamicTextUnreferencedLanguage</code>	34
6.20. Функция <code>DynamicTextReleaseLanguages</code>	35
6.21. Функция <code>DynamicTextIterateLanguages</code>	35
6.22. Функция <code>DynamicTextSetLock</code>	38

6.22. Функция DynamicTextTestAndSetLock	39
6.24. Функция DynamicTextReleaseLock	40
7. Библиотека TextListUtils	41
7.1. Основная информация.....	41
7.2. Перечисление ReturnValues.....	41
7.3. Функция GetText	42
7.4. Функция GetTextW	44
7.5. Функция GetTextListInfo	46
7.6. Комментарий к пунктам 7.7-7.8	48
7.7. Функция GetLanguageText.....	49
7.8. Функция GetLanguageTextW	51
7.9. Асинхронные ФБ и Non blocking API	52
Заключение	55

Введение

Данный документ описывает работу со списками текстов в среде **CODESYS V3.5**.

В [п. 1](#) приводится основная информация о списках текстов.

В [п. 2](#) описывается добавление списков текстов в проект и доступные для них настройки.

В [п. 3](#) описывается использование динамических текстов в визуализации.

В [п. 4](#) описывается создание мультязычной визуализации и способы переключения языков.

В [п. 5](#) описывается работа с перечислениями с поддержкой списка текстов на примере настройки элемента визуализации **Выпадающий список** (Combobox – Integer).

В [п. 6](#) приводится описание библиотеки **CmpDynamicText**, которая используется для работы со списками текстов в коде программы.

В [п. 7](#) приводится описание библиотеки **TextListUtils**, которая представляет собой удобную и простую в использовании обертку над библиотекой **CmpDynamicText**.

Автор: Евгений Кислов

1. Основная информация о списках текстов

Список текстов – это компонент **CODESYS V3.5**, который позволяет организованно хранить текстовую информацию в памяти контроллера. Списки текстов могут использоваться для следующих целей:

- хранения статических текстов, отображаемых в визуализации (например, в элементах **Метка, Кнопка, Выпадающий список** и т.д.);
- хранения статических текстов, используемых в коде программы (например, для формирования строковых переменных с сообщениями об ошибках);
- реализации [динамических текстов](#) в визуализации (т.е. текстов, переключаемых в элементе визуализации по заданному условию);
- реализации [мультиязычной визуализации](#) (с переключаемым языком текстов).

Каждый список текстов представляет собой файл с расширением **.txt**. Название файла совпадает с названием списка текстов, приведенного к нижнему регистру. Для списков текстов, расположенных в библиотеках, в название файла включаются пространства имен библиотек.

Пример: в проекте присутствует библиотека **VisuElemsAlarm**, в которую вложена библиотека **AlarmManager** со списком текстов **TL_AlarmStatus**. Имя файла данного списка текстов будет выглядеть как **visuelemsalarm.alarmmanager.tl_alarmstatus.txt**.

Если в менеджере визуализации установлена галочка **Использовать строки Unicode** – то файлы имеют кодировку [UCS-2](#) (эта кодировка [близка к UTF-16](#)). Если галочка не установлена – то файлы имеют кодировку [ASCII](#).

В памяти ПЛК файлы списков текстов размещаются в рабочей директории CODESYS в папке **/PlcLogic/Visu**.

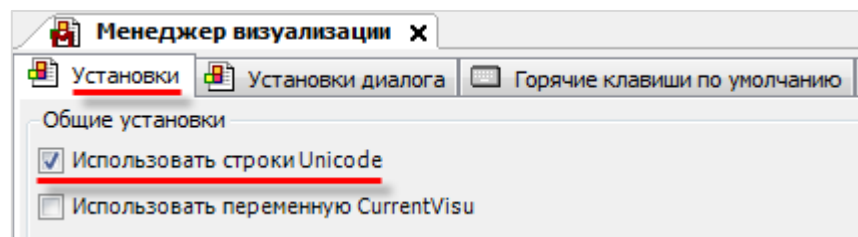


Рис. 1.1. Галочка **Использовать строки Unicode** в менеджере визуализации

2. Добавление списка текстов в проект. Настройки списка текстов

Для добавления списка текстов в проект CODESYS следует нажать ПКМ на узел **Application** и использовать команду **Добавление объекта – Список текстов**.

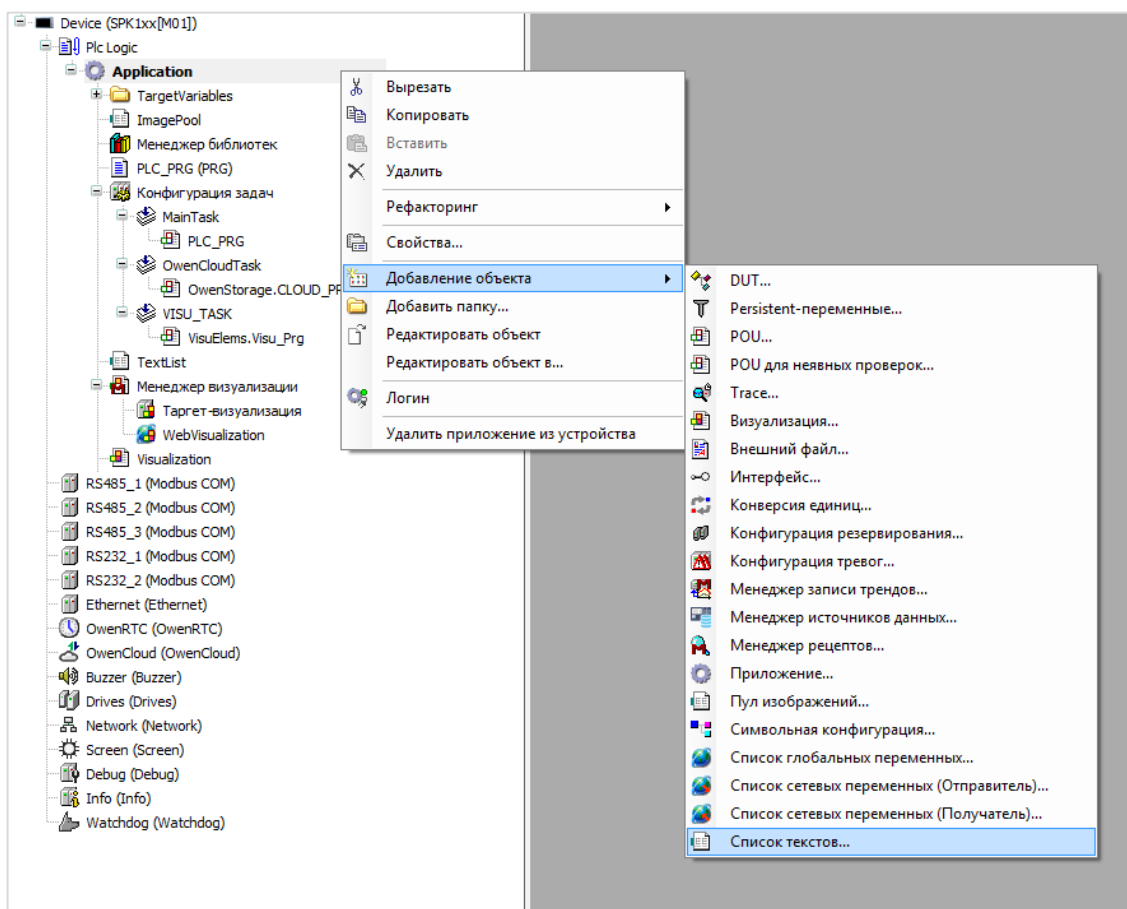
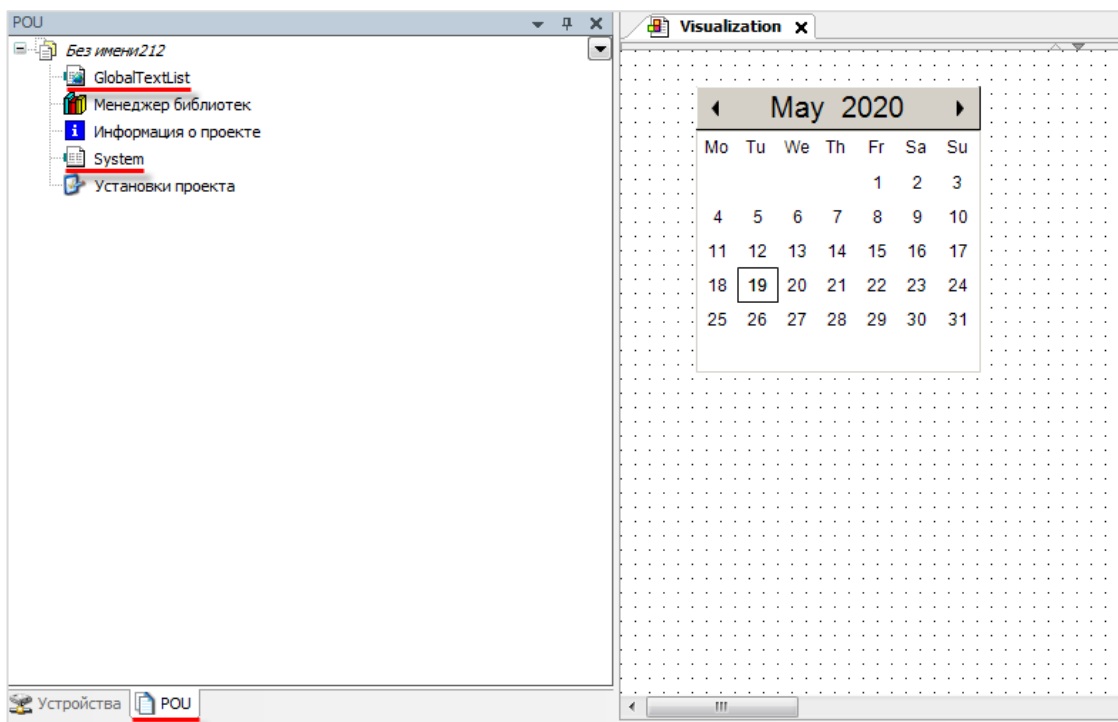


Рис. 2.1. Добавление списка текстов в проект CODESYS

Также в некоторых ситуациях списки текстов добавляются в проект автоматически:

- при добавлении группы тревог в компонент **Конфигурация тревог** автоматически создается одноименный список текстов, в котором хранятся сообщения этой группы. В этих текстах могут использоваться [специальные заполнители](#) (LATCH1 и т.д.), которые позволяют отображать в аварийном сообщении дополнительную информацию (например, значение триггерной переменной);
- на вкладке **POU** (см. рис. 2.2) по умолчанию присутствует список текстов **GlobalTextList**, в котором хранятся тексты, введенные в настройках элементов визуализации в параметрах **Тексты/Текст** и **Тексты/Подсказка**. Этот список не может редактироваться пользователем;
- при добавлении в проекте элементов визуализации **Элемент выбора даты** или **Элемент выбора даты и времени** на вкладке **POU** автоматически создается список текстов **System**, содержащий полные и краткие названия дней и месяцев.

Рис. 2.2. Списки текстов на вкладке **POU**

Пользователь может добавлять списки текстов не только в дереве проекта, но и на вкладке **POU** – в этом случае они будут доступны во всех устройствах проекта (если в проект добавлено несколько устройств).

Список текстов представляет собой таблицу, которая включает столбец идентификатора текста (**ID**) и один или несколько столбцов с текстовыми сообщениями. По умолчанию в списке текстов присутствует только один столбец с названием **По умолчанию**.

ID списка текстов представляет собой строку и может включать не только числа, но и символы (в т.ч. спецсимволы типа «.», «,», «:»).

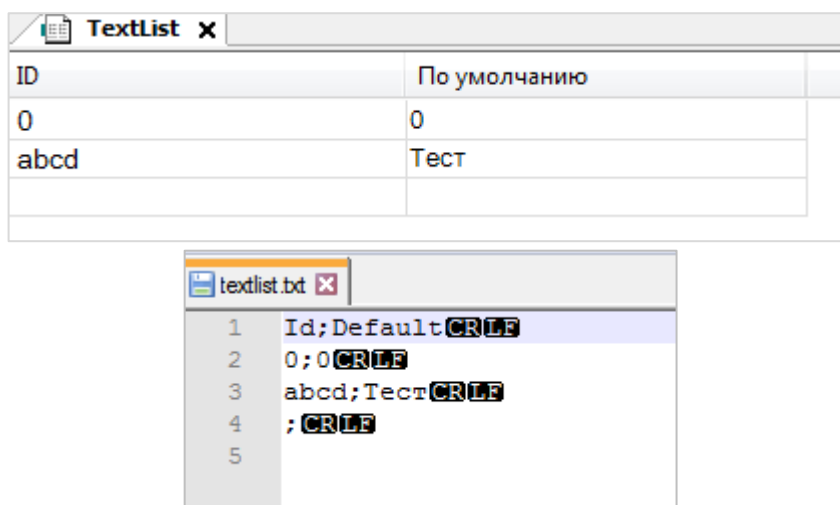


Рис. 2.3. Внешний вид списка текстов в проекте CODESYS и содержимое соответствующего текстового файла в памяти контроллера

**ПРИМЕЧАНИЕ**

Для переноса строки в вводимом тексте следует использовать комбинацию клавиш **Ctrl+Enter**.

При нажатии **ПКМ** на любой области списка текстов появляется контекстное меню:

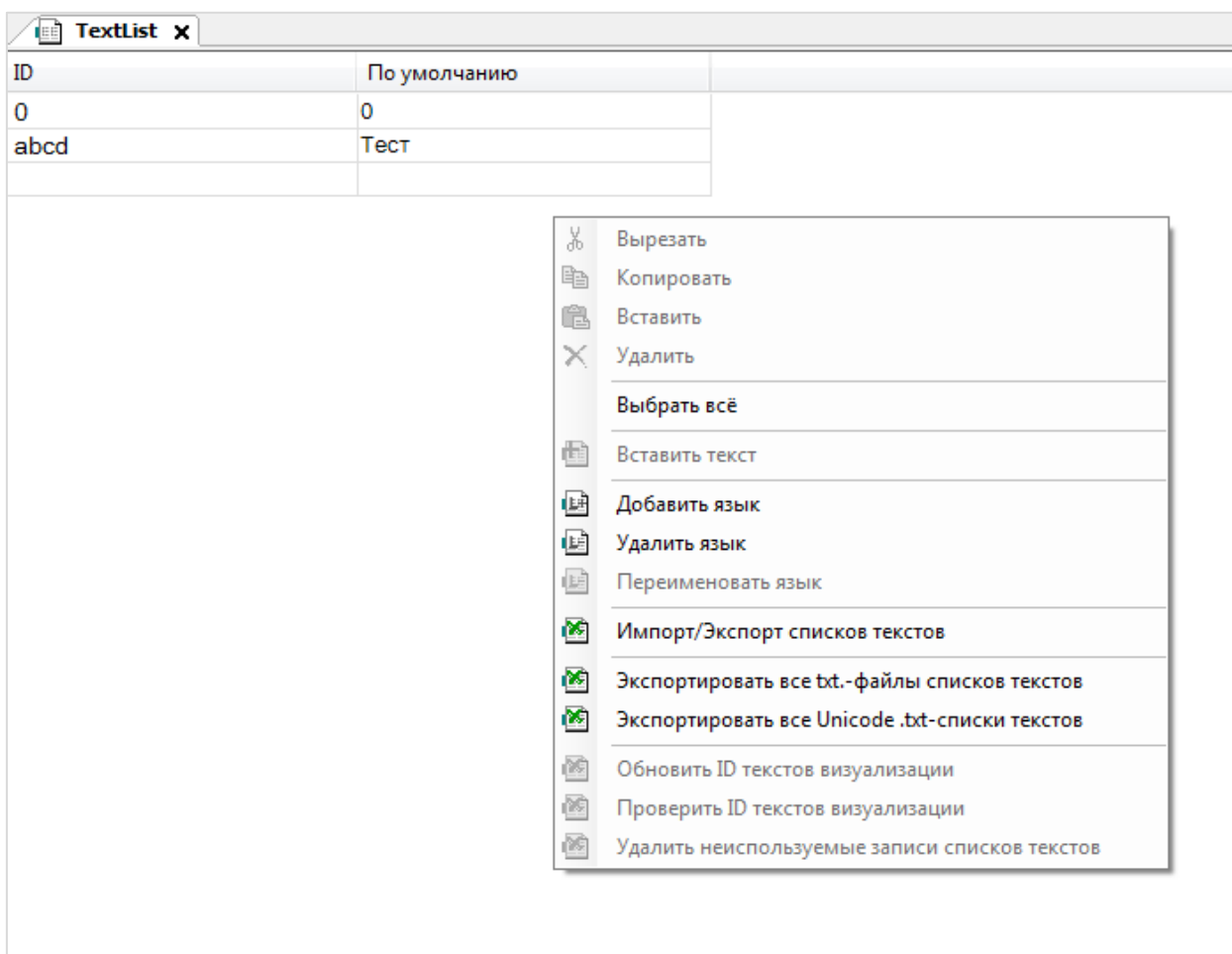


Рис. 2.4. Контекстное меню списка текстов

Команды **Вырезать/Копировать/Вставить/Удалить/Выбрать всё/Вставить текст** являются стандартными для текстовых редакторов. Остальные команды описаны ниже:

- **Добавить язык** – добавляет в список текстов новый столбец с заданным названием (см. информацию о реализации мультязычных проектов в [п. 3](#)). Этот язык становится доступным для выбора в **Менеджере визуализации** и действию **Изменить язык** (в событиях вкладки параметров **Конфигурация ввода** для элементов визуализации);
- **Удалить язык** – удаляет из списка текстов столбец, ячейка которого выделена в данный момент. Удалить столбец **По умолчанию** нельзя;
- **Переименовать язык** – переименовывает столбец, ячейка которого выделена в данный момент. Переименовать столбец **По умолчанию** нельзя;
- **Импорт/Экспорт списков текстов** – позволяет импортировать/экспортировать все списки текстов проекта в **единый** текстовый файл. Подробное описание команды приведено в [онлайн-справке CODESYS](#);

- **Экспортировать все .txt-файлы списков текстов** – экспортирует все списки текстов в виде отдельных файлов с кодировкой **ASCII** по пути, заданному в настройках среды (рис. 2.5) или проекта (рис. 2.7).
- **Экспортировать все Unicode .txt-списки текстов** – экспортирует все списки текстов в виде отдельных файлов с кодировкой **UCS2** по пути, заданному в настройках среды (рис. 2.5) или проекта (рис. 2.7). **Обратите внимание**, что в версиях CODESYS **V3.5 SP14** и **V3.5 SP15** данная команда работает некорректно (**CDS-66475**) – для экспортируемых файлов используется кодировка **ASCII**. Этот баг исправлен в версии **CODESYS V3.5 SP16**;
- **Проверить ID текстов визуализации, Обновить ID текстов визуализации** – эти команды могут использоваться только для системного списка текстов **GlobalTextList** в тех случаях, когда для него настроен контроль доступа через управление пользователями проекта. Они применяются для синхронизации статических текстов визуализаций и содержимого данного списка текстов;
- **Удалить неиспользуемые записи списки текстов** – эта команда может использоваться только для системного списка текстов **GlobalTextList**. Она позволяет удалить из списка текстов записи, которые больше не используются в визуализации (например, после удаления содержащего их ранее элемента).

В меню **Инструменты – Опции** на вкладке **Визуализация – Опции файла** можно указать директорию ПК, в которую будут экспортированы файлы списков текстов после выполнения команд **Экспортировать все .txt-файлы списков** и **Экспортировать все Unicode .txt-списки текстов**.

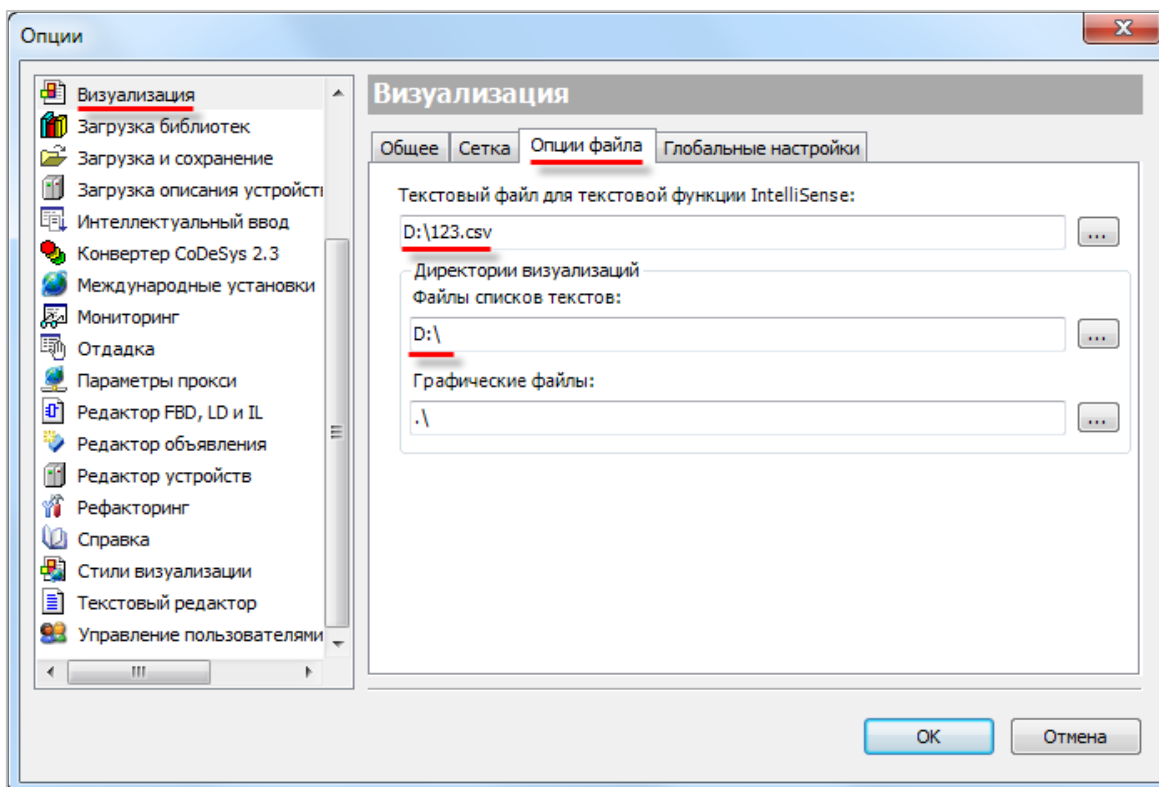


Рис. 2.5. Настройки списков текстов в меню **Инструменты – Опции – Визуализация – Опции файла**

В пункте **Текстовый файл для текстовой функции IntelliSense** (см. рис. 2.5) можно указать путь к общему файлу списка текстов, созданному с помощью команды **Импорт/Экспорт списков текстов** (также пользователь может сам создать файл подобного формата или отредактировать экспортированный).

В результате при наборе текстов в элементах визуализации (параметры **Тексты/Текст** и **Тексты/Подсказка**) будут отображаться предлагаемые варианты из подключенного файла:

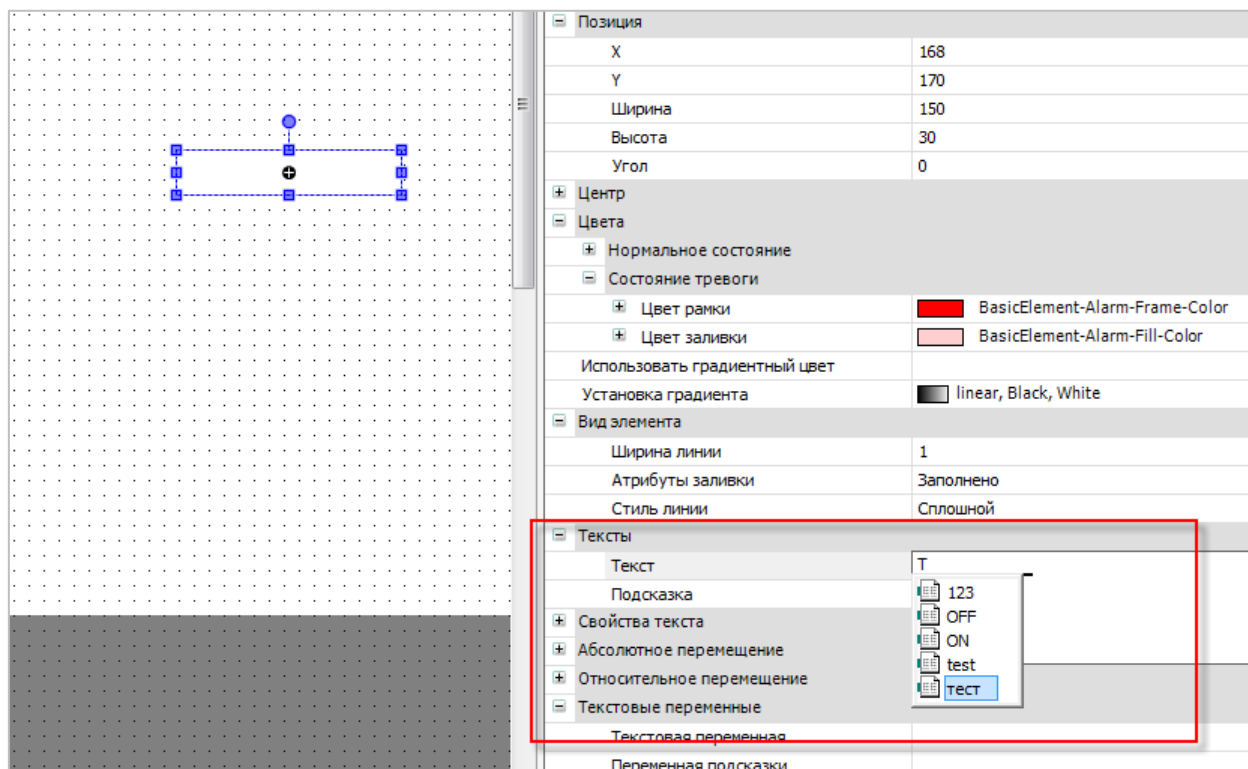


Рис. 2.6. Автодополнение при вводе текстов визуализации

В меню **Проект – Установки проекта** на вкладке **Визуализация – Общее** (см. рис. 2.7) можно указать директорию ПК, в которую будут экспортированы файлы списков текстов после выполнения команд **Экспортировать все .txt-файлы списков** и **Экспортировать все Unicode .txt-списки текстов**. В отличие от аналогичной настройки из меню **Инструменты – Опции** (вкладка **Визуализация – Опции файла**) – данная настройка имеет приоритет и сохраняется на уровне конкретного проекта CODESYS (а не на уровне среды разработки).

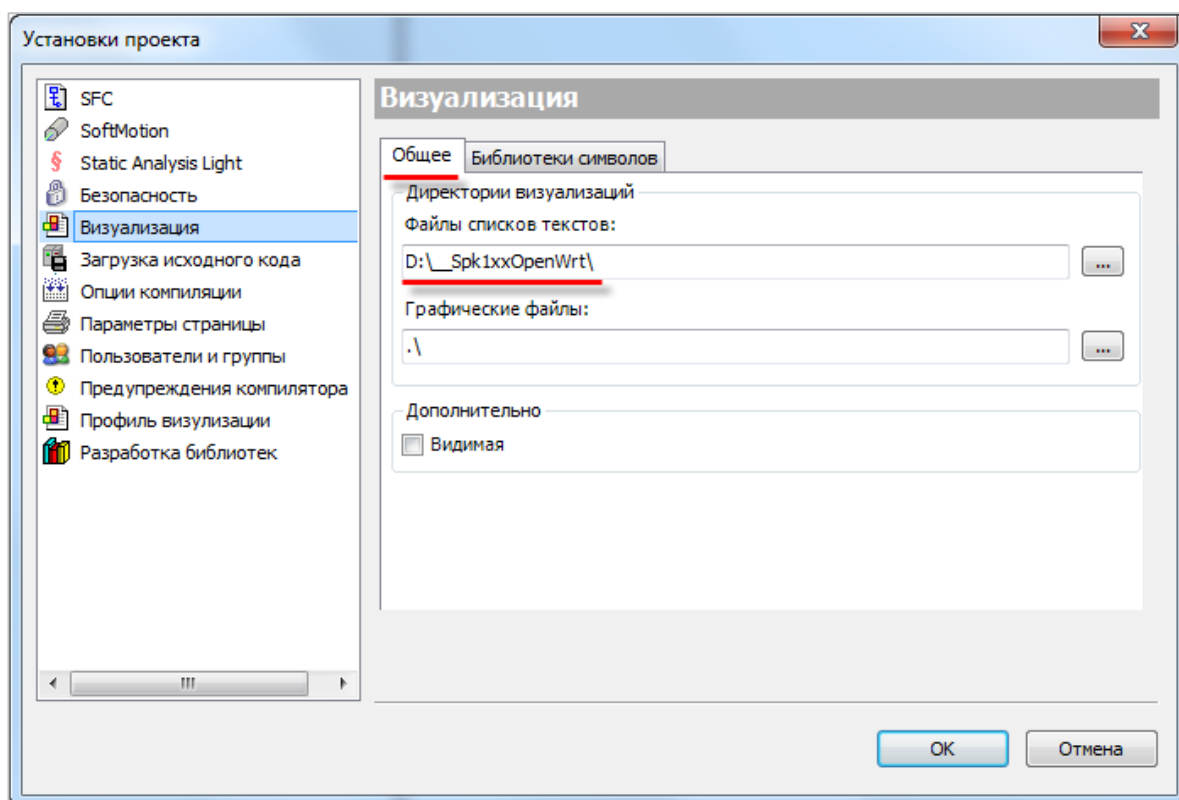


Рис. 2.7. Настройки списков текстов в меню **Проект – Установки проекта – Визуализация – Общее**

Если нажать на список текстов в дереве проекта **ПКМ** и выбрать команду **Свойства**, то в открывшемся окне будет доступна вкладка **Список текстов**. Если установлена галочка **Загрузка по визуализации**, то файл списка текстов будет загружаться в контроллер при загрузке проекта (если галочка снята – то файл загружаться не будет; это может быть полезным, если в контроллере уже есть файл данного списка текстов, который был как-то отредактирован в процессе работы программы – например, могли быть изменены названия рецептов и т.д.). Галочка **Внутренний** может быть использована в том случае, если список текстов создан в библиотеке – при ее добавлении данный список текстов не будет доступен в проекте.

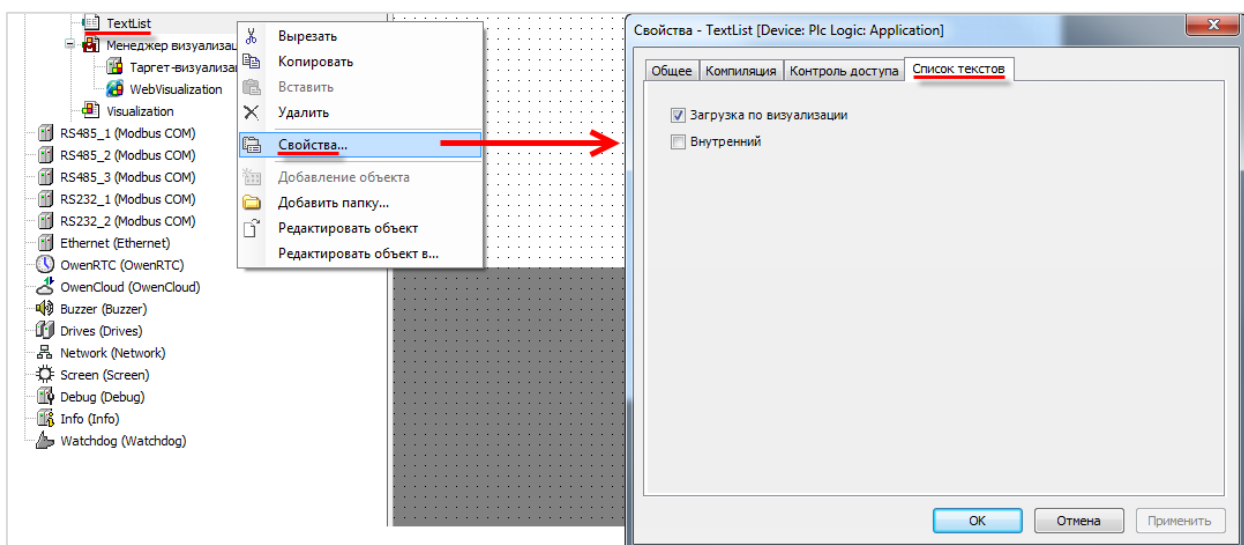
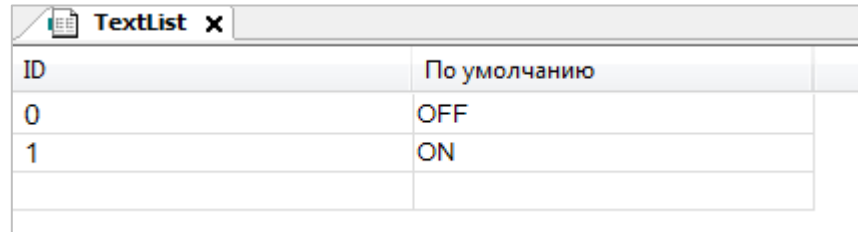


Рис. 2.8. Настройки списков текстов в меню **Свойства**

3. Использование динамических текстов в визуализации

Динамические тексты позволяют изменить текст элемента визуализации в процессе работы контроллера. Ниже приведен пример настройки динамического текста.

1. Создайте список текстов **TextList** со следующим содержимым:



ID	По умолчанию
0	OFF
1	ON

Рис. 3.1. Содержимое списка текстов

2. В программе **PLC_PRG** объявите переменную **iTextIndex** типа **INT**.

3. Добавьте элемент визуализации (например, **Прямоугольник**) и в его свойствах на вкладке **Динамические тексты** в параметре **Список текстов** укажите список текстов **TextList**, а к параметру **Индекс текста** привяжите переменную **PLC_PRG.iTextIndex**.

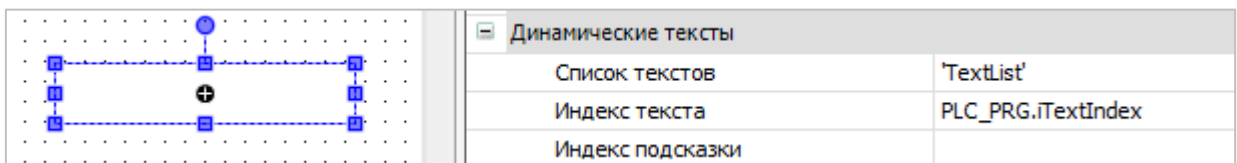


Рис. 3.2. Привязка списка текстов к элементу визуализации

4. Загрузите проект в контроллер или запустите его в эмуляции. Пока переменная **iTextIndex** будет иметь значение **0** – в элементе будет отображаться текст **OFF**. Если присвоить переменной значение **1**, то в элементе отобразится текст **ON**.



ПРИМЕЧАНИЕ

Параметр **Индекс подсказки** позволяет переключать текст подсказки, всплывающей при наведении на элемент курсора.



ПРИМЕЧАНИЕ

Параметры **Индекс текста** и **Индекс подсказки** имеют тип **STRING**. В рамках примера показан типичный случай, когда для списка текстов выбираются целочисленные значения ID и к параметру привязывается целочисленная переменная, которая автоматически будет конвертироваться в строку. Можно было бы использовать переменную типа **BOOL** (в рамках примера назовем ее **xSwitchText**) – тогда в параметре потребовалось бы указать выражение **TO_INT(PLC_PRG.xSwitchText)**. Явная конверсия к **INT** требуется из-за того, что без нее значение логической переменной при конвертации в строку выглядело бы как 'FALSE' или 'TRUE'.

4. Создание мультязычной визуализации

Рассмотрим пример создания мультязычной визуализации:

1. Создайте список текстов **TextList** со следующим содержанием:

ID	По умолчанию	en	ru
0	default text 0	text 0	текст 0
1	default text 1	text 1	текст 1

Рис. 4.1. Содержимое списка текстов

2. Добавьте элемент визуализации (например, **Прямоугольник**) и в его свойствах на вкладке **Динамические тексты** в параметре **Список текстов** укажите список текстов **TextList**.

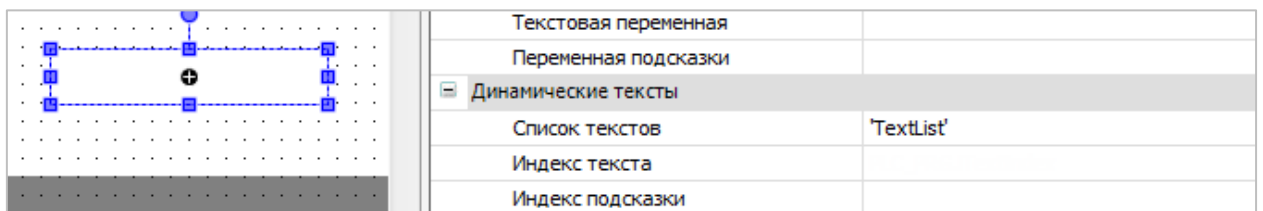


Рис. 4.2. Привязка списка текстов к элементу визуализации

3. Добавьте на экран две кнопки. В настройках кнопок на вкладке **Конфигурация ввода** добавьте событие **OnClick** и выберите действие **Изменить язык**. Укажите для одной кнопки язык 'en', а для второй – 'ru'.

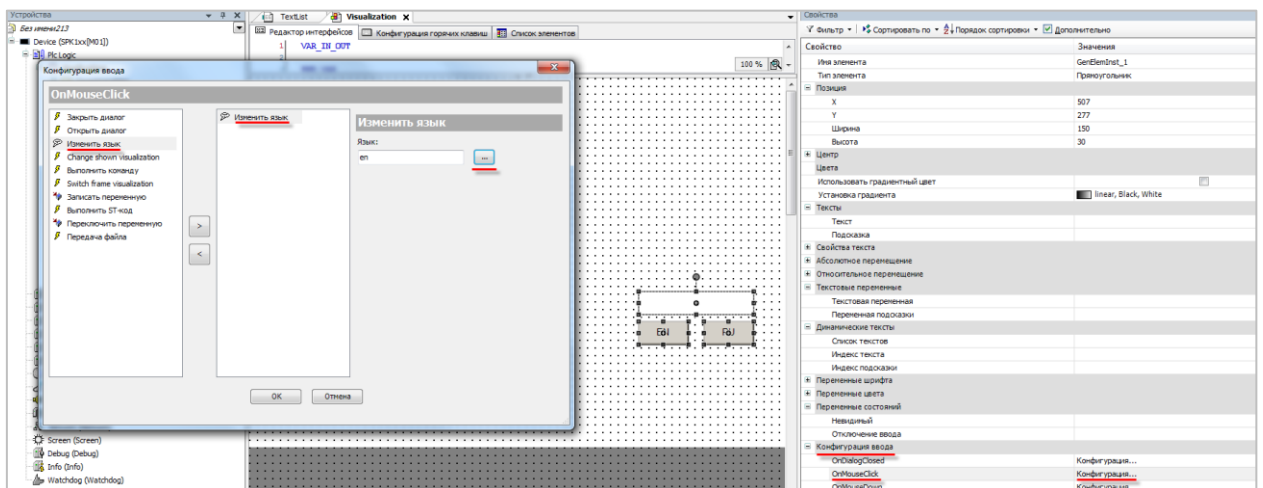


Рис. 4.3. Настройка действия **Изменить язык**

4. Загрузите проект в контроллер или запустите его в эмуляции. Нажимайте кнопки для переключения языков проекта.

**ПРИМЕЧАНИЕ**

Детальный пример по созданию мультязычного проекта приведен в [этом видео](#).

**ПРИМЕЧАНИЕ**

Язык проекта, используемый по умолчанию (в частности, после загрузки контроллера), выбирается в **Менеджере визуализации** на вкладке **Установки**. Пустая строка соответствует текстам столбца **По умолчанию**. В данном меню по умолчанию доступны языки **en/ru/de/fr/it/es/zh-CHS/ja**. Пользователь может добавлять в списки текстов языки с данными названиями или же произвольными названиями – во втором случае эти языки автоматически станут доступны для выбора в данном меню.

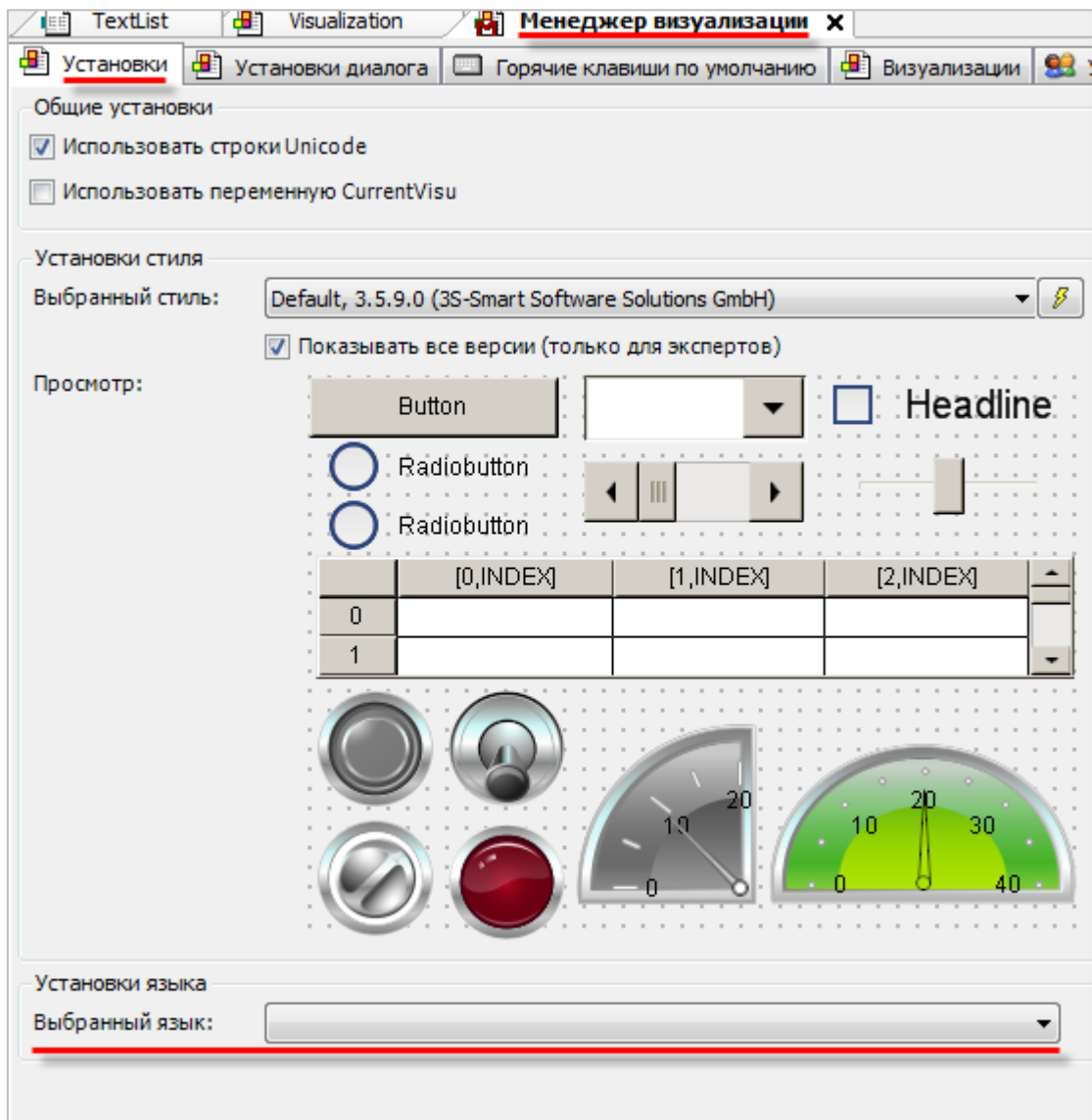


Рис. 4.4. Выбор языка визуализации в **Менеджере визуализации**

**ПРИМЕЧАНИЕ**

Для переключения языка визуализации из кода программы можно воспользоваться системной переменной **VisuElems.CURRENTLANGUAGE**. Переменная имеет тип **STRING**. В нее следует записать название языка, совпадающее с именем столбца из списка текстов ('en', 'ru' и т.д.). Также переменная может использоваться для определения текущего языка визуализации.

**ПРИМЕЧАНИЕ**

Язык визуализации является глобальным – т.е. переключается сразу для всех клиентов (в частности, это касается клиентов web-визуализации). В баг-трекере CODESYS зафиксировано пожелание по возможности переключения языка для конкретных клиентов (**CDS-42991**). Оно было реализовано в версии **V3.5 SP20** и плагине визуализации **4.7.0.0**. Соответственно, с этого момента упомянутая в предыдущем примечании системная переменная **VisuElems.CURRENTLANGUAGE** утратила свой смысл. См. подробности в [ЭТОМ ВИДЕО](#).

CODESYS V3 / CDS-42991 26 of 74

Visu, WebVisu: The language in the visualization should be clientspecific

Agile Board More Export

Details

Type:	+ New Feature	Status:	OPEN (View Workflow)
Affects Version/s:	None	Resolution:	Unresolved
Component/s:	Web Visualization	Fix Version/s:	Check for next SP
Labels:	None		
Target User Group:	End User		
Requested Version:	V3.5 SP16		

People

Assignee:	Administrator
Reporter:	Mirroring Service
Votes:	Remove vote for this issue
Watchers:	Stop watching this issue

Description

The language in the visualization should be clientspecific. It should be possible that different clients (especially web clients) can have different languages.

=== References ===

RS (final):
H:\Produktentwicklung\UpToDate\Spezifikationen\Visualisierung\RS_ClientabhangigeSprache.docx

FS (final):
H:\Produktentwicklung\UpToDate\Spezifikationen\Visualisierung\FS_ClientabhangigeSprache.docx

RS (dev):
svn://server04/DevelopmentDocs/trunk/Development/Visualization/Localization/RS_Clientabhän

FS (dev):
svn://server04/DevelopmentDocs/trunk/Development/Visualization/Localization/FS_Clientabhän

Dates

Due:	31/12/19
Created:	16/03/15 19:29
Updated:	29/04/20 14:34
Resolved:	16/03/15 19:30

Agile

View on Board

Рис. 4.5. Пожелание по индивидуальному переключению языка визуализации для конкретных клиентов в баг-трекере CODESYS

5. Перечисления с поддержкой текстов. Элемент визуализации «Выпадающий список»

В редакторе визуализации CODESYS на вкладке элементов **Стандартные элементы управления** присутствует элемент **Выпадающий список** (в русской локализации он назван **Комбинированное окно – Целочисленный**). Этот элемент предназначен для выбора значения из списка доступных (например, для выбора режима работы оборудования). К элементу привязывается список текстов и целочисленная переменная, в которую записывается ID выбранного элемента списка (таким образом, ID в списке текстов должен быть представлен целым числом). Соответственно, в коде программы потребуется установить соответствие между числом и его описанием из списка текстов. Обычно для решения подобных задач используется тип данных **Перечисление (ENUM)**. Поэтому для облегчения этой ситуации в **CODESYS V3.5 SP9** появилась поддержка списков текстов для перечислений. Ниже приведен пример их использования.

1. Добавьте в проект перечисление **COLOR** с поддержкой списка текстов (**ПКМ** на узел **Application – Добавить объект – DUT**, выбрать тип **Перечисление**, установить галочку **Поддержка списка текстов**).

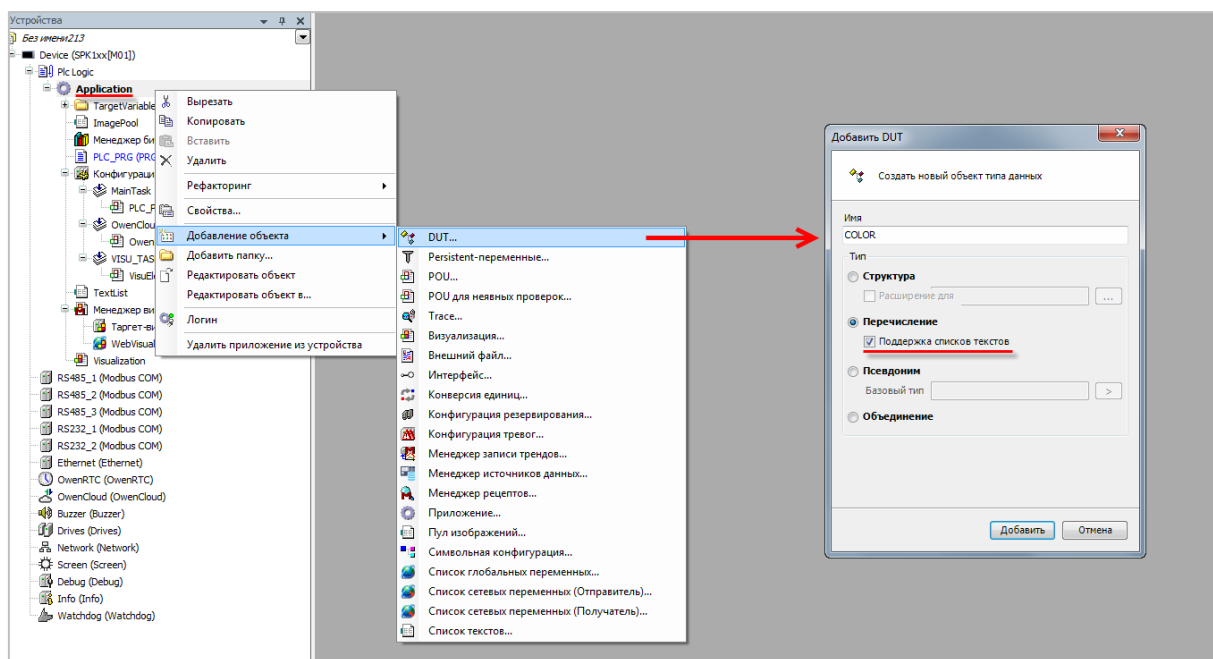


Рис. 5.1. Добавление в проект перечисления с поддержкой списка текстов

```

1  {attribute 'qualified_only'}
2  {attribute 'strict'}
3  TYPE COLOR :
4  (
5      RED   := 0,
6      GREEN := 1,
7      BLUE  := 2
8  );
9  END_TYPE
10

```

Рис. 5.2. Содержимое перечисления **COLOR**

2. Нажмите кнопку **Локализация**, чтобы переключить перечисление в режим списка текстов. Добавьте язык 'ru' и задайте тексты для элементов перечисления.

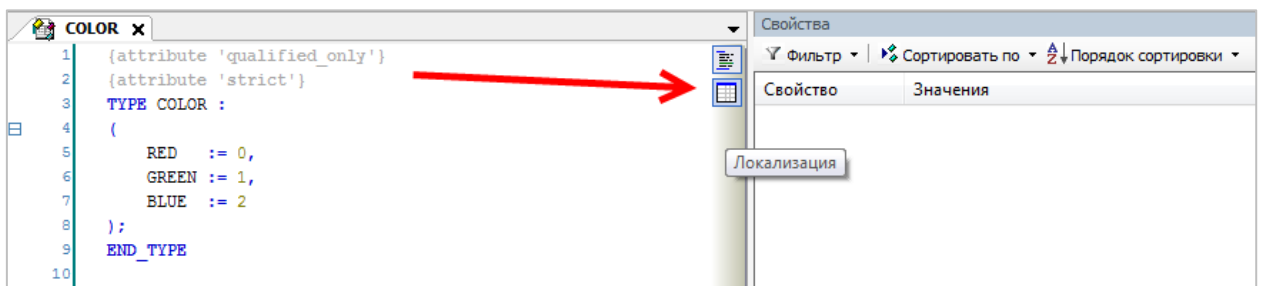


Рис. 5.3. Переключение перечисления в режим списка текстов

Member	Value	ru
RED	0	Красный
GREEN	1	Зеленый
BLUE	2	Синий

Рис. 5.4. Содержимое списка текстов перечисления

3. Объявите в программе **PLC_PRG** переменную **eColor** типа **COLOR**.

4. В менеджере визуализации выберите язык 'ru' (см. рис. 4.4).

5. В визуализации добавьте элемент **Комбинированное окно – Целочисленный** и привяжите к его параметру **Переменная** объявленную переменную. **Обратите внимание**, что после названия переменной автоматически отобразится название перечисления (точнее – название списка текстов, одноименного перечислению).

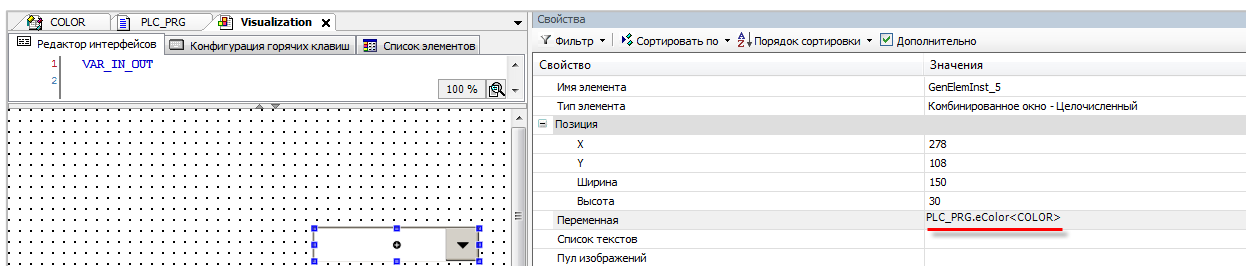


Рис. 5.5. Привязка экземпляра перечисления к элементу **Комбинированное окно – Целочисленный**

6. Загрузите проект в контроллер или запустите его в эмуляции. Выберите значение с помощью выпадающего списка.

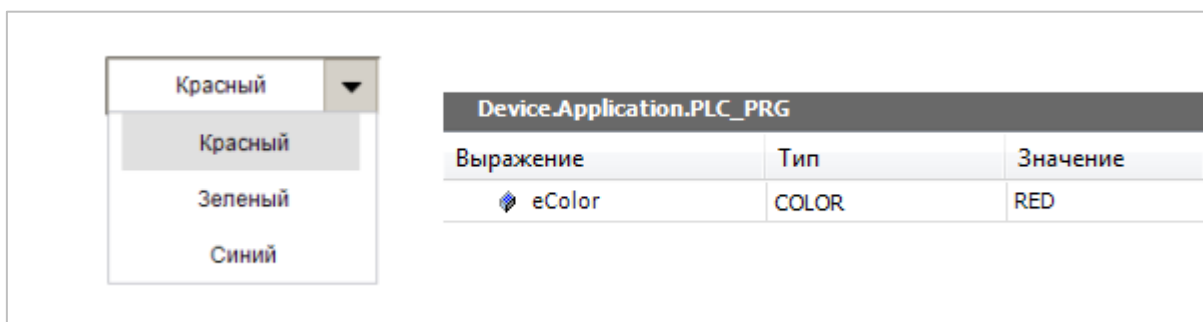


Рис. 5.6. Работа с элементом в визуализации



ПРИМЕЧАНИЕ

В версиях CODESYS V3.5 SP15 и SP16 данный функционал работает некорректно – в визуализации в выпадающем списке отображаются целочисленные значения, диапазон которых не ограничен диапазоном перечисления. Исправление бага ожидается в одной из следующих версий CODESYS.



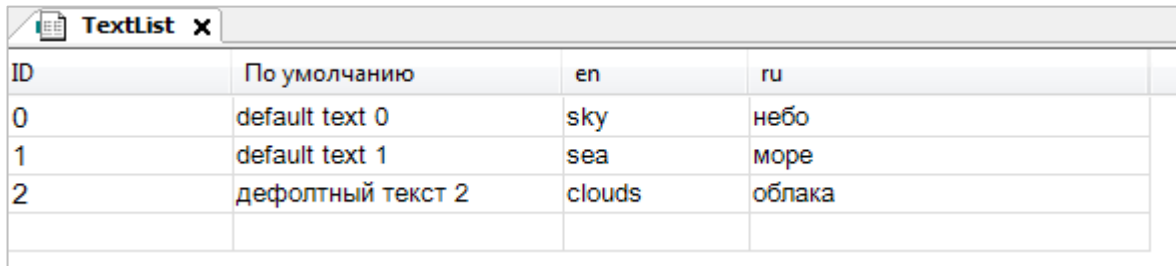
ПРИМЕЧАНИЕ

Вместо перечисления со встроенным списком текстов к элементу **Комбинированное окно – Целочисленный** можно привязать целочисленную переменную и указать отдельно созданный список текстов с целочисленными ID. Начиная с версии CODESYS **V3.5 SP15** вместо конкретного списка текстов можно привязать переменную типа **STRING**, в которую будет записываться имя нужного списка текстов – это позволяет переключать списки текстов, привязанные к элементу, в процессе работы проекта.

6. Библиотека CmpDynamicText

6.1. Основная информация

Библиотека **CmpDynamicText** позволяет считывать записи списков текстов в переменные программы. Далее описываются функции библиотеки (для версии **3.5.20.0**) на примере работы со списком **TextList**, который имеет следующее содержимое:



ID	По умолчанию	en	ru
0	default text 0	sky	небо
1	default text 1	sea	море
2	дефолтный текст 2	clouds	облака

Рис. 6.1. Содержимое списка текстов

Так как большинство функций библиотеки работают с файлами в памяти контроллера – то следует вызывать их событийно, а не циклически. Все функции библиотеки являются синхронными и выполняются в пределах одного цикла задачи контроллера. Пожелание по добавлению асинхронных ФБ для работы со списками текстов зафиксировано в баг-трекере CODESYS (**VSPRT-232**).

Ссылка на пример, в котором демонстрируется использование некоторых функций библиотеки (создан в **CODESYS V3.5 SP14 Patch 3**):

[Example_WorkingWithTextListFromIecCode_3514v1.projectarchive](http://3514v1.projectarchive.com/Example_WorkingWithTextListFromIecCode)

6.2. Функция DynamicTextChangeLanguage

Функция **DynamicTextChangeLanguage** позволяет установить язык списка текстов, который будет использоваться при вызове функций [DynamicTextGetText](#) и [DynamicTextGetTextW](#). Если на вход функции подана строка, содержимое которой не совпадает с названием ни одного из языков проекта – то выбирается «дефолтный» язык (т.е. язык с текстами столбца **По умолчанию**). Функция всегда возвращает **TRUE**.

Обратите внимание, что при вызове функции также переключается язык визуализации.

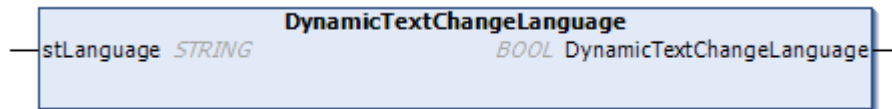


Рис. 6.2. Внешний вид функции **DynamicTextChangeLanguage** на языке CFC

6.3. Функция DynamicTextGetLanguage

Функция **DynamicTextGetLanguage** возвращает название языка, который был установлен в проекте с помощью действия **Изменить язык** (настроенным во вкладке **Конфигурация ввода** элемента визуализации), системной переменной **VisuElems.CURRENTLANGUAGE** или вызова функции [DynamicTextChangeLanguage](#).

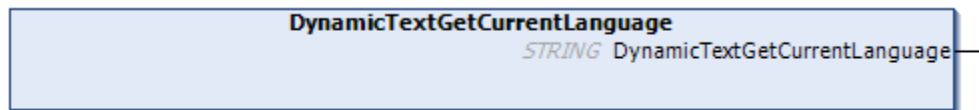


Рис. 6.3. Внешний вид функции **DynamicTextGetLanguage** на языке CFC

6.4. Функция DynamicTextGetDefaultText

Функция **DynamicTextGetDefaultText** позволяет считать текст по заданному ID из заданного списка текстов для «дефолтного» языка (см. в списке текстов столбец **По умолчанию**). ID и название списка текстов передаются в функцию в виде указателей на переменные типа **STRING**. Функция возвращает указатель на значение типа **STRING**.

Функция может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать функцию [DynamicTextGetDefaultTextW](#).

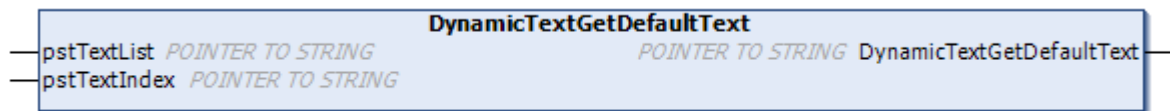


Рис. 6.4. Внешний вид функции **DynamicTextGetDefaultText** на языке CFC

```

PROGRAM PLC_PRG
VAR
  xGetDefaultText:  BOOL;

  sTextListName:   STRING := 'TextList';
  sTextId:         STRING := '0';
  psText:         POINTER TO STRING;

  sText:          STRING;
END_VAR

IF xGetDefaultText THEN

sText := CmpDynamicText.DynamicTextGetDefaultText (ADR(sTextListName), ADR(sTextId) ) ^;
xGetDefaultText := FALSE;

END_IF

```

Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetDefaultText	BOOL	FALSE			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
psText	POINTER TO STRING	16#00000000			
sText	STRING	'default text 0'			


```

1
2 IF xGetDefaultText FALSE THEN
3
4   sText := CmpDynamicText.DynamicTextGetDefaultText (ADR(sTextListName 'TextList' ), ADR(sTextId '0' ) ) ^;
5   xGetDefaultText FALSE := FALSE;
6
7 END_IF RETURN

```

Рис. 6.5. Пример использования функции **DynamicTextGetDefaultText** на языке ST

6.5. Функция DynamicTextGetDefaultTextW

Функция **DynamicTextGetDefaultTextW** позволяет считать текст по заданному ID из заданного списка текстов для «дефолтного» языка (см. в списке текстов столбец **По умолчанию**). ID и название списка текстов передаются в функцию в виде указателей на переменные типа **STRING**. Функция возвращает указатель на значение типа **STRING**, но фактически по указателю размещается значение типа **WSTRING**.

Функция используется для чтения текстов, которые включают в себя символы Unicode. Для чтения записей в ASCII-кодировке можно использовать функцию [DynamicTextGetDefaultText](#) (но также можно использовать и данную функцию). Для корректной работы функции в Менеджере визуализации должна быть установлена галочка **Использовать строки Unicode**.

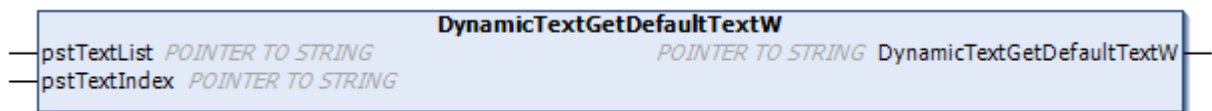


Рис. 6.6. Внешний вид функции **DynamicTextGetDefaultTextW** на языке CFC

```

PROGRAM PLC_PRG
VAR
  xGetDefaultTextW: BOOL;

  sTextListName:   STRING := 'TextList';
  sTextId:         STRING := '2';
  pwsText:         POINTER TO WSTRING;

  wsText:          WSTRING;
END_VAR

IF xGetDefaultTextW THEN

pwsText := CmpDynamicText.DynamicTextGetDefaultTextW (ADR(sTextListName) , ADR(sTextId) );
wsText  := pwsText^;
xGetDefaultTextW := FALSE;

END_IF
  
```

Device.Application.PLC_PRG					
Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetDefaultTextW	BOOL	FALSE			
sTextListName	STRING	'TextList'			
sTextId	STRING	'2'			
pwsText	POINTER TO WSTRING	16#008A8458			
wsText	WSTRING	"дефолтный текст 2"			


```

1
2 IF xGetDefaultTextW FALSE THEN
3
4   pwsText 16#008A8458 := CmpDynamicText.DynamicTextGetDefaultTextW (ADR(sTextListName TextList) , ADR(sTextId 2) );
5   wsText дефолтный := pwsText^ дефолтный ;
6   xGetDefaultTextW FALSE := FALSE;
7
8 END_IF RETURN
  
```

Рис. 6.7. Пример использования функции **DynamicTextGetDefaultTextW** на языке ST

6.6. Функция DynamicTextGetText

Функция **DynamicTextGetText** позволяет считать текст по заданному ID из заданного списка текстов для текущего языка проекта (язык может быть установлен, например, с помощью функции [DynamicTextChangeLanguage](#)). ID и название списка текстов передаются в функцию в виде указателей на переменные типа **STRING**. Функция возвращает указатель на значение типа **STRING**.

Функция может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать функцию [DynamicTextGetTextW](#).

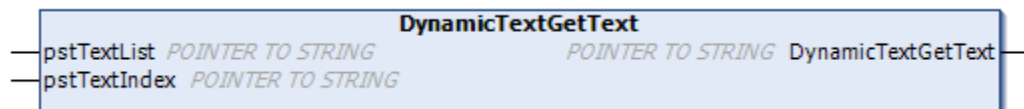


Рис. 6.8. Внешний вид функции **DynamicTextGetText** на языке CFC

В примере ниже считывается текст языка 'en'. Чтобы исключить переключение текстов в визуализации – перед считыванием текста определяется текущий язык проекта (с помощью функции [DynamicTextGetLanguage](#)) и после считывания данный язык снова устанавливается в проекте (с помощью функции [DynamicTextChangeLanguage](#)).

```
PROGRAM PLC_PRG
VAR
  xGetText:          BOOL;

  sCurrentLanguage:  STRING(20);

  sTextListName:     STRING := 'TextList';
  sTextId:           STRING := '0';
  psText:           POINTER TO STRING;

  sText:            STRING;
END_VAR

IF xGetText THEN

  sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
  CmpDynamicText.DynamicTextChangeLanguage('en');
  sText := CmpDynamicText.DynamicTextGetText(ADR(sTextListName), ADR(sTextId) )^;
  CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
  xGetText := FALSE;

END_IF
```

Device.Application.PLC_PRG					
Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetText	BOOL	FALSE			
sCurrentLanguage	STRING(20)	'en'			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
psText	POINTER TO STRING	16#00000000			
sText	STRING	'sky'			


```
1
2 IF xGetText FALSE THEN
3
4     sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
5     CmpDynamicText.DynamicTextChangeLanguage('en');
6     sText := CmpDynamicText.DynamicTextGetText(ADR(sTextListName), ADR(sTextId));
7     CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
8
9     xGetText := FALSE;
10 END_IF
11 RETURN
```

Рис. 6.9. Пример использования функции **DynamicTextGetText** на языке ST

6.7. Функция DynamicTextGetTextW

Функция **DynamicTextGetTextW** позволяет считать текст по заданному ID из заданного списка текстов для текущего языка проекта (язык может быть установлен, например, с помощью функции [DynamicTextChangeLanguage](#)). ID и название списка текстов передаются в функцию в виде указателей на переменные типа **STRING**. Функция возвращает указатель на значение типа **STRING**, но фактически по указателю размещается значение типа **WSTRING**.

Функция используется для чтения текстов, которые включают в себя символы Unicode. Для чтения записей в ASCII-кодировке можно использовать функцию [DynamicTextGetText](#) (но также можно использовать и данную функцию). Для корректной работы функции в Менеджере визуализации должна быть установлена галочка **Использовать строки Unicode**.

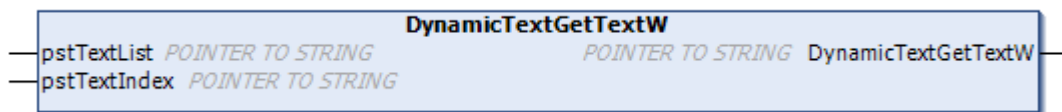


Рис. 6.10. Внешний вид функции **DynamicTextGetTextW** на языке CFC

В примере ниже считывается текст языка 'ru'. Чтобы исключить переключение текстов в визуализации – перед считыванием текста определяется текущий язык проекта (с помощью функции [DynamicTextGetLanguage](#)) и после считывания данный язык снова устанавливается в проекте (с помощью функции [DynamicTextChangeLanguage](#)).

```

PROGRAM PLC_PRG
VAR
  xGetTextW:          BOOL;

  sCurrentLanguage:  STRING(20);

  sTextListName:     STRING := 'TextList';
  sTextId:           STRING := '0';
  pwsText:           POINTER TO WSTRING;

  wsText:           WSTRING;
END_VAR

IF xGetTextW THEN

  sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
  CmpDynamicText.DynamicTextChangeLanguage('ru');
  pwsText := CmpDynamicText.DynamicTextGetTextW(ADR(sTextListName), ADR(sTextId) );
  wsText := pwsText^;
  CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
  xGetTextW := FALSE;

END_IF

```

Device.Application.PLC_PRG					
Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetTextW	BOOL	FALSE			
sCurrentLanguage	STRING(20)	'en'			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
pwsText	POINTER TO WSTRING	16#00911566			
wsText	WSTRING	"небо"			


```

1
2 IF xGetTextW[FALSE] THEN
3
4   sCurrentLanguage[en] := CmpDynamicText.DynamicTextGetCurrentLanguage();
5   CmpDynamicText.DynamicTextChangeLanguage('ru');
6   pwsText[16#00911566] := CmpDynamicText.DynamicTextGetTextW(ADR(sTextListName[TextList]), ADR(sTextId[0]));
7   wsText[небо] := pwsText[небо];
8   CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage[en]);
9
10  xGetTextW[FALSE] := FALSE;
11 END_IF
12 RETURN

```

Рис. 6.11. Пример использования функции **DynamicTextGetTextW** на языке ST

6.8. Функция DynamicTextLoadDefaultTexts

Функция **DynamicTextLoadDefaultTexts** перезагружает «дефолтные» тексты (из столбца **По умолчанию**) из файлов списков текстов, добавленных в проекте или созданных и зарегистрированных в процессе работы проекта с помощью функций [DynamicTextRegisterFile](#) или [DynamicFileRegisterPath](#).

Это позволяет обновить тексты в элементах визуализации, к которым привязаны списки текстов, в процессе работы проекта.

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.



ПРИМЕЧАНИЕ

Фактически функция перезагружает не только «дефолтные» тексты, а вообще все тексты из списков текстов – и при этом ее поведение совпадает с поведением функции [DynamicTextReloadTexts](#).

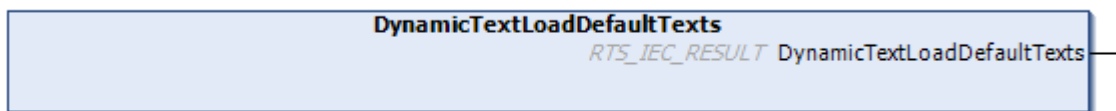


Рис. 6.12. Внешний вид функции **DynamicTextLoadDefaultTexts** на языке CFC

6.9. Функция DynamicTextRegisterFile

Функция **DynamicTextRegisterFile** регистрирует в системе исполнения файл, размещенный по пути **szFile**, как файл списка текстов. После этого файл обрабатывается при выполнении функций [DynamicTextLoadDefaultTexts](#) и [DynamicTextReloadTexts](#). Структура файла должна соответствовать ожидаемой (см. рис. 2.3).

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

См. также обратную функцию [DynamicTextUnRegisterFile](#).

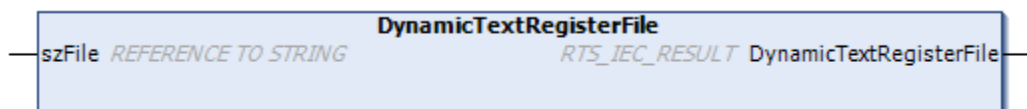


Рис. 6.13. Внешний вид функции **DynamicTextRegisterFile** на языке CFC

6.10. Функция DynamicTextRegisterPath

Функция **DynamicTextRegisterPath** регистрирует в системе исполнения директорию, размещенную по пути **szPath**, как директорию списков текстов. После этого все **.txt** файлы директории обрабатываются при выполнении функций [DynamicTextLoadDefaultTexts](#) и [DynamicTextReloadTexts](#). Структура файлов должна соответствовать ожидаемой (см. рис. 2.3).

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

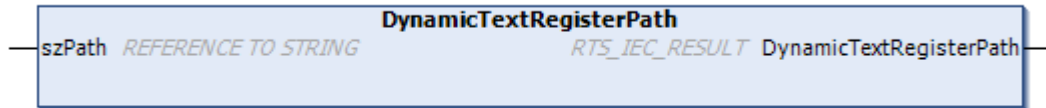


Рис. 6.14. Внешний вид функции **DynamicTextRegisterPath** на языке CFC

6.11. Функция DynamicTextReloadTexts

Функция **DynamicTextReloadTexts** перезагружает тексты текущего языка проекта (язык может быть установлен, например, с помощью функции [DynamicTextChangeLanguage](#)) из файлов списков текстов, добавленных в проекте или созданных и зарегистрированных в процессе работы проекта с помощью функций [DynamicTextRegisterFile](#) или [DynamicFileRegisterPath](#).

Это позволяет обновить тексты в элементах визуализации, к которым привязаны списки текстов, в процессе работы проекта.

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.



ПРИМЕЧАНИЕ

Фактически функция перезагружает не только тексты текущего языка проекта, а вообще все тексты из списков текстов – и при этом ее поведение совпадает с поведением функции [DynamicTextLoadDefaultTexts](#).

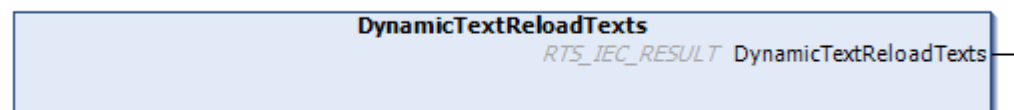


Рис. 6.15. Внешний вид функции **DynamicTextReloadTexts** на языке CFC

6.12. Функция DynamicTextUnRegisterFile

Функция **DynamicTextUnRegisterFile** отменяет регистрацию файла списков текстов, размещенного по пути **szFile** и ранее зарегистрированного в системе исполнения с помощью функции [DynamicTextRegisterFile](#).

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

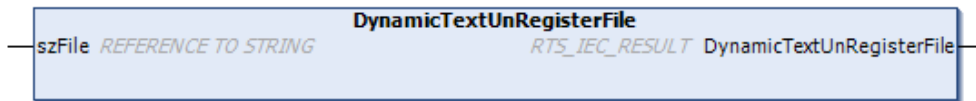


Рис. 6.16. Внешний вид функции **DynamicTextUnRegisterFile** на языке CFC

6.13. Функция DynamicTextIterateIndices

Функция **DynamicTextIterateIndices** позволяет итерационно обработать файл списка текстов для получения его ID. На вход **pstTextlist** передается указатель на переменную типа **STRING**, содержащую имя нужного списка текстов. На вход **pfIterationCallback** передается указатель на callback-функцию. Для каждой записи в списке текстов будет произведен вызов этой функции. Ожидаемая сигнатура функции:

- Вход типа **POINTER TO STRING**, которому автоматически будет присвоен указатель на строку, в которую будет записан ID текущей итерируемой записи списка текстов. Этот указатель нельзя разыменовывать напрямую – сначала требуется скопировать его в локальную переменную функции. Значение указателя действительно только во время вызова функции;
- Вход типа **__UXINT** (платформено-зависимый тип данных: для 32-битных систем исполнения ему соответствует тип **UDINT**, а 64-битных – **ULINT**). На этот вход автоматически передается значение **iterationData** (со входа функции **DynamicTextIterateIndices**). Это значение, например, может использоваться для указания максимально допустимого числа итераций;
- Выход типа **BOOL**. Если выводу присвоить значение **FALSE** – то обработка файла списков текстов прекращается (это можно сделать при получении искомого ID).

Обратите внимание, что порядок возвращаемых ID при итерациях не определен и может отличаться от их порядка в списке текстов.

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

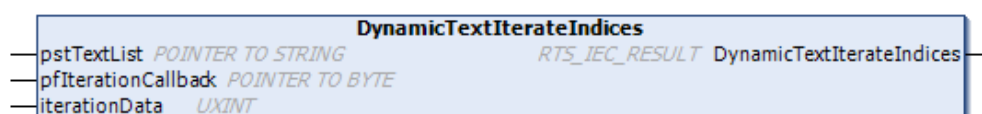


Рис. 6.17. Внешний вид функции **DynamicTextIterateIndices** на языке CFC

6.14. Комментарий к пунктам 6.15-6.24

Как было упомянуто в описании функций [DynamicTextGetText](#) и [DynamicTextGetTextW](#) – они возвращают текст из списка текстов на том языке, который в данный момент установлен в проекте. Из-за этого приходится совместно с ними использовать функцию [DynamicTextGetLanguage](#) для определения текущего языка проекта и функцию [DynamicTextChangeLanguage](#) для установки нужного языка списка текстов и последующего восстановления ранее установленного языка. В результате в момент вызова функций в визуализации происходит кратковременное отображение текстов на «другом» языке.

Очевидно, что это непрозрачное и неудобное для пользователя поведение. Информация об этом была передана разработчикам CODESYS (подробнее см. в [Заключении](#)), в результате чего в **CODESYS V3.5 SP20** библиотека **CmpDynamicText** была обновлена до версии **3.5.20.0**, в которой появились две новые папки с функциями:

- **DynamicLanguage**, которая содержит функции для работы с языками списков текстов;
- **LockHandling**, которая содержит функции для синхронизации доступа к спискам текстов на основе [семафора](#).

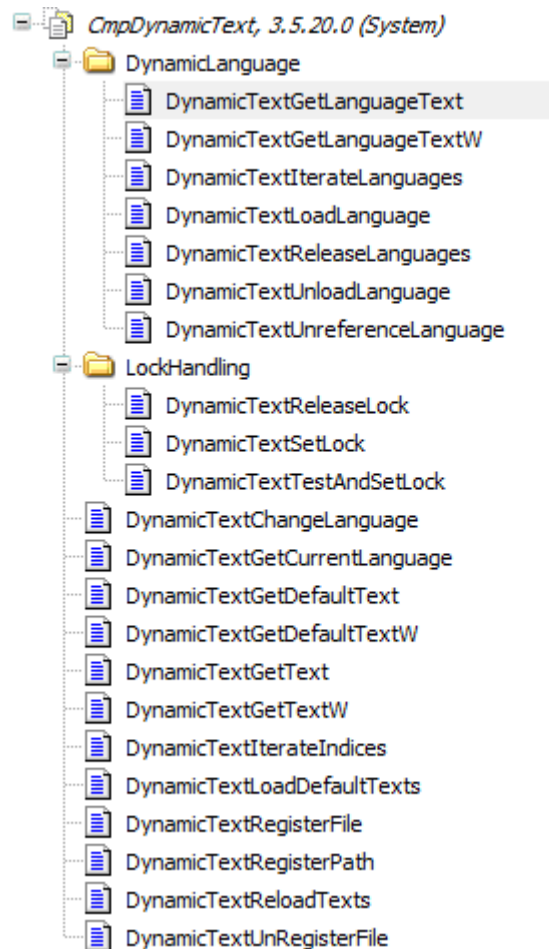


Рис. 6.18. Список функций библиотеки **CmpDynamicText** версии **3.5.20.0**

6.15. Функция DynamicTextGetLanguageText

Функция **DynamicTextGetLanguageText** позволяет считать текст по заданному ID (**stTextIndex**) из заданного списка текстов (**stTextList**) для заданного языка (**stLanguage**). Все входы функции имеют тип **STRING**. Функция возвращает указатель на значение типа **STRING**.

Перед вызовом функции необходимо загрузить тексты нужного языка в память с помощью функции [DynamicTextLoadLanguage](#). После чтения текста их можно выгрузить (см. [п. 6.18](#) – 6.20).

Если язык не загружен или список текстов/запись с заданным ID не существует, то функция возвращает указатель на пустую строку.

Функция может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать функцию [DynamicTextGetLanguageTextW](#).

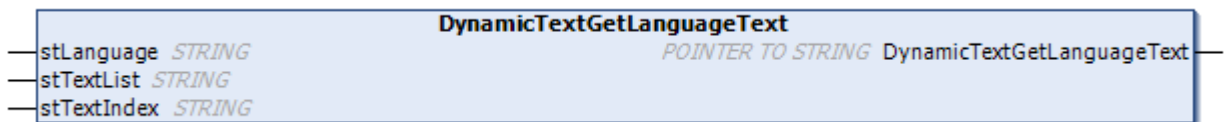


Рис. 6.19. Внешний вид функции **DynamicTextGetLanguageText** на языке CFC

```
PROGRAM PLC_PRG
VAR
  xGetText:          BOOL;

  sLanguage:        STRING := 'en';
  sTextListName:    STRING := 'TextList';
  sTextId:          STRING := '0';
  psText:           POINTER TO STRING;

  sText:            STRING;
END_VAR

IF xGetText THEN

  CmpDynamicText.DynamicTextLoadLanguage (sLanguage) ;
  sText := CmpDynamicText.DynamicTextGetLanguageText (sLanguage, sTextListName,
    sTextId) ^;
  CmpDynamicText.DynamicTextUnloadLanguage (sLanguage) ;
  xGetText := FALSE;

END_IF
```

Expression	Type	Value	Prepared value	Address	Comm...
xGetText	BOOL	FALSE			
sLanguage	STRING	'en'			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
psText	POINTE...	16#00000000			
sText	STRING	'sky'			


```

1  IF xGetText FALSE THEN
2
3  CmpDynamicText.DynamicTextLoadLanguage (sLanguage 'en' );
4  sText 'sky' := CmpDynamicText.DynamicTextGetLanguageText (sLanguage 'en', sTextListName 'TextList', sTextId '0' ) ^;
5  CmpDynamicText.DynamicTextUnloadLanguage (sLanguage 'en' );
6  xGetText FALSE := FALSE;
7
8  END_IF
```

Рис. 6.20. Пример использования функции **DynamicTextGetLanguageText** на языке ST

6.16. Функция DynamicTextGetLanguageTextW

Функция **DynamicTextGetLanguageTextW** позволяет считать текст по заданному ID (**stTextIndex**) из заданного списка текстов (**stTextList**) для заданного языка (**stLanguage**). Все входы функции имеют тип **STRING**. Функция возвращает указатель на значение типа **WSTRING**.

Перед вызовом функции необходимо загрузить тексты нужного языка в память с помощью функции [DynamicTextLoadLanguage](#). После чтения текста их можно выгрузить (см. [п. 6.18](#) – 6.20).

Если язык не загружен или список текстов/запись с заданным ID не существует, то функция возвращает указатель на пустую строку.

Функция используется для чтения текстов, которые включают в себя символы Unicode. Для чтения записей в ASCII-кодировке можно использовать функцию [DynamicTextGetLanguageText](#) (но также можно использовать и данную функцию). Для корректной работы функции в Менеджере визуализации должна быть установлена галочка **Использовать строки Unicode**.

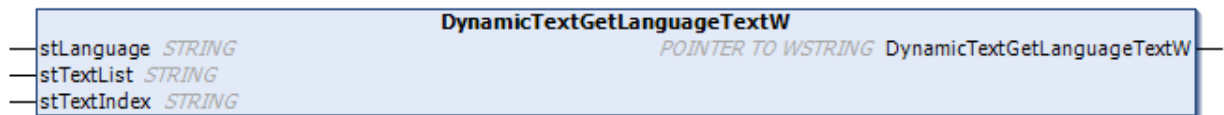


Рис. 6.21. Внешний вид функции **DynamicTextGetLanguageTextW** на языке CFC

```
PROGRAM PLC_PRG
VAR
  xGetText:          BOOL;

  sLanguage:        STRING := 'ru';
  sTextListName:    STRING := 'TextList';
  sTextId:          STRING := '1';
  pwsText:          POINTER TO WSTRING;

  wsText:           WSTRING;
END_VAR

IF xGetText THEN

  CmpDynamicText.DynamicTextLoadLanguage(sLanguage);
  wsText := CmpDynamicText.DynamicTextGetLanguageTextW(sLanguage, sTextListName,
    sTextId)^;
  CmpDynamicText.DynamicTextUnloadLanguage(sLanguage);
  xGetText := FALSE;

END_IF
```

Expression	Type	Value	Prepared value	Address	Comm...
xGetText	BOOL	FALSE			
sLanguage	STRING	'ru'			
sTextListName	STRING	'TextList'			
sTextId	STRING	'1'			
pwsText	POINTE...	16#00000000			
wsText	WSTRING	'nope'			


```

1 IF xGetText FALSE THEN
2
3   CmpDynamicText.DynamicTextLoadLanguage (sLanguage 'ru' );
4   wsText 'nope' := CmpDynamicText.DynamicTextGetLanguageTextW (sLanguage 'ru', sTextListName 'TextList', sTextId '1' );
5   CmpDynamicText.DynamicTextUnloadLanguage (sLanguage 'ru' );
6   xGetText FALSE := FALSE;
7
8 END_IF

```

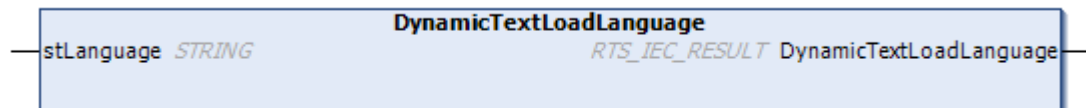
Рис. 6.22. Пример использования функции `DynamicTextGetLanguageTextW` на языке ST

6.17. Функция `DynamicTextLoadLanguage`

Функция `DynamicTextLoadLanguage` загружает тексты на заданном языке для возможности их последующего чтения функциями [DynamicTextGetLanguageText](#) и [DynamicTextGetLanguageTextW](#). В отличие от функции [DynamicTextChangeLanguage](#) – вызов данной функции не приводит к переключению языка визуализации проекта.

Если язык уже был загружен, то функция выполняет инкремент счетчика, соответствующего количеству загрузок этого языка.

Функция возвращает код ошибки в виде переменной типа `RTS_IEC_RESULT` из библиотеки `SysTypes`. Описание кодов ошибок приведено в библиотеке `CmpErrors`.

Рис. 6.23. Внешний вид функции `DynamicTextLoadLanguage` на языке CFC

6.18. Функция `DynamicTextUnloadLanguage`

Функция `DynamicTextUnloadLanguage` выполняет декремент счетчика, соответствующего количеству загрузок заданного языка. Если значение счетчика равно 0, то вызов функции выгружает язык из памяти.

Функция возвращает код ошибки в виде переменной типа `RTS_IEC_RESULT` из библиотеки `SysTypes`. Описание кодов ошибок приведено в библиотеке `CmpErrors`.

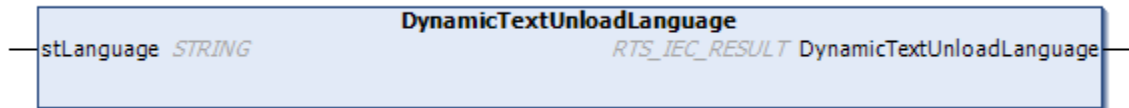


Рис. 6.24. Внешний вид функции `DynamicTextUnloadLanguage` на языке CFC

6.19. Функция `DynamicTextUnreferencedLanguage`

Функция `DynamicTextUnreferencedLanguage` выполняет декремент счетчика, соответствующего количеству загрузок заданного языка. Если значение счетчика равно 0, то вызов функции, в отличие от `DynamicTextUnloadLanguage`, не выгружает язык из памяти.

Функция возвращает код ошибки в виде переменной типа `RTS_IEC_RESULT` из библиотеки `SysTypes`. Описание кодов ошибок приведено в библиотеке `CmpErrors`.

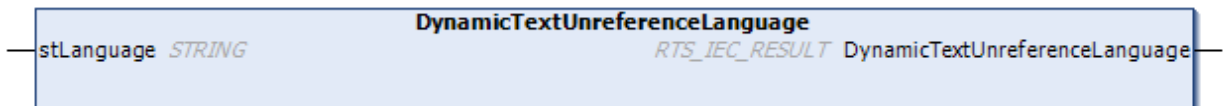


Рис. 6.25. Внешний вид функции `DynamicTextUnreferencedLanguage` на языке CFC

6.20. Функция DynamicTextReleaseLanguages

Функция **DynamicTextReleaseLanguages** проверяет для каждого языка значение счетчика, соответствующего количеству загрузок данного языка. Если значение счетчика равно 0, то выполняется выгрузка этого языка из памяти. Если вход **xForceRelease** имеет значение **TRUE**, то выполняется выгрузка всех языков вне зависимости от значения счетчиков.

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

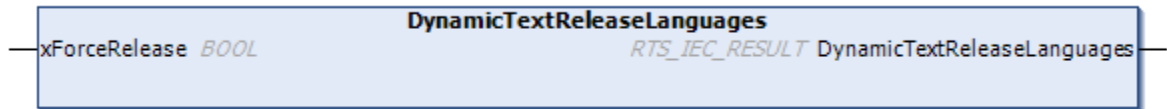


Рис. 6.26. Внешний вид функции **DynamicTextReleaseLanguages** на языке CFC

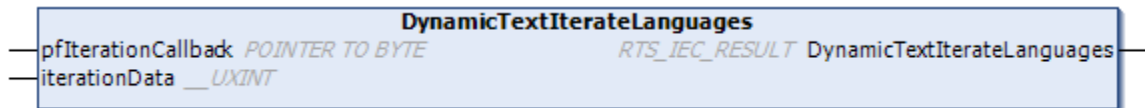
6.21. Функция DynamicTextIterateLanguages

Функция **DynamicTextIterateLanguages** позволяет получить список загруженных языков списков текстов проекта. На вход **pfIterationCallback** передается указатель на callback-функцию. Для каждого языка проекта будет произведен вызов этой функции. Ожидаемая сигнатура функции:

- Вход типа **POINTER TO STRING**, которому автоматически будет присвоен указатель на строку, в которую будет записано название языка списка текстов. Этот указатель нельзя разыменовывать напрямую – сначала требуется скопировать его в локальную переменную функции. Значение указателя действительно только во время вызова функции;
- Вход типа **__UXINT** (платформено-зависимый тип данных: для 32-битных систем исполнения ему соответствует тип **UDINT**, а 64-битных – **ULINT**). На этот вход автоматически передается значение **iterationData** (со входа функции **DynamicTextIterateLanguages**). Это значение, например, может использоваться для указания максимально допустимого числа итераций;
- Выход типа **BOOL**. Если выводу присвоить значение **FALSE** – то обработка языков прекращается.

Обратите внимание, что порядок возвращаемых названий при итерациях не определен и может отличаться от их порядка в списке текстов.

Функция возвращает код ошибки в виде переменной типа **RTS_IEC_RESULT** из библиотеки **SysTypes**. Описание кодов ошибок приведено в библиотеке **CmpErrors**.

Рис. 6.27. Внешний вид функции **DynamicTextIterateLanguages** на языке CFC

Ниже приведен пример использования функции.

Список глобальных переменных **GVL**:

```

{attribute 'qualified_only'}
VAR_GLOBAL
  // Список загруженных языков проекта
  asLanguageTitles:    ARRAY [1..c_usiLanguagesCount] OF STRING;
  // Индекс для доступа к массиву
  // Используется в функции GetLanguagesNames
  usiLanguageIndex:    USINT := 1;
END_VAR
VAR_GLOBAL CONSTANT
  // Максимальное количество языков, которое ожидается в проекте
  c_usiLanguagesCount: USINT := 3;
END_VAR

```

Callback-функция GetLanguagesNames:

```

FUNCTION GetLanguagesNames : BOOL
VAR_INPUT
  psLanguageName:    POINTER TO STRING;
  iterationData:     __UXINT;
END_VAR
VAR
  _psLanguageName:   POINTER TO STRING;
END_VAR

_psLanguageName := psLanguageName;

IF GVL.usiLanguageIndex = iterationData OR _psLanguageName^ = '' THEN
  GetLanguagesNames := FALSE;
ELSE
  GVL.asLanguageTitles[GVL.usiLanguageIndex] := _psLanguageName^;
  GVL.usiLanguageIndex:= GVL.usiLanguageIndex + 1;
END_IF
END_FUNCTION

```

Код программы с синтетическим примером использования функции:

```

PROGRAM PLC_PRG
VAR
    xGetLanguageTitles: BOOL;
END_VAR

IF xGetLanguageTitles THEN

    CmpDynamicText.DynamicTextLoadLanguage('ru');
    CmpDynamicText.DynamicTextLoadLanguage('en');
    CmpDynamicText.DynamicTextIterateLanguages(ADR(GetLanguagesNames),
        GVL.c_usiLanguagesCount);
    CmpDynamicText.DynamicTextReleaseLanguages(xForceRelease := TRUE);

    GVL.usiLanguageIndex:= 1;
    xGetLanguageTitles := FALSE;

END_IF

```

После того, как переменной **xGetLanguageTitles** будет присвоено значение **TRUE**, массив в **GVL** будет заполнен следующими значениями:

Device.Application.GVL		
Expression	Type	Value
asLanguageTitles	ARRAY ...	
asLanguageTitles[1]	STRING	'en'
asLanguageTitles[2]	STRING	'ru'
asLanguageTitles[3]	STRING	"
asLanguageIndex	USINT	1
c_usiLanguagesCount	USINT	3

Рис. 6.28. Результат вызова функции **DynamicTextIterateLanguages**

6.22. Функция `DynamicTextSetLock`

Функция `DynamicTextSetLock` используется для синхронизации доступа к спискам текстов с помощью [семафора](#). Это требуется, чтобы обеспечить согласованный доступ (чтение и запись) к спискам текстов из разных задач.

Вызов функции устанавливает блокировку на доступ к спискам текстов. Если блокировка успешно установлена, то функция возвращает значение `TRUE`.

Если блокировка уже установлена в контексте другой задачи, то данная задача (в которой произошел вызов функции `DynamicTextSetLock`) блокируется до того момента, пока она не сможет установить блокировку. Функция `DynamicTextSetLock` является атомарной; ее выполнение не может быть прервано другой задачей – в том числе, на многоядерных платформах.

Для снятия блокировки используется функция [DynamicTextReleaseLock](#); каждому вернувшему значение `TRUE` вызову функции `DynamicTextSetLock` должен соответствовать свой вызов функции [DynamicTextReleaseLock](#).

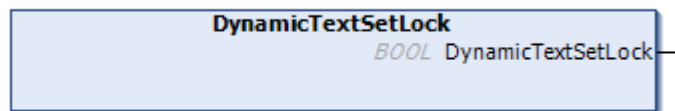


Рис. 6.29. Внешний вид функции `DynamicTextSetLock` на языке CFC

6.22. Функция `DynamicTextTestAndSetLock`

Функция `DynamicTextTestAndSetLock` используется для синхронизации доступа к спискам текстов с помощью [семафора](#). Это требуется, чтобы обеспечить согласованный доступ (чтение и запись) к спискам текстов из разных задач.

Вызов функции проверяет наличие блокировки на доступ к спискам текстов. Если блокировка не установлена, то функция устанавливает блокировку и возвращает значение `TRUE`. В отличие от функции [DynamicTextSetLock](#) – если блокировка уже была установлена, то задача, в которой произошел вызов функции `DynamicTextTestAndSetLock`, не оказывается заблокированной.

Функция `DynamicTextTestAndSetLock` является атомарной; ее выполнение не может быть прервано другой задачей – в том числе, на многоядерных платформах.

Для снятия блокировки используется функция [DynamicTextReleaseLock](#); каждому вернувшему значение `TRUE` вызову функции `DynamicTextTestAndSetLock` должен соответствовать свой вызов функции [DynamicTextReleaseLock](#).

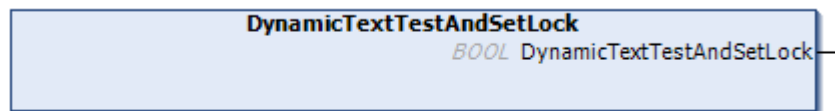


Рис. 6.30. Внешний вид функции `DynamicTextTestAndSetLock` на языке CFC

6.24. Функция `DynamicTextReleaseLock`

Функция `DynamicTextReleaseLock` снимает блокировку доступа к спискам текстов, установленную вызовом функции [DynamicTextSetLock](#) или [DynamicTextTestAndSetLock](#). Если блокировка успешно снята, то функция возвращает значение `TRUE`.

Функция `DynamicTextTestAndSetLock` является атомарной; ее выполнение не может быть прервано другой задачей – в том числе, на многоядерных платформах.

Каждому вернувшему значение `TRUE` вызову функций [DynamicTextSetLock](#) и [DynamicTextTestAndSetLock](#) должен соответствовать свой вызов функции `DynamicTextReleaseLock`.

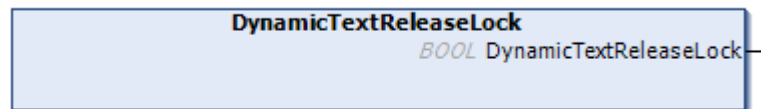


Рис. 6.31. Внешний вид функции `DynamicTextReleaseLock` на языке CFC

7. Библиотека TextListUtils

7.1. Основная информация

Библиотека **TextListUtils** представляет собой обертку над библиотекой [CmpDynamicText](#) с более простым интерфейсом – например, название списка текстов и ID из него передаются напрямую (в **CmpDynamicText** – через указатели), а считанный текст помещается по указателю, переданному пользователем в функцию (в **CmpDynamicText** – пользователь получает указатель на текст, который должен разыменовать самостоятельно).

Далее описываются функции библиотеки (для версии **4.6.0.0**) на примере работы со списком **TextList** из [п. 6.1](#).

Так как большинство функций библиотеки работают с файлами в памяти контроллера – то следует вызывать их событийно, а не циклически. Все функции библиотеки являются синхронными и выполняются в пределах одного цикла задачи контроллера. Пожелание по добавлению асинхронных ФБ для работы со списками текстов зафиксировано в баг-трекере CODESYS (**VSPRT-232**). Ссылка на пример, в котором демонстрируется использование некоторых функций библиотеки (создан в **CODESYS V3.5 SP14 Patch 3**):

[Example WorkingWithTextListFromIecCode_3514v1.projectarchive](#)

7.2. Перечисление ReturnValues

Перечисление **ReturnValues** (тип **DWORD**) содержит коды ошибок, которые могут быть возвращены при вызове функций библиотеки:

- **ERR_OK (0)** – нет ошибок, текст был успешно считан;
- **ERR_FAILED (1)** – операция не удалась, текст не был считан;
- **ERR_PARAMETER (2)** – неверные входные параметры, текст не был считан;
- **ERR_INSUFFICIENT_BUFFER (3)** – недостаточный размер буфер, текст был обрезан при считывании;
- **ERR_LOCKED (4)** – доступ к файлу списка текстов заблокирован вызовом функции [DynamicTextSetLock](#) или [DynamicTextTestAndSetLock](#). Данная ошибка может быть возвращена при вызове функций папки [Non blocking API](#).

7.3. Функция GetText

Функция **GetText** позволяет считать текст по заданному ID из заданного списка текстов для текущего языка проекта (язык может быть установлен, например, с помощью функции [DynamicTextChangeLanguage](#)). ID и название списка текстов передаются в функцию в виде переменных типа **STRING**. Считанное значение типа **STRING** размещается в буфере по указателю **psText** размером **diSize** байт. Функция возвращает код ошибки из перечисления [ReturnValues](#).

Функция может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать функцию [GetTextW](#).

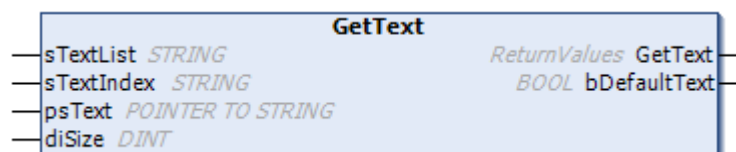


Рис. 7.1. Внешний вид функции **GetText** на языке CFC

В примере ниже считывается текст языка 'en'. Чтобы исключить переключение текстов в визуализации – перед считыванием текста определяется текущий язык проекта (с помощью функции [DynamicTextGetLanguage](#)) и после считывания данный язык снова устанавливается в проекте (с помощью функции [DynamicTextChangeLanguage](#)). Для считывания «дефолтного» текста (из столбца **По умолчанию**) следует установить язык, который отсутствует в проекте. В этом случае выход **bDefaultText**, появившийся в версии библиотеки **4.6.0.0**, будет иметь значение **TRUE**.

```
PROGRAM PLC_PRG
VAR
  xGetText:      BOOL;
  sCurrentLanguage:  STRING(20);
  sTextListName:   STRING := 'TextList';
  sTextId:         STRING := '0';
  sText:           STRING;
END_VAR

IF xGetText THEN

  sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
  CmpDynamicText.DynamicTextChangeLanguage('en');
  TLU.GetText(sTextListName, sTextId, ADR(sText), SIZEOF(sText));
  CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
  xGetText := FALSE;

END_IF
```

Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetText	BOOL	FALSE			
sCurrentLanguage	STRING(20)	"			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
sText	STRING	'sky'			


```
1 IF xGetText FALSE THEN
2
3     sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage ();
4     CmpDynamicText.DynamicTextChangeLanguage ('en');
5     TLU.GetText (sTextListName 'TextList', sTextId '0', ADR (sText 'sky'), SIZEOF (sText 'sky') );
6     CmpDynamicText.DynamicTextChangeLanguage (sCurrentLanguage );
7     xGetText FALSE := FALSE;
8
9 END_IF RETURN
```

Рис. 7.2. Пример использования функции **GetText** на языке ST

7.4. Функция GetTextW

Функция **GetTextW** позволяет считать текст по заданному ID из заданного списка текстов для текущего языка проекта (язык может быть установлен, например, с помощью функции [DynamicTextChangeLanguage](#)). ID и название списка текстов передаются в функцию в виде переменных типа **STRING**. Считанное значение типа **WSTRING** размещается в буфере по указателю **pwsText** размером **diSize** байт. Функция возвращает код ошибки из перечисления [ReturnValues](#).

Функция используется для чтения текстов, которые включают в себя символы Unicode. Для чтения записей в ASCII-кодировке можно использовать функцию [GetText](#) (но также можно использовать и данную функцию). Для корректной работы функции в Менеджере визуализации должна быть установлена галочка **Использовать строки Unicode**.

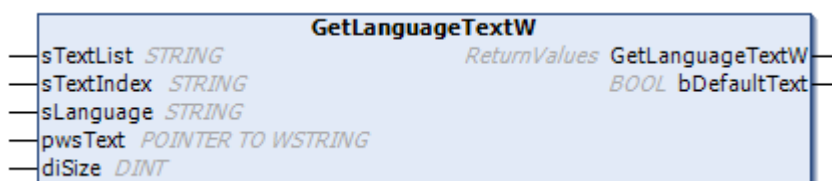


Рис. 7.3. Внешний вид функции **GetTextW** на языке CFC

В примере ниже считывается текст языка 'ru'. Чтобы исключить переключение текстов в визуализации – перед считыванием текста определяется текущий язык проекта (с помощью функции [DynamicTextGetLanguage](#)) и после считывания данный язык снова устанавливается в проекте (с помощью функции [DynamicTextChangeLanguage](#)). Для считывания «дефолтного» текста (из столбца **По умолчанию**) следует установить язык, который отсутствует в проекте. В этом случае выход **bDefaultText**, появившийся в версии библиотеки **4.6.0.0**, будет иметь значение **TRUE**.

```
PROGRAM PLC_PRG
VAR
  xGetTextW:      BOOL;
  sCurrentLanguage:  STRING(20);
  sTextListName:   STRING := 'TextList';
  sTextId:         STRING := '0';
  wsText:         WSTRING;
END_VAR

IF xGetTextW THEN

sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
CmpDynamicText.DynamicTextChangeLanguage('ru');
TLU.GetTextW(sTextListName, sTextId, ADR(wsText), SIZEOF(wsText) );
CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
xGetTextW := FALSE;

END_IF
```

Device.Application.PLC_PRG					
Выражение	Тип	Значение	Подготовленное ...	Адрес	Комментарий
xGetTextW	BOOL	FALSE			
sCurrentLanguage	STRING(20)	"			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
wsText	WSTRING	"небо"			


```
1 IF xGetTextWFALSE THEN
2
3     sCurrentLanguage := CmpDynamicText.DynamicTextGetCurrentLanguage();
4     CmpDynamicText.DynamicTextChangeLanguage('ru');
5     TLU.GetTextW(sTextListName'TextList', sTextId'0', ADR(wsText"небо"), SIZEOF(wsText"небо"));
6     CmpDynamicText.DynamicTextChangeLanguage(sCurrentLanguage);
7     xGetTextWFALSE := FALSE;
8
9 END_IFRETURN
```

Рис. 7.4. Пример использования функции **GetTextW** на языке ST

7.5. Функция GetTextListInfo

Функция **GetTextListInfo** позволяет итерационно обработать файл списка текстов для получения его ID. На вход **stTextListName** подается имя списка текстов. Функция возвращает экземпляр интерфейса **ITextListInfo**, который имеет следующие свойства и методы:

- Свойство **Name** (STRING(255)) – возвращает имя списка текстов;
- Свойство **NumberOfEntries** (UDINT) – возвращает число записей в списке текстов;
- Метод **GetIdIterator** – возвращает экземпляр интерфейса типа **COL.IIterator** из библиотеки **Element Collections**. С помощью методов интерфейса можно производить итерации по записям списка текстов и считывать их ID;
- Метод **Release** – этот метод необходимо вызвать после окончания итераций по списку текстов. После этого экземпляр интерфейса, полученный после вызова метода **GetIdIterator**, больше не может быть использован.

Обратите внимание, что порядок возвращаемых ID при итерациях не определен и может отличаться от их порядка в списке текстов.

Пример использования функции от [разработчиков CODEYS](#) (с незначительными изменениями):

```
PROGRAM PLC_PRG
VAR
  itfTextListInfo:          TLU.ITextListInfo;
  // количество записей в списке текстов
  udiRecordCount:          UDINT;
  iterator:                 COL.IIterator;
  itfElem:                  COL.IElement;
  itfStringElem:           COL.IStringElement;
  // массив ID из списка текстов
  asIdList:                 ARRAY [0..10] OF STRING;
  i:                        INT;
  sTextListName:           STRING := 'TextList';
  xGetIdList:               BOOL;
END_VAR

IF xGetIdList THEN

  i := 0;
  xGetIdList := FALSE;
  itfTextListInfo := TLU.GetTextListInfo(stTextListName:=sTextListName);

  IF itfTextListInfo <> 0 THEN

    udiRecordCount := itfTextListInfo.NumberOfEntries;
    iterator := itfTextListInfo.GetIdIterator();

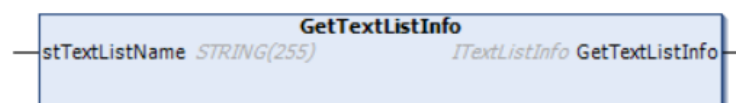
    IF iterator <> 0 THEN
      WHILE iterator.HasNext() DO
        iterator.Next(itfElement=>itfElem);
        _QUERYINTERFACE(itfElem, itfStringElem);
        asIdList[i] := itfStringElem.StringValue;
        i := i + 1;
      END_WHILE
    END_IF
    itfTextListInfo.Release();
  END_IF
END_IF
```

Device.Application.PLC_PRG		
Выражение	Тип	Значение
itfTextListInfo	TLU.ITextListInfo	16#06AB8ADC
udiRecordCount	UDINT	3
iterator	COL.Iterator	16#06AB8C1C
itfElem	COL.IElement	16#06AF6AEC
itfStringElem	COL.IStringElement	16#06AF6AEC
asIdList	ARRAY [0..10] OF S...	
asIdList[0]	STRING	'0'
asIdList[1]	STRING	'1'
asIdList[2]	STRING	'2'
asIdList[3]	STRING	"
asIdList[4]	STRING	"
asIdList[5]	STRING	"
asIdList[6]	STRING	"
asIdList[7]	STRING	"
asIdList[8]	STRING	"
asIdList[9]	STRING	"
asIdList[10]	STRING	"
i	INT	3
sTextListName	STRING	'TextList'
xGetIdList	BOOL	FALSE

```

1 IF xGetIdList FALSE THEN
2
3   i_3 := 0;
4   xGetIdList FALSE := FALSE;
5   itfTextListInfo := TLU.GetTextListInfo(stTextListName:=sTextListName "TextList");
6
7   IF itfTextListInfo <> 0 THEN
8
9     udiRecordCount_3 := itfTextListInfo.NumberOfEntries;
10    iterator := itfTextListInfo.GetIdIterator();
11
12    IF iterator <> 0 THEN
13      WHILE iterator.HasNext() DO
14        iterator.Next(itfElement=>itfElem);
15        _QUERYINTERFACE(itfElem, itfStringElem);
16        asIdList[i_3] := itfStringElem.StringValue;
17        i_3 := i_3 + 1;
18      END_WHILE
19    END_IF
20
21    itfTextListInfo.Release();
22
23  END_IF
24 END_IF RETURN

```

Рис. 7.5. Пример использования функции **GetTextListInfo** на языке STРис. 7.6. Внешний вид функции **GetTextListInfo** на языке CFC

7.6. Комментарий к пунктам 7.7-7.8

Как уже упоминалось в [п. 6.14](#) – при выпуске **CODESYS V3.5 SP20** библиотека **CmpDynamicText** была обновлена до версии **3.5.20.0** и дополнена функциями для работы с языками списков текстов. Вскоре библиотека **TextListUtils** была обновлена до версии **4.5.0.0**; эта версия вошла в состав плагина **CODESYS Visualization Support 4.5.0.0**.

В данной версии были добавлены две новые функции:

- [GetLanguageText](#) (аналог [DynamicTextGetLanguageText](#));
- [GetLanguageTextW](#) (аналог [DynamicTextGetLanguageTextW](#)).

Как и их аналоги – они должны использоваться совместно с функцией [DynamicTextLoadLanguage](#) для загрузки текстов на заданном тексте.

7.7. Функция GetLanguageText

Функция **GetLanguageText** позволяет считать текст по заданному ID из заданного списка текстов для заданного языка проекта. ID, название списка текстов и название языка передаются в функцию в виде переменных типа **STRING**. Считанное значение типа **STRING** размещается в буфере по указателю **psText** размером **diSize** байт. Функция возвращает код ошибки из перечисления [ReturnValues](#).

Перед вызовом функции необходимо загрузить тексты нужного языка в память с помощью функции [DynamicTextLoadLanguage](#). После чтения текста их можно выгрузить (см. [п. 6.18](#) – 6.20).

Функция может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать функцию [GetLanguageTextW](#).

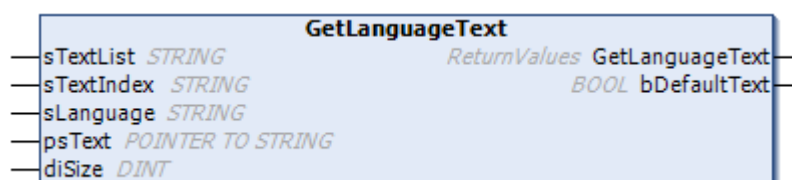


Рис. 7.7. Внешний вид функции **GetLanguageText** на языке CFC

В примере ниже считывается текст языка 'en'. Для считывания «дефолтного» текста (из столбца **По умолчанию**) следует установить язык, который отсутствует в проекте. В этом случае выход **bDefaultText**, появившийся в версии библиотеки **4.6.0.0**, будет иметь значение **TRUE**.

```
PROGRAM PLC_PRG
VAR
  xGetText:      BOOL;

  sLanguage:     STRING := 'en';
  sTextListName: STRING := 'TextList';
  sTextId:       STRING := '0';
  psText:        POINTER TO STRING;

  sText:         STRING;
END_VAR

IF xGetText THEN

  CmpDynamicText.DynamicTextLoadLanguage(sLanguage);
  TLU.GetLanguageText(sTextListName, sTextId, sLanguage, ADR(sText), SIZEOF(sText));
  CmpDynamicText.DynamicTextUnloadLanguage(sLanguage);
  xGetText := FALSE;

END_IF
```

The screenshot displays a variable declaration table and a code snippet in a PLC programming environment. The table lists variables: xGetText (BOOL, FALSE), sLanguage (STRING, 'en'), sTextListName (STRING, 'TextList'), sTextId (STRING, '0'), psText (POINTER TO ST..., 16#00000000), and sText (STRING, 'sky'). Below the table, a code snippet shows an IF statement where xGetText is FALSE. The code includes calls to CmpDynamicText.DynamicTextLoadLanguage, TLU.GetLanguageText, and CmpDynamicText.DynamicTextUnloadLanguage, all with parameters corresponding to the variables in the table above.

Expression	Type	Value	Prepar...	Address	Comm...
xGetText	BOOL	FALSE			
sLanguage	STRING	'en'			
sTextListName	STRING	'TextList'			
sTextId	STRING	'0'			
psText	POINTER TO ST...	16#00000000			
sText	STRING	'sky'			

```
1 IF xGetText[FALSE] THEN
2
3   CmpDynamicText.DynamicTextLoadLanguage(sLanguage['en']);
4   TLU.GetLanguageText(sTextListName['TextList'], sTextId['0'], sLanguage['en'], ADR(sText['sky']), SIZEOF(sText['sky']));
5   CmpDynamicText.DynamicTextUnloadLanguage(sLanguage['en']);
6   xGetText[FALSE] := FALSE;
7
8 END_IF
```

Рис. 7.8. Пример использования функции **GetLanguageText** на языке ST

7.8. Функция GetLanguageTextW

Функция **GetLanguageTextW** позволяет считать текст по заданному ID из заданного списка текстов для заданного языка проекта. ID, название списка текстов и название языка передаются в функцию в виде переменных типа **STRING**. Считанное значение типа **WSTRING** размещается в буфере по указателю **pwsText** размером **diSize** байт. Функция возвращает код ошибки из перечисления [ReturnValues](#).

Перед вызовом функции необходимо загрузить тексты нужного языка в память с помощью функции [DynamicTextLoadLanguage](#). После чтения текста их можно выгрузить (см. [п. 6.18](#) – 6.20).

Функция используется для чтения текстов, которые включают в себя символы Unicode. Для чтения записей в ASCII-кодировке можно использовать функцию [GetLanguageText](#) (но также можно использовать и данную функцию). Для корректной работы функции в Менеджере визуализации должна быть установлена галочка **Использовать строки Unicode**.

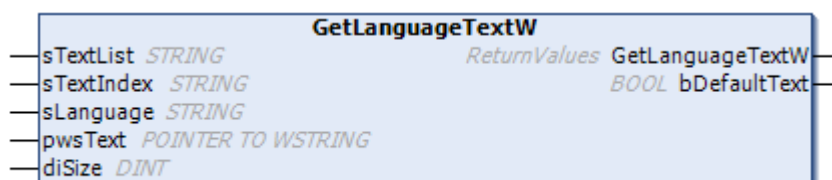


Рис. 7.9. Внешний вид функции **GetLanguageTextW** на языке CFC

В примере ниже считывается текст языка 'ru'. Для считывания «дефолтного» текста (из столбца **По умолчанию**) следует установить язык, который отсутствует в проекте. В этом случае выход **bDefaultText**, появившийся в версии библиотеки **4.6.0.0**, будет иметь значение **TRUE**.

```
PROGRAM PLC_PRG
VAR
  xGetText:          BOOL;

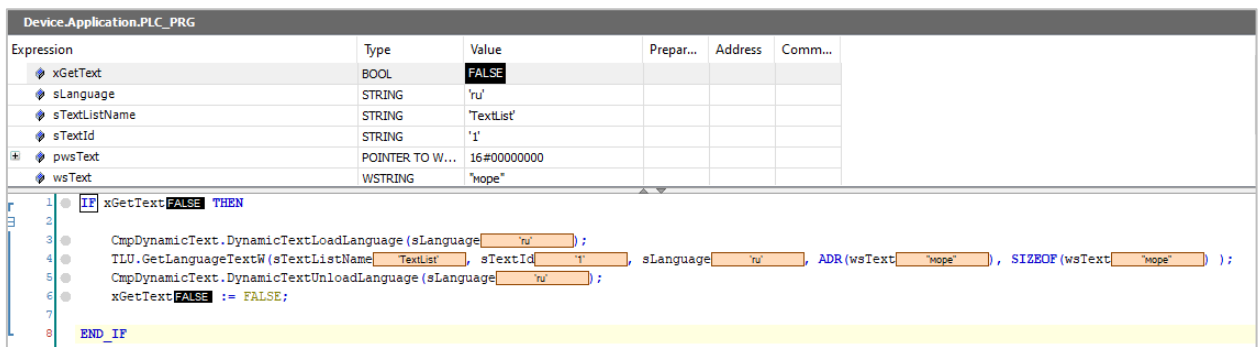
  sLanguage:         STRING := 'ru';
  sTextListName:     STRING := 'TextList';
  sTextId:           STRING := '1';
  pwsText:           POINTER TO WSTRING;

  wsText:           WSTRING;
END_VAR

IF xGetText THEN

  CmpDynamicText.DynamicTextLoadLanguage(sLanguage);
  TLU.GetLanguageTextW(sTextListName, sTextId, sLanguage, ADR(wsText), SIZEOF(wsText));
  CmpDynamicText.DynamicTextUnloadLanguage(sLanguage);
  xGetText := FALSE;

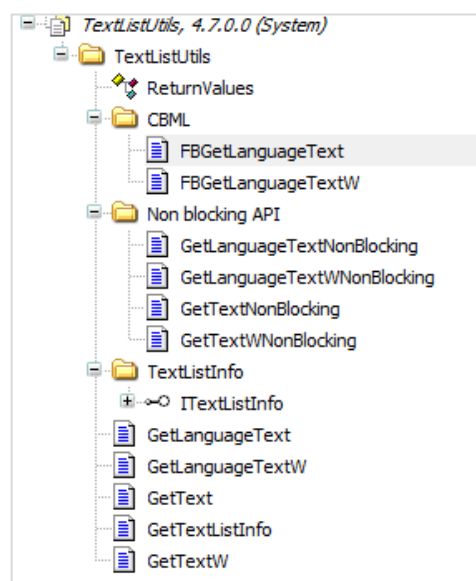
END_IF
```

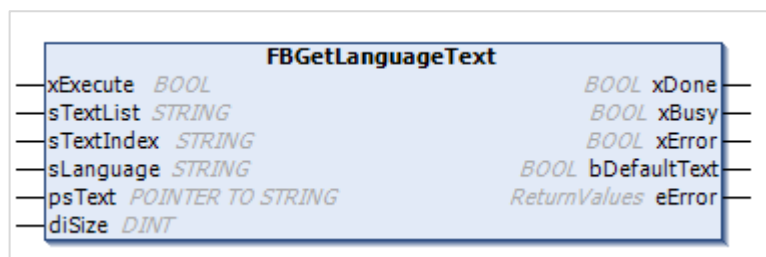
Рис. 7.10. Пример использования функции **GetLanguageTextW** на языке ST

7.9. Асинхронные ФБ и Non blocking API

В декабре 2025 года состоялся выход плагина **CODESYS Visualization Support 4.7.0.0**, в рамках которого библиотека **TextListUtils** была обновлена до версии **4.7.0.0**. В этой версии добавились две новые папки:

- **CMBL** – содержит функциональные блоки **FBGetLanguageText** и **FBGetLanguageTextW**. Они используются для той же цели, что и функции [GetLanguageText](#) и [GetLanguageTextW](#), но в отличие от них – выполняются асинхронно, не блокируя поток управления задачи;
- **Non blocking API** – содержит функции **GetTextNonBlocking**, **GetTextWNonBlocking**, **GetLanguageTextNonBlocking**, **GetLanguageTextWNonBlocking**. От таких же функций без постфикса **Blocking** они отличаются только тем, что если файл списка текстов, к которому происходит обращение, заблокирован вызовом функции [DynamicTextSetLock](#) или [DynamicTextTestAndSetLock](#), то эти функции не блокируют поток управления до **освобождения файла**, а возвращают **ошибку ERR_LOCKED**. Соответственно, такие функции должны вызываться до тех пор, пока не вернут **ERR_OK**.

Рис. 7.11. Список объектов библиотеки **TextListUtils** версии **4.7.0.0**

Рис. 7.11. Внешний вид ФБ **FBGetLanguageText** на языке CFC

ФБ **FBGetLanguageText** позволяет считать текст по заданному ID из заданного списка текстов для заданного языка проекта. ID, название списка текстов и название языка передаются в виде переменных типа **STRING**. Считанное значение типа **STRING** размещается в буфере по указателю **psText** размером **diSize** байт.

Выполнение ФБ начинается по переднему фронту входа **xExecute**. В процессе работы выход **xBusy** имеет значение **TRUE**. Если ФБ успешно завершил работу, то выход **xDone** принимает значение **TRUE**. Если в процессе выполнения ФБ произошла ошибка, то выход **xError** принимает значение **TRUE**, а на выходе **eError** отображается код ошибки из перечисления [ReturnValues](#).

Для считывания «дефолтного» текста (из столбца **По умолчанию**) следует установить язык, который отсутствует в проекте. В этом случае выход **bDefaultText** будет иметь значение **TRUE**.

Сброс выходов блока происходит по заднему фронту входа **xExecute**.

Перед вызовом ФБ необходимо загрузить тексты нужного языка в память с помощью функции [DynamicTextLoadLanguage](#). После чтения текста их можно выгрузить (см. [п. 6.18](#) – 6.20).

ФБ может использоваться только для чтения текстов, которые включают в себя исключительно символы ASCII-кодировки. Для чтения текстов, содержащих символы Unicode, следует использовать ФБ **FBGetLanguageTextW**.

```
PROGRAM PLC_PRG
VAR
  xGetText:          BOOL;
  fbTrig:            R_TRIG;
  fbGetText:         TLU.FBGetLanguageText;

  sLanguage:         STRING := 'en';
  sTextListName:     STRING := 'TextList';
  sTextId:           STRING := '0';
  psText:            POINTER TO STRING;

  sText:             STRING;
END_VAR

fbTrig(CLK := xGetText);

IF fbTrig.Q THEN
  CmpDynamicText.DynamicTextLoadLanguage(sLanguage);
END_IF
```

```

fbGetText
(
  xExecute := fbTrig.Q,
  sTextList := sTextListName,
  sTextIndex := sTextId,
  sLanguage := sLanguage,
  psText := ADR(sText),
  diSize := SIZEOF(sText)
);

IF fbGetText.xDone OR fbGetText.xError THEN
  CmpDynamicText.DynamicTextUnloadLanguage(sLanguage);
END_IF

```

Device.Application.PLC_PRG			
Expression	Type	Value	Pr
xGetText	BOOL	TRUE	
fbTrig	R_TRIG		
fbGetText	TLU.FBGetLanguage...		
sLanguage	STRING	'en'	
sTextListName	STRING	'TextList'	
sTextId	STRING	'0'	
psText	POINTER TO STRING	16#00000000	
sText	STRING	'sky'	


```

1 ● fbTrig(CLK TRUE := xGetText TRUE);
2
3 ● IF fbTrig.Q FALSE THEN
4 ●   CmpDynamicText.DynamicTextLoadLanguage(sLanguage 'en');
5 END_IF
6
7 ● fbGetText
8 (
9   xExecute FALSE := fbTrig.Q FALSE,
10  sTextList 'TextList' := sTextListName 'TextList',
11  sTextIndex '0' := sTextId '0',
12  sLanguage 'en' := sLanguage 'en',
13  psText 16#07B10A33 := ADR(sText 'sky'),
14  diSize 81 := SIZEOF(sText 'sky')
15 );
16
17 ● IF fbGetText.xDone FALSE OR fbGetText.xError FALSE THEN
18 ●   CmpDynamicText.DynamicTextUnloadLanguage(sLanguage 'en');
19 ● END_IF RETURN

```

Рис. 7.12. Пример использования ФБ **FBGetLanguageText** на языке ST

Заклучение

По результатам написания данного документа были сформулированы два пожелания к библиотекам для работы со списками текстов, которые теперь зафиксированы в баг-трекере CODESYS:

- **CDS-71364:** Visu, DynamicTextChangeLanguage: It should be possible to read a entry from textlist without switching the visu language;
- **CDS-71365:** Visu, CmpDynamicText, TextListUtils: Function to rewrite strings in a textlist are missing.

Первое пожелание касается добавления в функции [GetText](#) и [GetTextW](#) входа, который будет определять язык для считываемой записи списка текстов. Это позволит избежать вызова функции [DynamicTextChangeLanguage](#). Также в рамках данного пожелания указано, что текущее поведение функции [DynamicTextChangeLanguage](#) является некорректным и непрозрачным для пользователя, так как она переключает не только язык списков текстов, используемый другими функциями библиотеки, но и язык визуализации. Данное пожелание было реализовано:

- [в версии 3.5.20.0 библиотеки CmpDynamicText](#);
- [в версии 4.5.0.0 библиотеки TextListUtils](#).

Второе пожелание касается добавления в библиотеки функций, которые позволят редактировать списки текстов из кода программы (например, это может быть полезным в случае необходимости изменения названий рецептов, логинов операторов и т. д.). Данное пожелание до сих пор не реализовано.

CODESYS V3 / CDS-71364

Visu, DynamicTextChangeLanguage: It should be possible to read a entry from textlist without switching the visu language

Agile Board More ▾

Details

Type:	↑ Improvement	Status:	OPEN (View Workflow)
Affects Version/s:	None	Resolution:	Unresolved
Component/s:	None	Fix Version/s:	Not Planned
Labels:	None		
Target User Group:	End User		

Description

To get an entry from a text list in a certain language it is necessary to select it first via "DynamicTextChangeLanguage". This function switch the language to for visu too. It's unexpected behavior for user. A possible solution could be a new "TextListUtils.GetText2" with the language as input.

Activity

All [Comments](#)

There are no comments yet on this issue.

People

Assignee:

Reporter:

Votes:

Watchers:

Dates

Created:


Updated:

Resolved:

Agile

View on Board


Рис. Z1. Текст заявки CDS-71364

 CODESYS V3 / CDS-71365

Visu, CmpDynamicText, TextListUtils: Function to rewrite strings in a textlist are missing

Agile Board More ▾

Details

Type:	 Improvement	Status:	OPEN (View Workflow)
Affects Version/s:	None	Resolution:	Unresolved
Component/s:	None	Fix Version/s:	Not Planned
Labels:	None		
Target User Group:	End User		

Description

If customer want to change entry of a textlist, a rewrite function like " DynamicTextSetText, DynamicTextSetTextW, DynamicTextSetDefalutText, DynamicTextSetDefaultTextW, SetText, SetTextW" are missing.

WORKAROUND:
It is possible to edit the textfile, copy them to the PLC and load the new file dynamically via "DynamicTextReloadTexts"

Activity

[All](#) [Comments](#)

There are no comments yet on this issue.

People

Assignee:

Reporter:

Votes:

Watchers:

Dates

Created:

Updated:

Resolved:

Agile

[View on Board](#)

Рис. Z2. Текст заявки CDS-71365