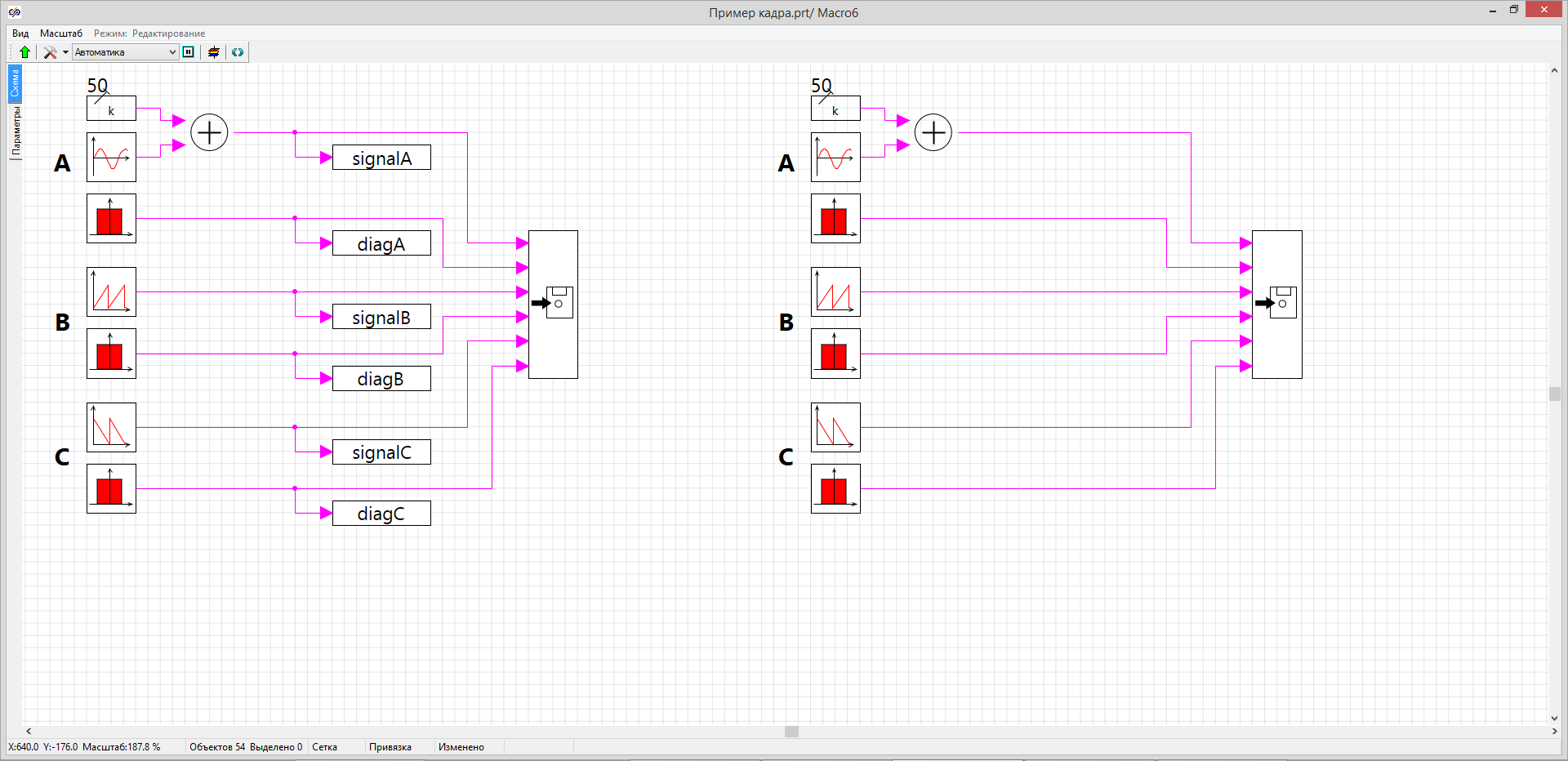
**Пример создания простого видеокадра**

В примере дается общее представление о процессе создания видеокадров в SimInTech.

**Модель**

Дана упрощенная модель, имитирующая некий измерительный комплекс, регистрирующий сигналы A, B и C.



**Модель измерительной системы**

Диапазон измерения всех трех сигналов – от 0 до 100 единиц.

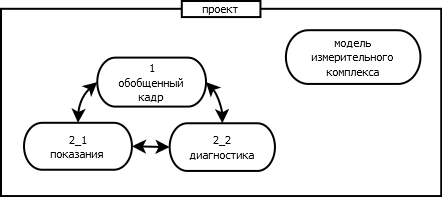
В качестве имитатора неисправностей для каждого сигнала дополнительно вводится сигнал генерации случайных величин по закону равномерного распределения с диапазоном значений от 0 до 1,2 единиц. Фактом неисправности измерительного канала должно считаться превышение этими сигналами порога в 1 единицу.

**Задача**

Задача состоит в создании набора из трех видеокадров: кадра верхнего уровня с обобщенной сигнализацией и двух детальных кадров. Один детальный кадр с отображением измеряемых величин, другой – с диагностикой измерительных каналов.

Кадры должны быть организованы на основе блоков «Субмодель» и должны размещаться в одном проекте с моделью, однако пользователь кадров не должен иметь возможности переключиться куда-либо внутри проекта кроме трех указанных выше кадров.

Предполагаемая структура проекта выглядит следующим образом.



**Структура организации видеокадров в проекте**

Кадр верхнего уровня должен содержать обобщенную сигнализацию о превышении измеряемыми величинами порогов и обобщенной диагностики в виде двух световых табло: «Измерения» и «Диагностика».

В норме оба табло должны иметь зеленый фон.

Увеличение любых двух из трех измеряемых показаний более 60 единиц должно приводить к появлению предупредительной сигнализации – желтое табло «Измерения».

Увеличение любых двух из трех показаний более 80 единиц должно приводить к появлению аварийной сигнализации – красное табло «Измерения».

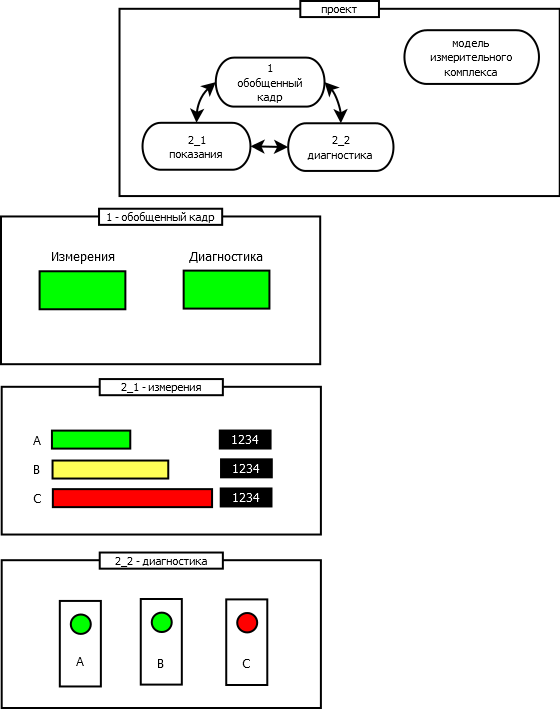
Появление сигнала неисправности в любом из трех каналов должно приводить к появлению аварийной диагностической сигнализации – красное табло «Диагностика».

Детальный кадр с показаниями должен отображать все три измеряемых величины в виде аналоговых шкал и в виде цифровых значений. При превышении порогов в 60 и 80 единиц каждым сигналом соответсвтующая шкала должна изменять свой цвет с зеленого на желтый и на красный соответственно. При возникновении сигнала неисправности в каком-либо измерительном канале показания по данному каналу не должны отображаться.

Также на кадре должна присутствовать обобщенная сигнализация по превышению порогов, аналогичная кадру верхнего уровня.

На кадре диагностики должны отображаться три световых индикатора, по одному на каждый измерительный канал. При значениях сигнала неисправности >=1 индикатор соответствующего канала должен менять цвет с зеленого на красный.

Ниже представлены эскизы видеокадров.



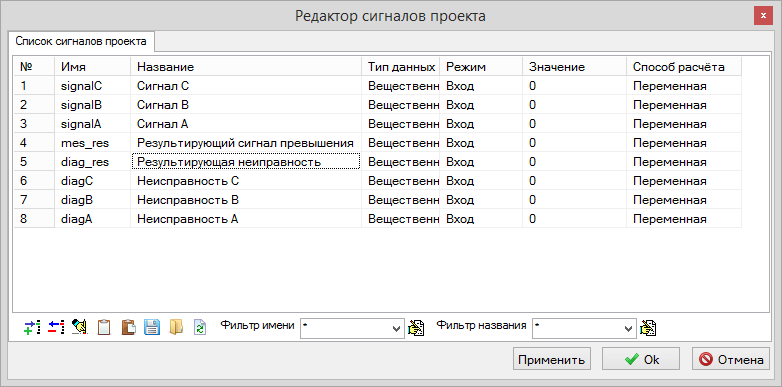
**Эскизы кадров**

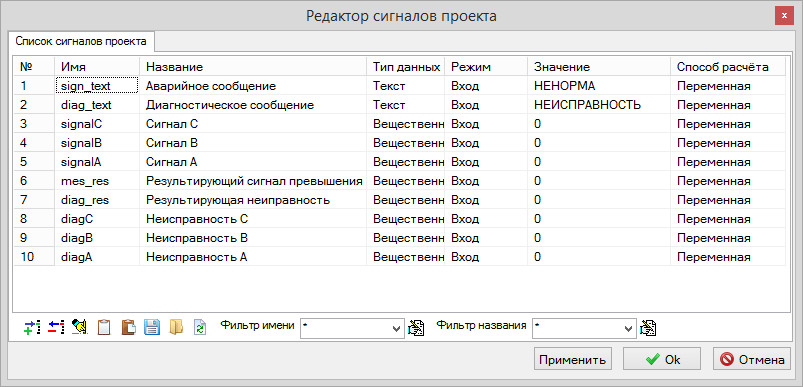
**Создание набора сигналов**

Для того, чтобы иметь возможность визуализировать значения, генерируемые в расчетной модели, нам нужно создать под хранение этих значений необходимое количество сигналов проекта. В случае, когда для записи, хранения и передачи расчетных величин используется база данных, и требуемые сигналы уже присутствуют в этой базе, данный этап может быть пропущен. В остальном никаких отличий между этими двумя случаями нет.

Нам будет необходимо иметьсигналы для аналоговых значений, регистрируемых по каналам A, B и C, а также сопутствующие им сигналы, имитирующие сигнализацию неисправности измеристельного канала. Также нам понадобятся два обобщенных сигнала: один с информацией о превышении уставок для табло «Сигнализация» и один с информацией о наличии неисправноти для табло «Неисправность». Сигналы проекта создаются и конфигурируются в окне редактора сигналов, вызываемом через меню в Главном Окне: **«Сервис» → «Сигналы…»**.

Перечень необходимых сигналов и их параметры приведены на рисунке ниже.

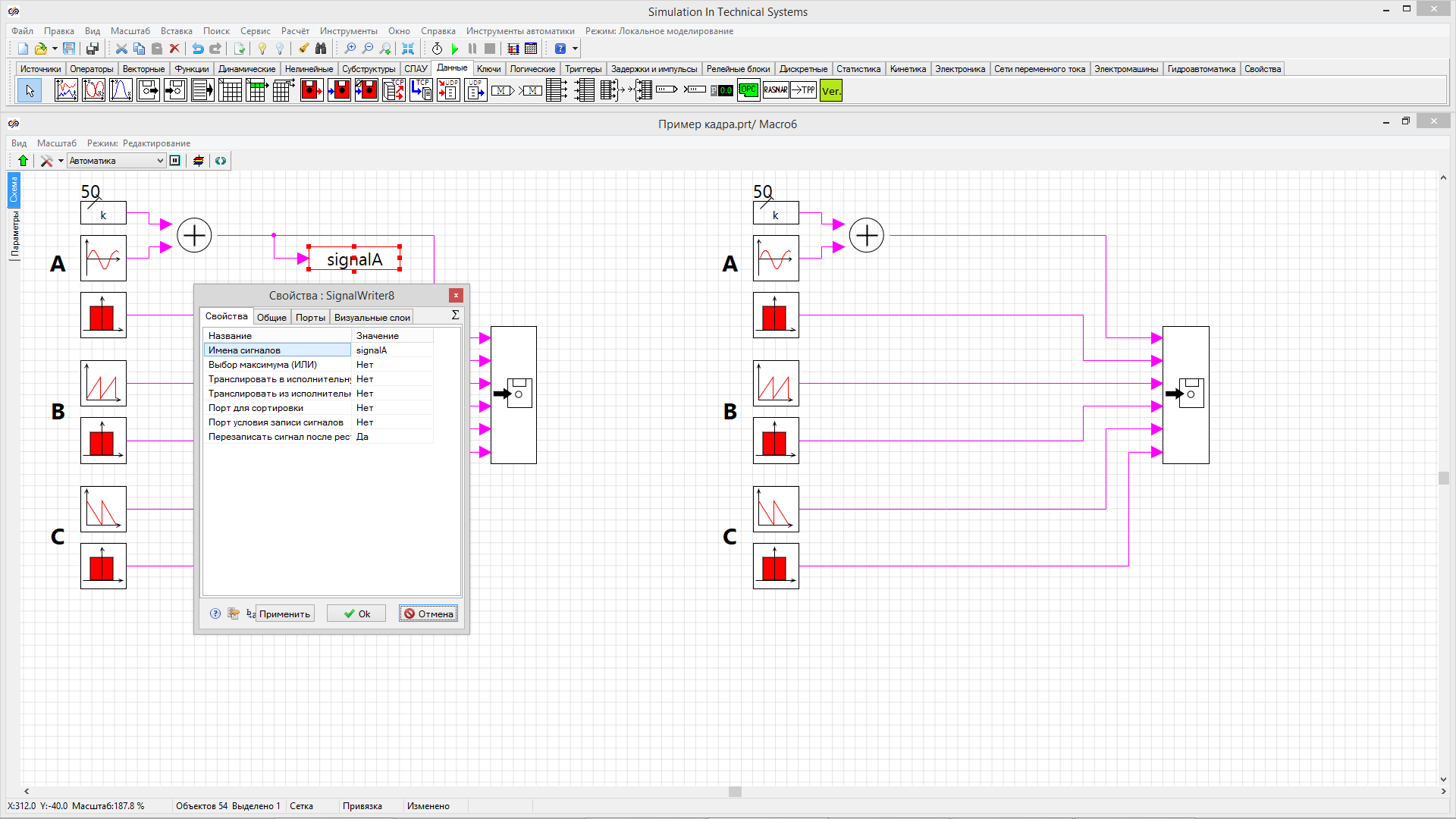




**Сигналы проекта**

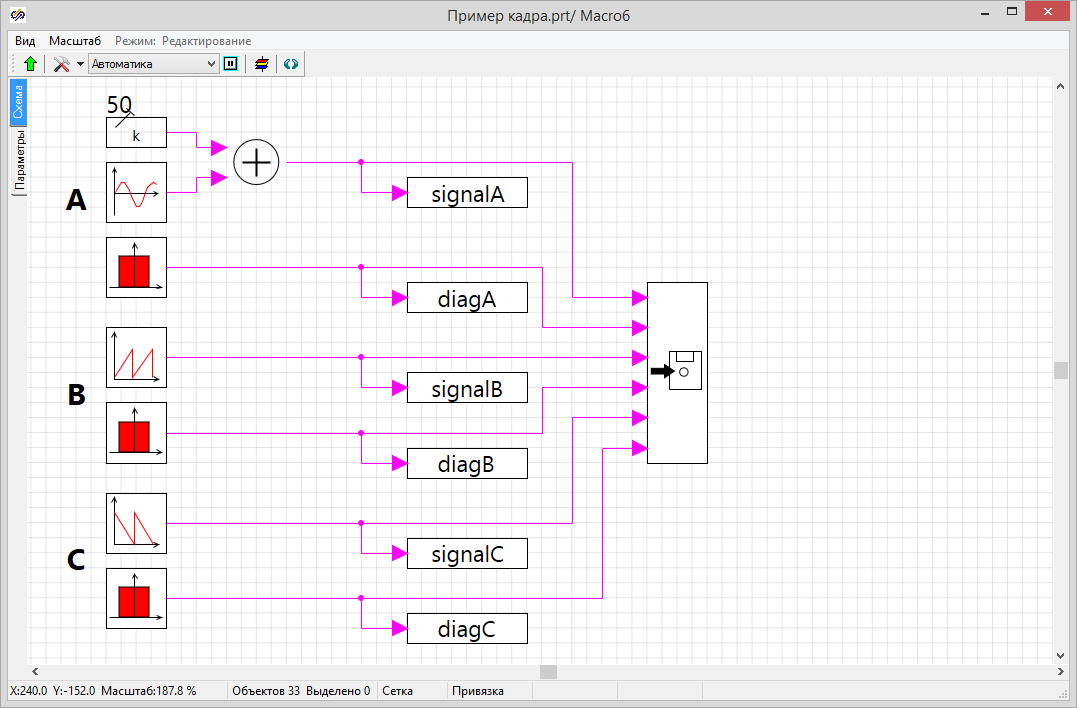
Теперь нужно дополнить блок-схему модели блоками записи значений в сигналы проекта. Для этого используются блоки  «Запись в список сигналов» из вкладки «Данные» в главном окне проекта.

Для указания сигнала, в который должно записываться, подключаемое к блоку значение, имя этого сигнала нужно вписать с поле свойства «Имена сигналов» (вкладка «Свойства») данного блока. Окно свойств блока вызывается по двойному клику ЛКМ на этом блоке.



**Пример указания имени сигнала для записи расчетных значений с помощью блока «Запись в блок сигналов»**

В результате наша схема должна выгдядеть так.



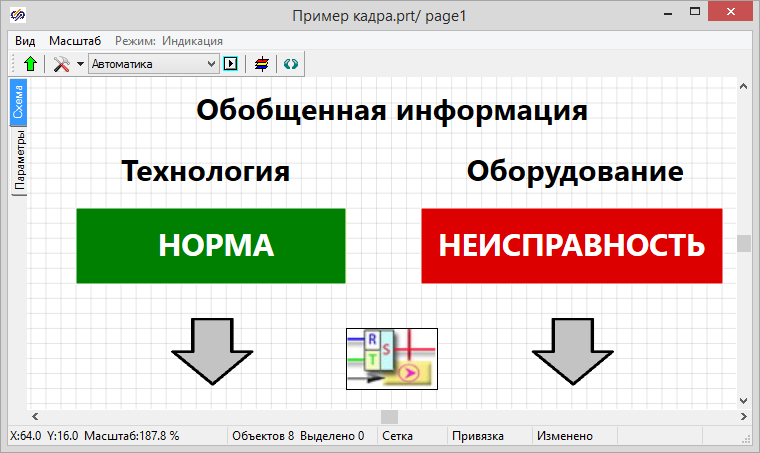
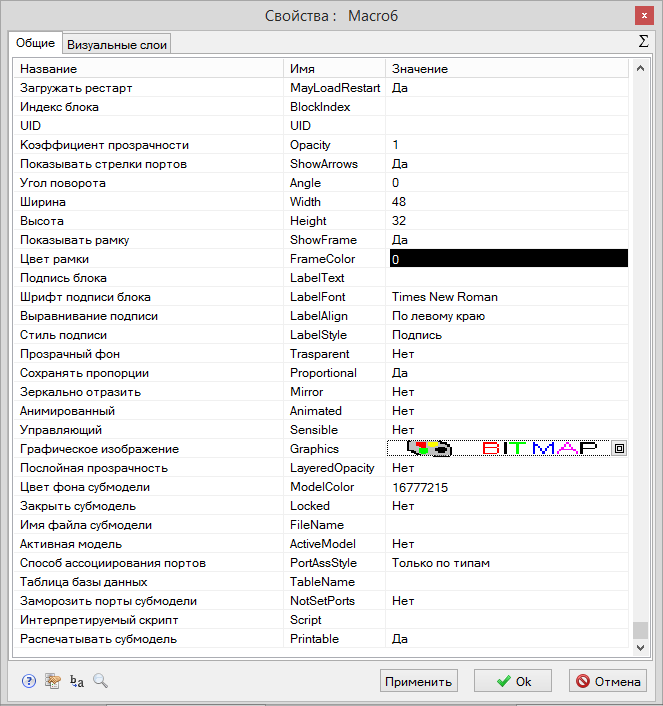
**Расчетная модель проекта с блоками для записи расчетных значений в сигналы проекта**

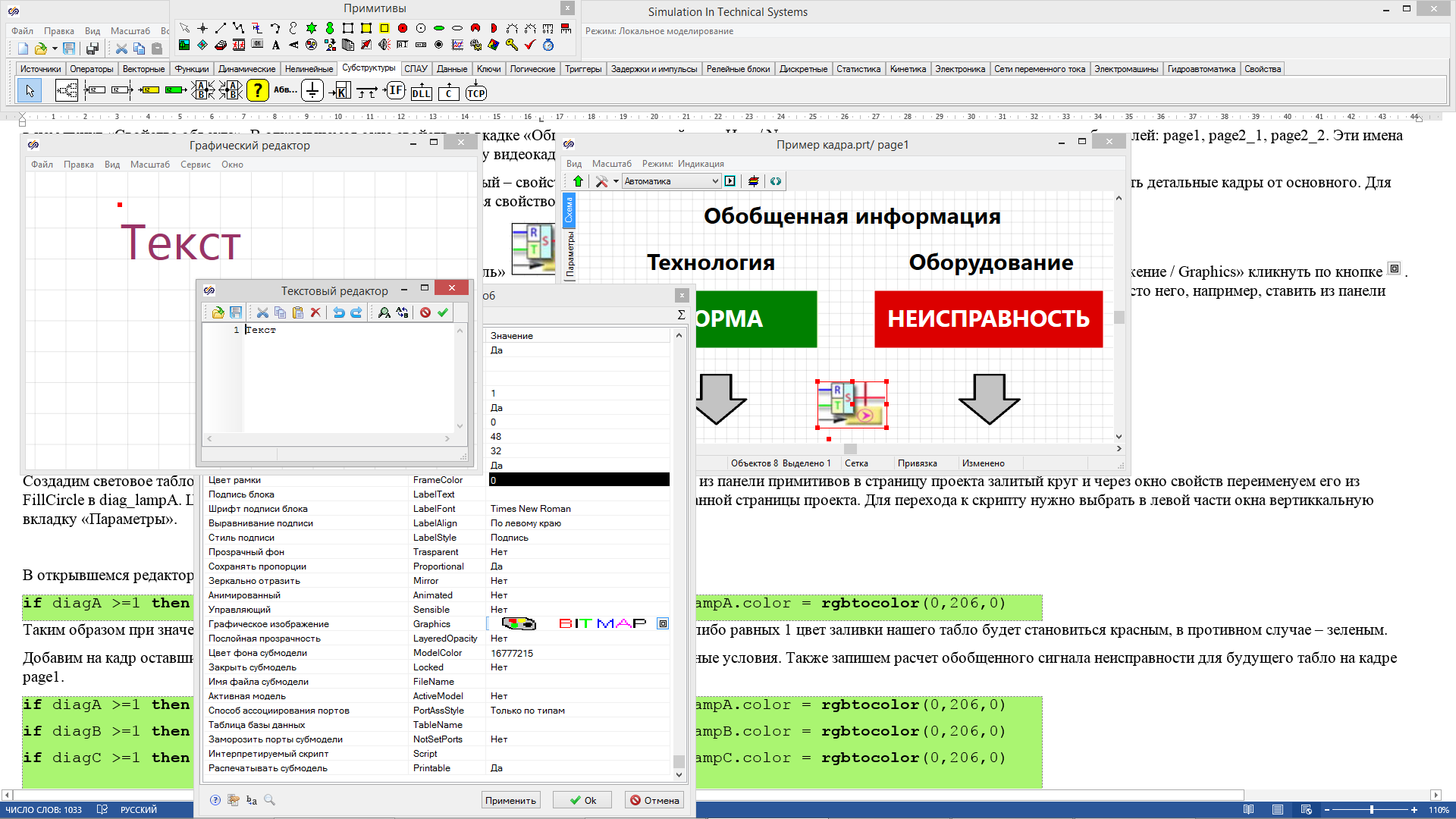
**Создание структуры видеокадров**

Добавим в схемное окно редактируемого проекта четыре блока «Субмодель». В один из этих блоков нужно перенести расчетную модель. Это позволит лучше упорядочить содержимое нашего проекта и никак не повлияет на процесс расчета и доступность рассчитываемых значений и сигналов проекта. Для переноса модели достаточно выделить все ее элементы, включая линии связи, затем вырезать их из основного окна проекта, открыть блок «Субмодель» по двойному клику ЛКМ на нем и вставить содержимое буфера обмена в открывшуюся страницу субмодели.

Затем изменим имена оставшихся трех блоков «Субмодель», которые будут выполнять роль видеокадров. Для этого нужно выделить один из блоков, по клику ПКМ вызвать контекстное меню и выбрать в нем пункт «Свойства объекта». В открывшемся окне свойств, на вкадке «Общие», в строке свойства «Имя / Name» нужно вписать следующие имена для субмоделей: page1, page2\_1, page2\_2. Эти имена будут использоваться, например, при организации переходов между видеокадрами.

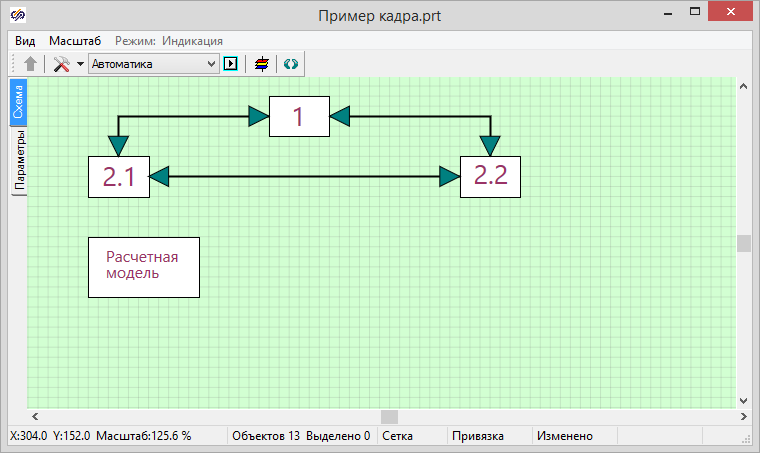
Также изменим фон окна для субмоделей 2\_1 и 2\_2 с белого на серый – свойство «Цвет фона субмодели / ModelColor», чтобы пользователю было проще отличить детальные кадры от основного. Для изменения фона основной страницы проекта можно воспользоваться свойством «Цвет фона…» в меню «Вид» схемного окна проекта.

Кроме того, можно изменить внешний вид самих блоков «Субмодель»  на основной странице. Для этого нужно в поле свойства «графическое изображение / Graphics» кликнуть по кнопке . Откроется окно графического редактора со вставленным по умолчанию растровым изображением для блока «Субмодель». Его можно выделить и удалить. А вместо него, например, ставить из панели примитивов текст с кратким названием блока.



**Редактирование изображения для блока «Субмодель»**

При желании можно дополнить получившееся изображение в схемном окне проекта поясняющими статическими надписями и рисунками.

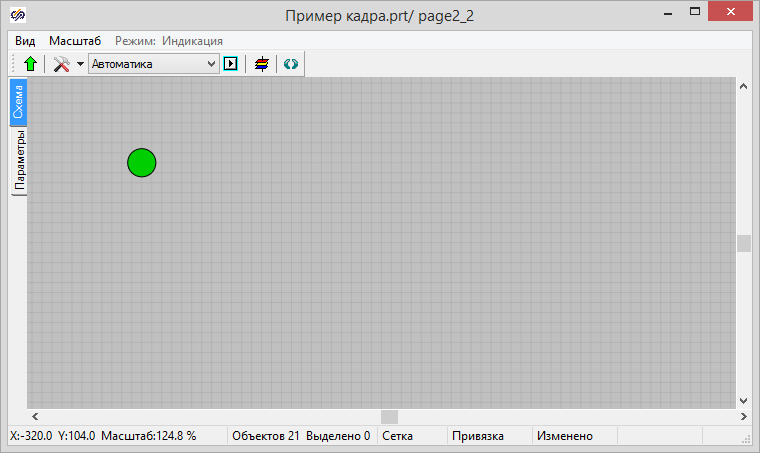


**Пример оформления главной страницы проекта**

**Кадр диагностики**

Откроем блок «page2\_2». Открывшаяся страница является будущим кадром диагностики.

Создадим световое табло сигнализации для измерительного канала «А». В качестве табло добавим из панели примитивов в страницу проекта залитый круг и через окно свойств переименуем его из FillCircle в diag\_lampA. Цвет по умолчанию – зеленый. Дальнейшуюю логику опишем в скрипте данной страницы проекта. Для перехода к скрипту нужно выбрать в левой части окна вертиккальную вкладку «Параметры».



**Световое табло для сигнализации о неисправности измерительного канала A**

В открывшемся редакторе нужно написать следуюший скрипт.

|  |
| --- |
| **if** diagA >=1 **then** diag\_lampA.color = **rgbtocolor**(240,0,0) **else** diag\_lampA.color = **rgbtocolor**(0,206,0) |

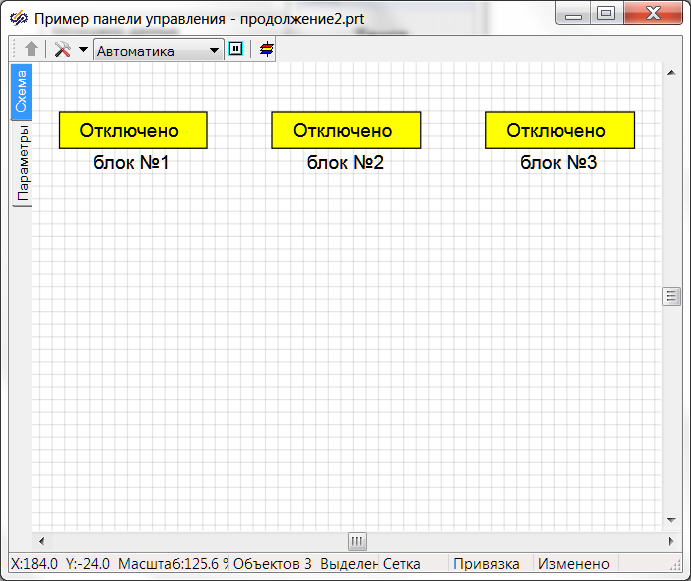
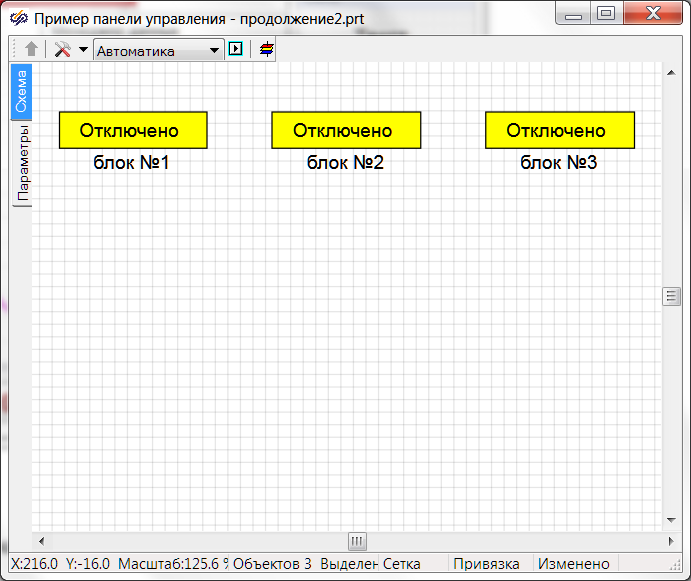
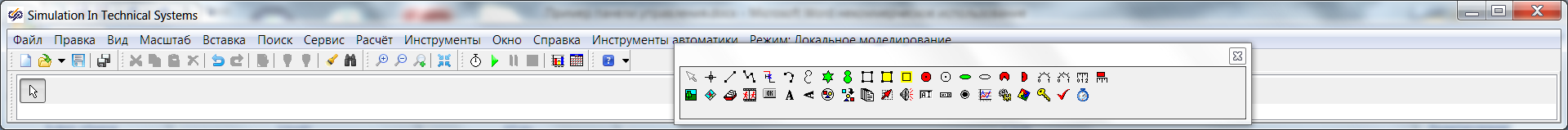
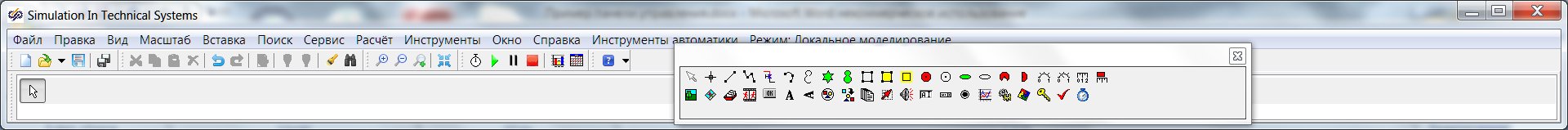
Таким образом, при значениях случайно изменяющегося диагностического сигнала diagA больше, либо равных 1 цвет заливки нашего табло будет становиться красным, в противном случае – зеленым.

Добавим на кадр оставшиеся табло для измерительных каналов B и С и запишем для них аналогичные условия.

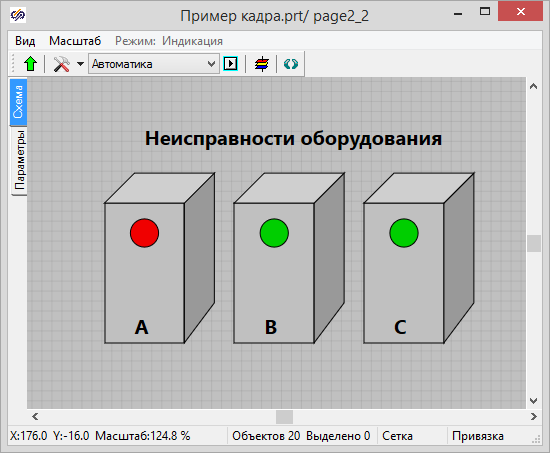
|  |
| --- |
| **if** diagA >=1 **then** diag\_lampA.color = **rgbtocolor**(240,0,0) **else** diag\_lampA.color = **rgbtocolor**(0,206,0)  **if** diagB >=1 **then** diag\_lampB.color = **rgbtocolor**(240,0,0) **else** diag\_lampB.color = **rgbtocolor**(0,206,0)  **if** diagC >=1 **then** diag\_lampC.color = **rgbtocolor**(240,0,0) **else** diag\_lampC.color = **rgbtocolor**(0,206,0) |

Также запишем расчет обобщенного сигнала неисправности для будущего табло обобщенной диагностики на кадре page1.

|  |
| --- |
| **if** (diagA>=1) **OR** (diagB>=1) **OR** (diagC>=1) **then** diag\_res = 2 **else** diag\_res = 0; |

Проверить работу табло можно перейдя обратно во вкладку «Схема» и запустив расчет. Для этого переведем схемное окно из режима «Редактирование» в режим «Индикация» по кнопке  / на панели инструментов схемного окна и запустим расчет проекта по горячей клавише «F9» или по кнопке  на панели инструментов главного окна. Для продолжения работы расчет нужно остановить сочетанием горячих клавиш Shift+F9 или по кнопке на панели инструментов главного окна.

Теперь можно дополнить данный кадр текстом с названием и некоторыми статическими изображениями.

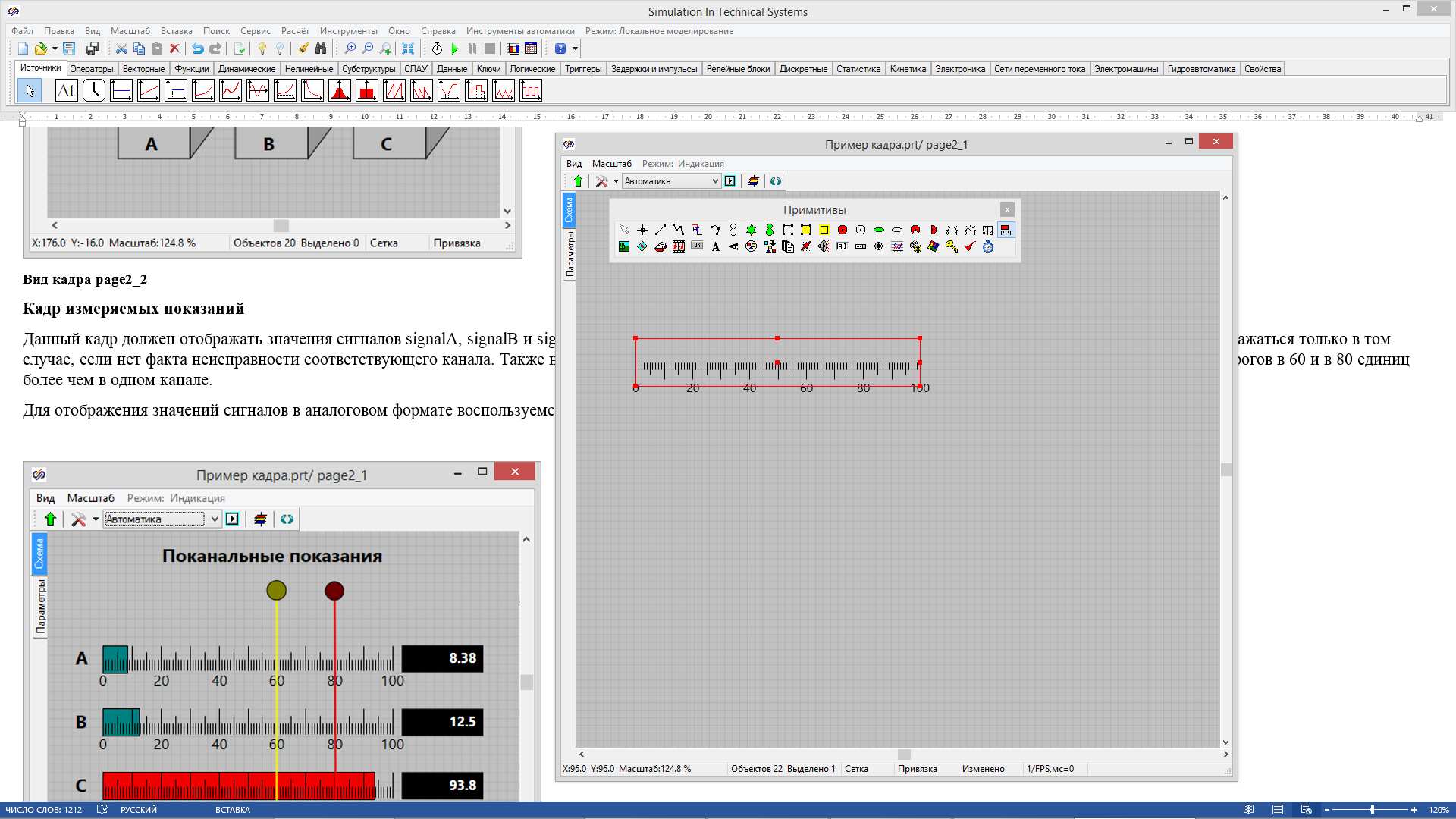


**Вид кадра page2\_2**

**Кадр измеряемых показаний**

Данный кадр должен отображать значения сигналов signalA, signalB и signalC в аналоговом и цифровом представлениях. Кроме того, значения сигналов должны отображаться только в том случае, если нет факта неисправности соответствующего канала. Также на этом кадре должна отображаться обобщенная сигнализация об одновременном переходе порогов в 60 и в 80 единиц более чем в одном канале.

Для отображения значений сигналов в аналоговом формате воспользуемся готовым шкальным примитивом «Линейный прибор» из панели примитивов.

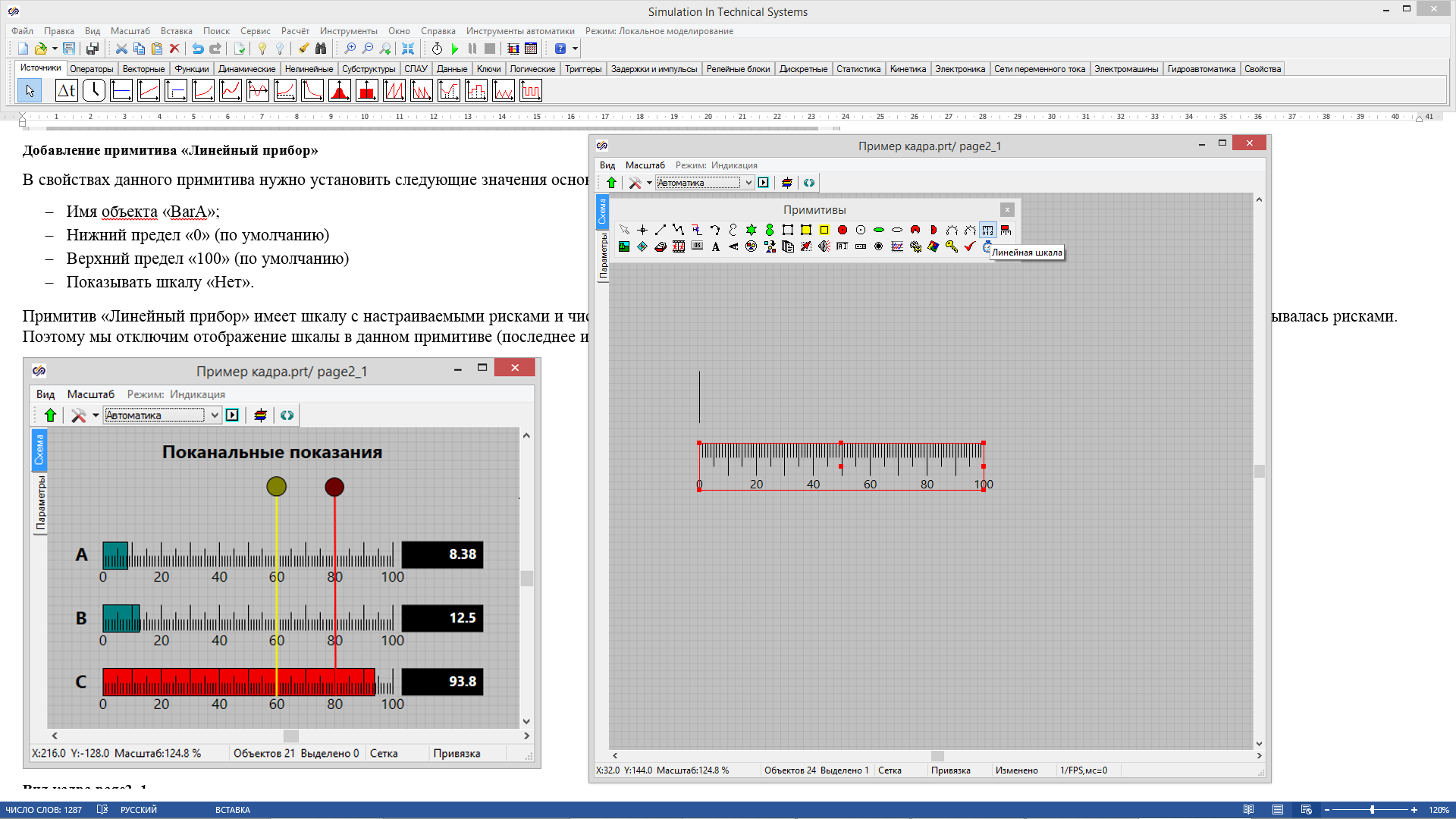


**Добавление примитива «Линейный прибор»**

В свойствах данного примитива нужно установить следующие значения основных свойств:

* Имя объекта «BarA»;
* Нижний предел «0» (по умолчанию);
* Верхний предел «100» (по умолчанию);
* Показывать шкалу «Нет».

Примитив «Линейный прибор» имеет шкалу с настраиваемыми рисками и числовыми подписями, но его незльзя настроить таким образом, чтобы шкала полностью перекрывалась рисками. Поэтому мы отключим отображение шкалы в данном примитиве (последнее изменявшееся выше свойство) и возьмем шкалу от другого примитива «Линейная шкала».

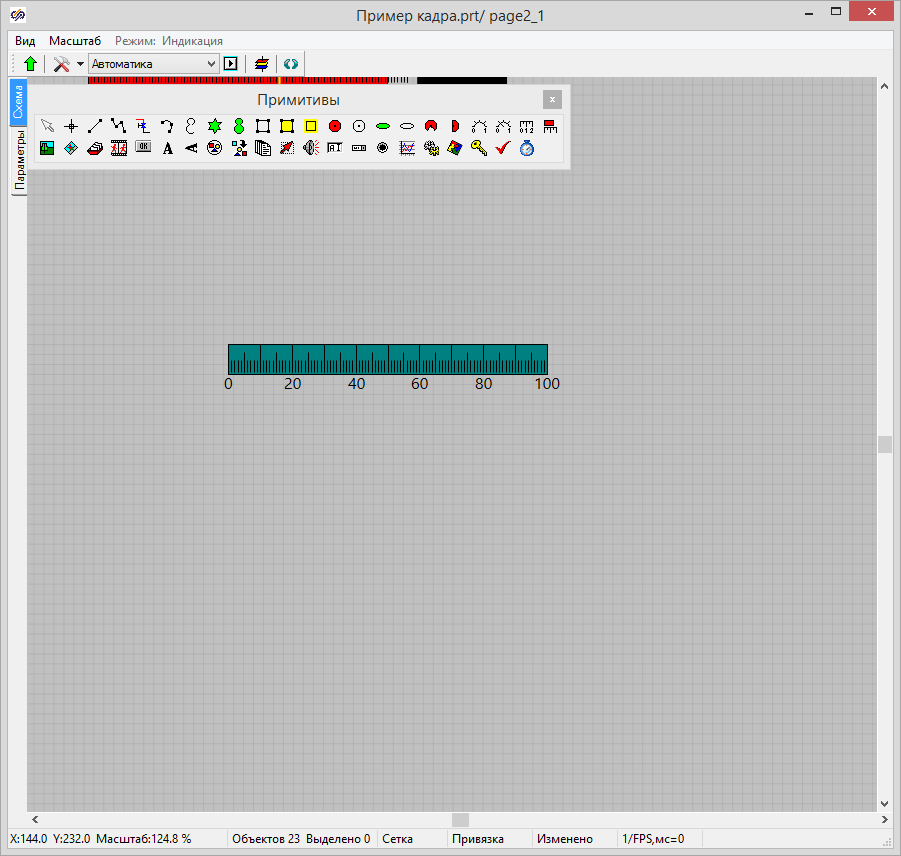


**Добавление примитива «Линейная шкала»**

Основные настройки для данного примитива:

* Штрихи наоборот «Да»;
* Нижний предел «0» (по умолчанию);
* Верхний предел «100» (по умолчанию);
* Координаты точек – см. ниже.

Для того, чтобы значения, отображаемые линейным прибором правильно согласовались на присоединенной шкале нужно совместить два примитива так, чтобы совпадали их левая и правая границы. Этого можно достичь копированием содержимого свойств «Координаты точек» и «Ширина» из одного примитива в другой либо простыми манипуляциями мышью в схемном окне проекта с включенной привязкой к сетке (меню схемного окна: **«Вид→Привязка»**). Для отладки можно задать в свойстве «Значение / Value» линейного прибора какое-нибудь значение.



**Пример совмещения шкалы и линейного прибора с отключенной шкалой**

Теперь нужно связать сигнал signalA с линейным прибором. Это можно сделать различными способами. В данном случае мы воспользуемся записью в скрипте данной страницы проекта.

|  |
| --- |
| barA.value = signalA; |

Работу кадра, как и ранее, можно проверить запустив расчет.

Для учета неисправности канала придется ввести промежуточную переменную signalAd, которая будет равна значению сигнала при отсутствии неисправности и «-1» при ее наличии. Таким образом не будет отображаться шкала и результирующее значение не будет вносить вклад при оценке количества сигналов, превысивших пороги.

|  |
| --- |
| **if** diagA >=1 **then** signalAd=-1 **else** signalAd = signalA;  barA.value = signalAd; |

Также мы должны закрашивать шкалу различными цветами (свойство barcolor примитива barA) по достижении порогов в 60 и 80 единиц. Для этого скрипт нужно дополнить следующим кодом.

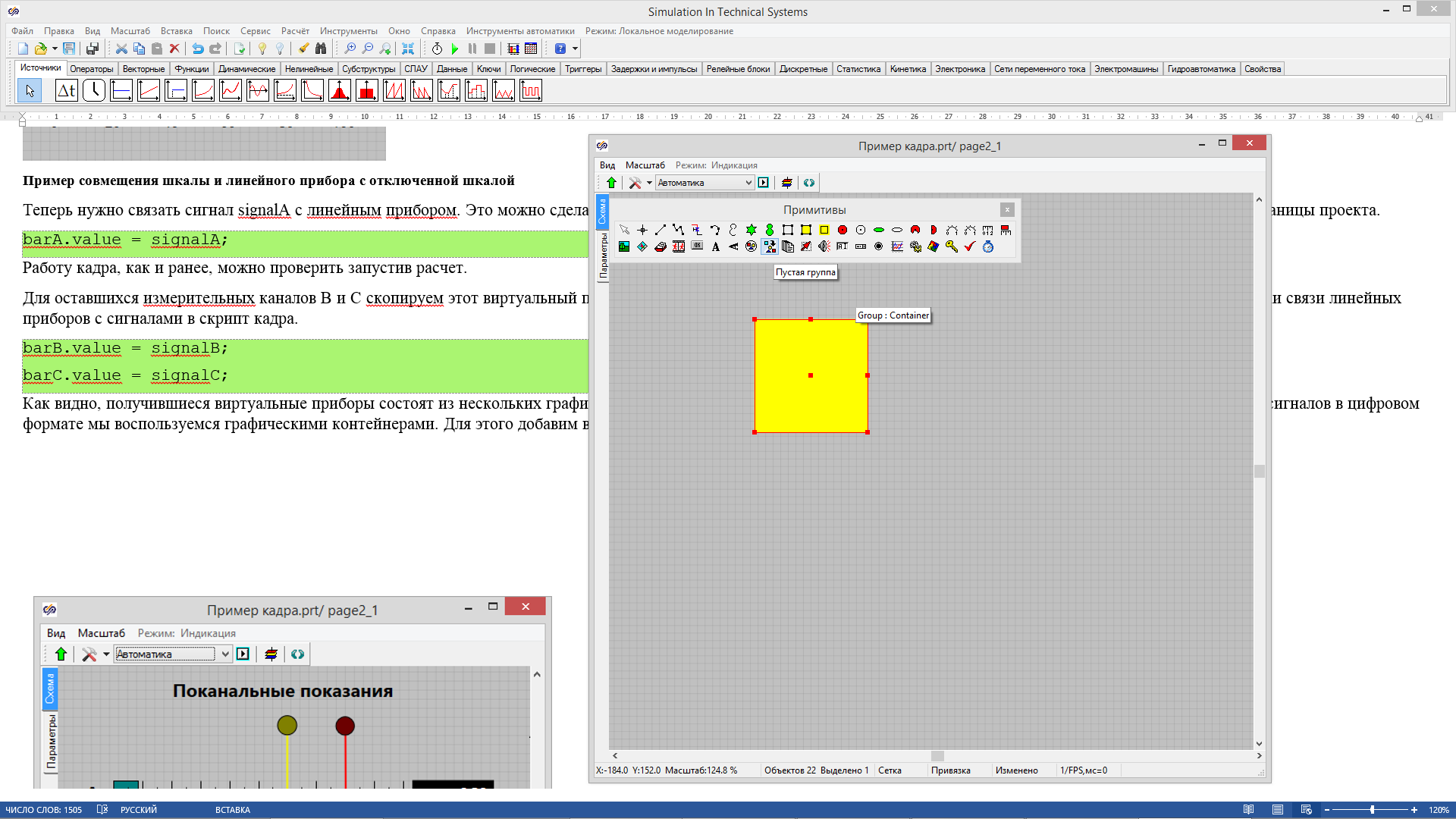
|  |
| --- |
| **if** signalA>=80  **then** barA.barcolor = **rgbtocolor**(240,0,0) //красный  **else**  **if** signalA>=60  **then** barA.barcolor = **rgbtocolor**(240,240,0) //желтый  **else** barA.barcolor = **rgbtocolor**(0,128,128); //цвет по умолчанию |

В данном случае можно использовать исходный сигнал, т.к. здесь определяется только цвет шкалы, а ее видимость определена ранее, поэтому нет необходимости применять какую-либо уникальную расцветку шкалы для случая неисправности измерительного канала.

Для оставшихся измерительных каналов B и C скопируем этот виртуальный прибор, переименуем копии линейных приборов в BarB и BarC соответсвенно и добавим строки связи линейных приборов с сигналами в скрипт кадра.

|  |
| --- |
| **if** diagA >=1 **then** signalAd=-1 **else** signalAd = signalA;  **if** diagB >=1 **then** signalBd=-1 **else** signalBd = signalB;  **if** diagC >=1 **then** signalCd=-1 **else** signalCd = signalC;  barA.value = signalAd;  barB.value = signalBd;  barC.value = signalCd;  **if** signalA>=80  **then** barA.barcolor = **rgbtocolor**(240,0,0) //красный  **else**  **if** signalA>=60  **then** barA.barcolor = **rgbtocolor**(240,240,0) //желтый  **else** barA.barcolor = **rgbtocolor**(0,128,128); //цвет по умолчанию  **if** signalB>=80  **then** barB.barcolor = **rgbtocolor**(240,0,0) //красный  **else**  **if** signalB>=60  **then** barB.barcolor = **rgbtocolor**(240,240,0) //желтый  **else** barB.barcolor = **rgbtocolor**(0,128,128); //цвет по умолчанию  **if** signalC>=80  **then** barC.barcolor = **rgbtocolor**(240,0,0) //красный  **else**  **if** signalC>=60  **then** barC.barcolor = **rgbtocolor**(240,240,0) //желтый  **else** barC.barcolor = **rgbtocolor**(0,128,128); //цвет по умолчанию |

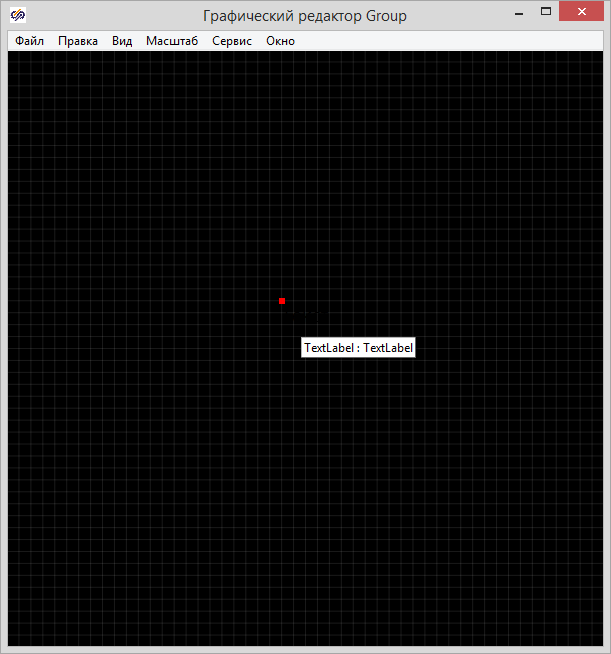
Как видно, получившиеся виртуальные приборы состоят из нескольких графических примитивов, что затрудняет манипуляции с ним. Поэтому для отображения значений сигналов в цифровом формате мы воспользуемся графическими контейнерами. Для этого добавим в схемное окно примитив «Пустая группа».



**Добавление примитива «Пустая группа»**

Сразу изменим фон примитива на черный. Для этого надо вызвать окно свойств группы, кликнув ПКМ на ее изображении и выбрав пункт контекстного меню «Свойства объекта». Цвет фона группы выбирается в одноименном свойстве «Цвет / Color». Название группы нужно изменить на «GroupA». Затем нужно закрыть окно свойств группы по кнопке «Оk».

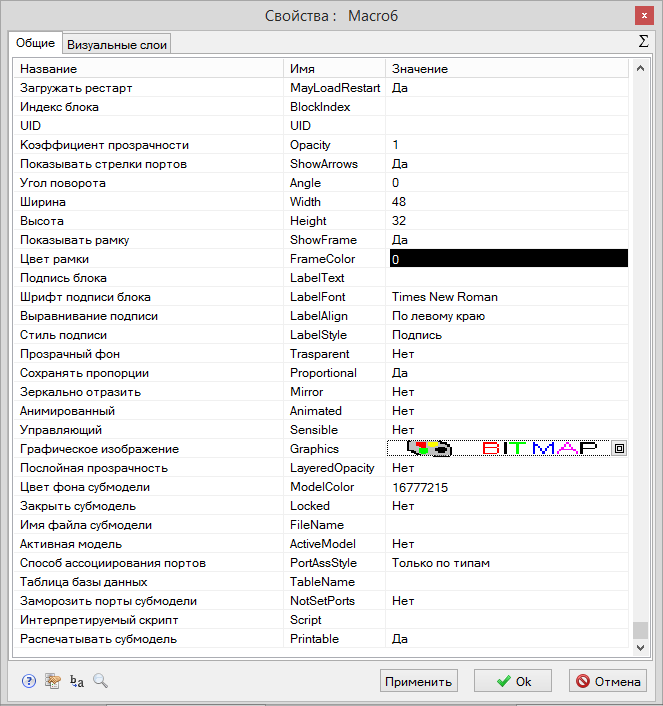
Далее приступим к редактированию содержимого группы. Для этого нужно дважды кликнуть ЛКМ на изображении группы – откроется окно графического редактора. Для отображения значений сигналов в цифровом формате воспользуемся примитивом «Текст».

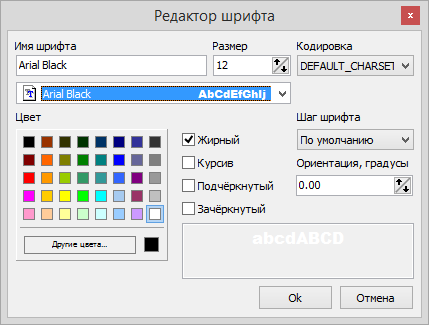


**Окно редактирования содержимого графической группы с добавленным примитивом «Текст»**

Для редактирования свойств примитива «Текст» нужно вызвать контекстное меню по лику ПКМ на его изображении и выбрать пункт «Свойства объекта». Нужно установить следующие ключевые свойства данного примитива:

* Имя объекта «Textlabel» (по умолчанию);
* Видимость при выполнении «Да» (по умолчанию);
* Текст «»;
* Способ показа цифр «Показывать справа»;
* Прозрачный фон «Да» (по умолчанию);
* Положение точки вставки «Справа»;
* Стиль выравнивания «Справа»;
* Шрифт – см. ниже.

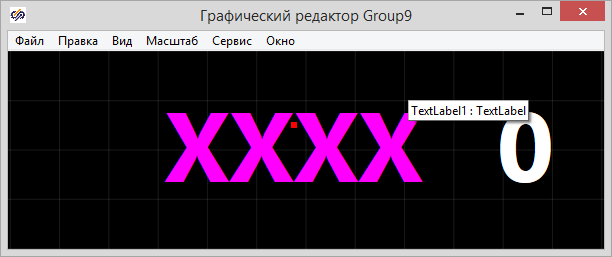
Для настройки цвета текста нужно открыть настройки шрифта кликом по кнопке  в поле свойства «Шрифт». Откроется соответствующее окно, в котором нужно выбрать белый цвет.



**Выбор цвета текста для примитива «Текст»**

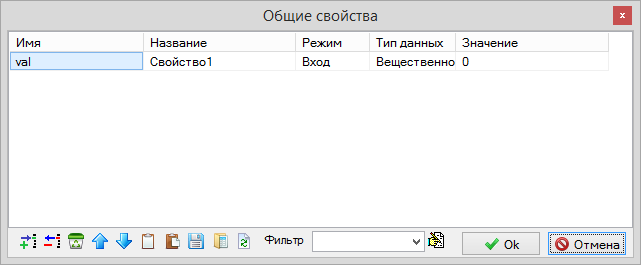
Затем нужно закрыть все окна настройки примитива «Текст», сместить сам примитив в правую часть окна графического редактора и сузить окно редактора так, чтобы было достаточно места для отображения будущих значений. Также нужно добавить в центр нашей группы ещё один текстовый примитив с текстом «XXXX», который будет отображаться при появлении неисправности. Свойства для данного примитива:

* Имя объекта «Textlabel1» (по умолчанию);
* Видимость при выполнении «Нет»;
* Текст «XXXX»;
* Способ показа цифр «Не показывать» (по умолчанию);
* Прозрачный фон «Да» (по умолчанию);
* Положение точки вставки «По центру»;
* Стиль выравнивания «По центру»;
* Шрифт – пурпурный цвет.



**Вид содержимого группы**

Для возможности передавать значения сигналов в свойства графических примитивов, находящихся внутри группы нужно создать в этой группе глобальные (общие) свойства. Эти свойства с одной стороны, будут входить в перечень свойств графической группы, доступных в схемном окне проекта и смогут принимать значения сигналов проекта, а с другой стороны, будут связываться внутри группы со свойствами графических примитивов. Для добавления глобальных свойств используется соответствующий редактор, вызываемый из меню окна графического редактора при редактировании соответствующей группы: **«ГР: Сервис→Глобальные свойства»**. Поскольку для отображения мы будем использовать переменную signalAd и подобные ей, в значениях которых уже закодирован факт неисправности измерительного канала, то нам нужно получить только это значение, для которого достаточно одного глобального свойства. Параметры добавляемого глобального свойства приведены на скриншоте ниже.

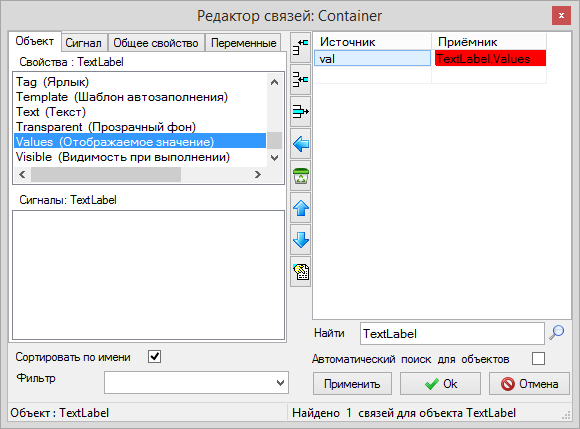


**Параметры глобального свойства для отображения измеряемых значений в цифровом формате**

После добавления глобального свойства нужно описать связи и логику отображения внутри редактируемой группы.н

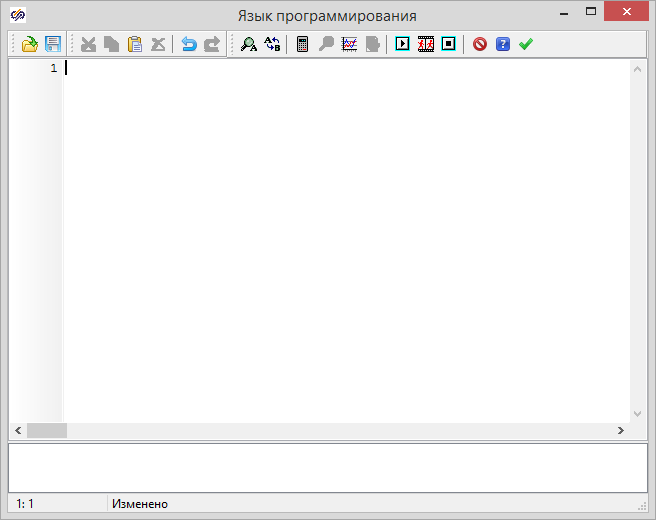
Связь между глобальным свойством val и свойством values текстового примитива TextLabel можно задать с помощью редактора связей: **«ГР: Сервис→Связи…»**.

В левой части окна во вкладке «Объект» нужно найти свойство «Values» (объект «TextLabel» при этом должен быть выделен) и перетащить его в колонку «Приемник» в правой части окна. Затем нужно открыть вкладку «Общее свойство» и перетащить свойство «val» в колонку «Источник» напротив заполенной ранее ячейки.



**Окно редактора связей**

Остальную логику отображения внутри группы нужно описать с помощью скрипта. Для этого нужно открыть соответствующий редактор, вызвав его из меню: **«Сервис» → «Скрипт»**. Нужно написать код, который будет изменять значения свойства «Visible» двух наших текстовых примитивов на противоположные при изменении значения val на «-1» и обратно.

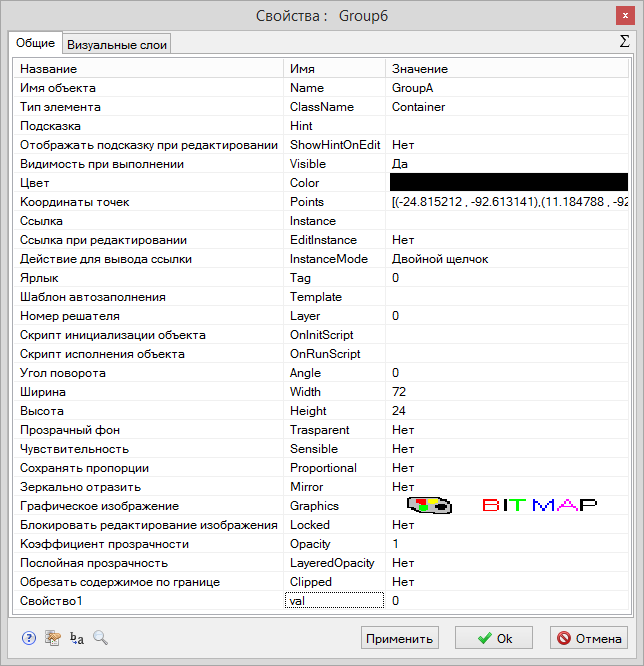


**Окно редактирования скриптов**

Предлагаемый код скрипта приведен ниже.

|  |
| --- |
| **if** val < 0  **then**  **begin**  textlabel.visible = 0  textlabel1.visible = 1  **end**  **else**  **begin**  textlabel.visible = 1  textlabel1.visible = 0  **end** |

Теперь можно закрыть все окна, связанные с графическим редактором и окно самого графического редактора с подтверждением вносимых изменений.

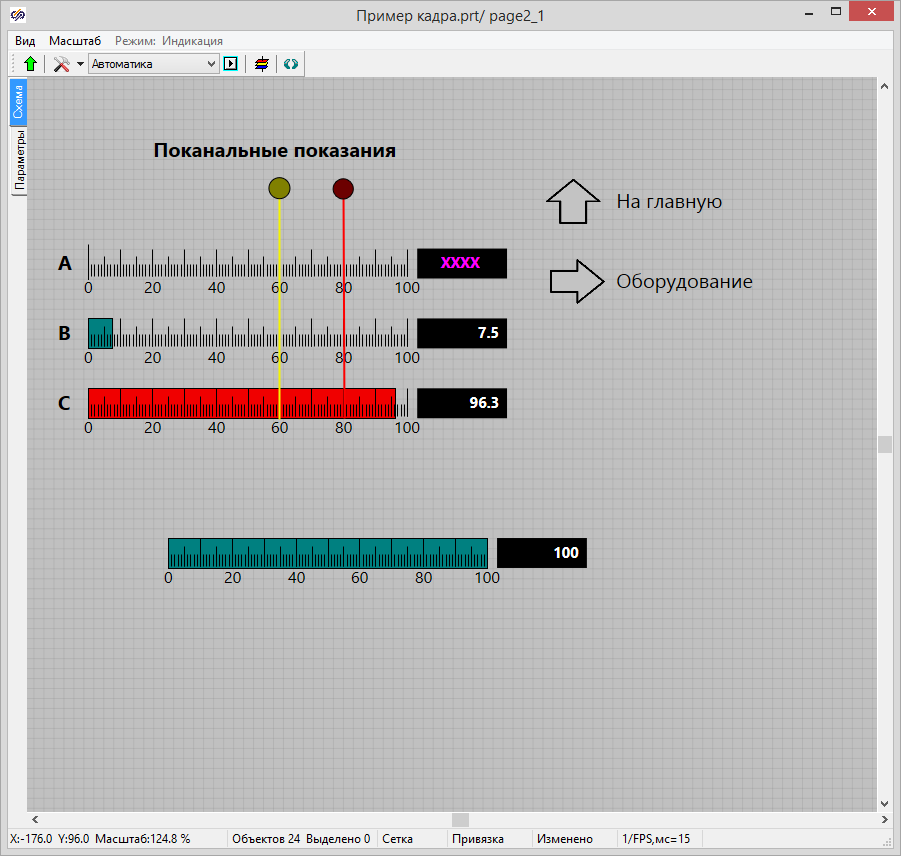


**Окно свойств группы «GroupA» с глобальным свойством «val»**

Новый виртуальный показывающий прибор готов, но ещё не связан с показаниями, которые он должен отображать. Дополним скрипт текущей страницы проекта следующим кодом:

|  |
| --- |
| groupA.val = signalAd; |

Теперь можно проверить работу показывающего прибора, запустив расчет.



**Пример внешнеого вида шкального и цифрового приборов**

После успешной отладки нужно сделать копии группы «GroupA» и переименовать копии в «GroupB» и «GroupC». Также нужно внести соответствующие дополнения в скрипт.

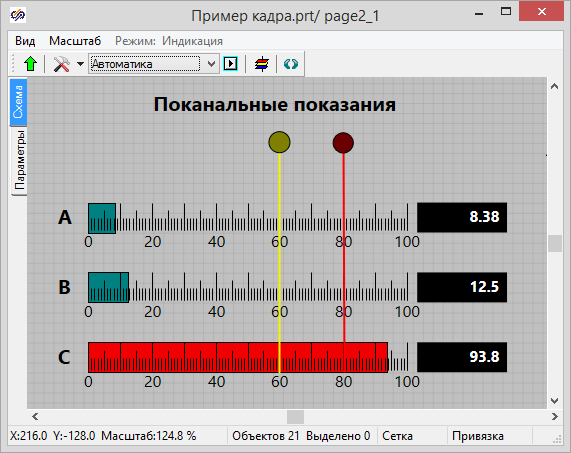
|  |
| --- |
| groupA.val = signalAd;  groupB.val = signalBd;  groupC.val = signalCd; |

По условиям примера на текущем кадре должны также отображаться факты одновременного превышения в двух и более каналах уставки в 60 единиц и аналогичная сигнализации для уставки в 80 единиц.

Создадим два табло на основе примитивов FillCircle и переименуем их соответственно в «lamp\_yell» и «lamp\_red». Для организации засветки каждого из этих табло дополним скрипт следующим кодом:

|  |
| --- |
| **if** ((signalAd>=80) **AND** (signalBd>=80)) **OR** ((signalAd>=80) **AND** (signalCd>=80)) **OR** ((signalBd>=80) **AND** (signalCd>=80))  **then** mes\_res = 2  **else**  **if** ((signalAd>=60) **AND** (signalBd>=60)) **OR** ((signalAd>=60) **AND** (signalCd>=60)) **OR** ((signalBd>=60) **AND** (signalCd>=60))  **then** mes\_res = 1  **else** mes\_res = 0;  **if** mes\_res >= 1  **then** lamp\_yell.color = rgbtocolor(240,240,0) //желтый  **else** lamp\_yell.color = rgbtocolor(128,128,0); //приглушенный желтый  **if** mes\_res = 2  **then** lamp\_red.color = rgbtocolor(240,0,0) //красный  **else** lamp\_red.color = rgbtocolor(108,0,0); //приглушенный красный |

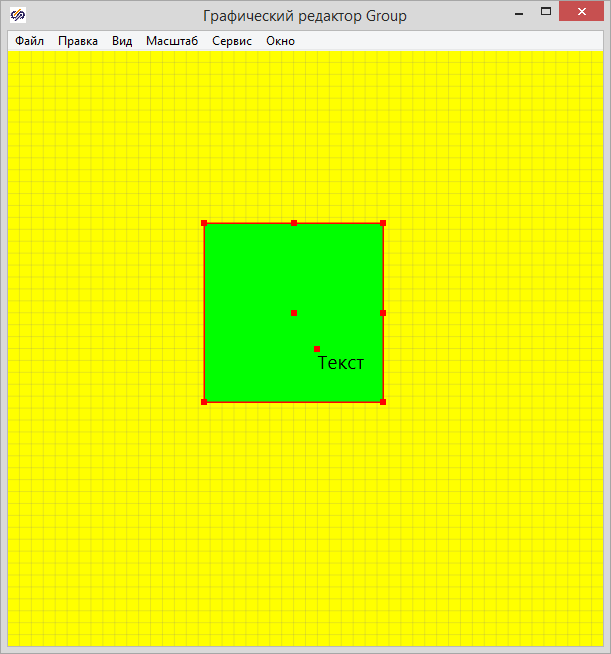
Получившийся кадр можно дополнить пояснящими подписями.



**Вид кадра page2\_1**

**Кадр верхнего уровня**

Перейдем в субмодель «page1». Данная страница проекта должна стать кадром с обобщенной сигнализацией. Для этого мы создадим еще один виртуальный показывающий прибор на основе графической группы. Для этого добавим в текущую страницу проекта примтив «пустая группа» и откроем его для редактирования двойным кликом ЛКМ. В октрывшемся окне графического редакторе добавим графический примитив «залитый прямоугольник» и примтив «Текст».



**Вид редактируемой группы**

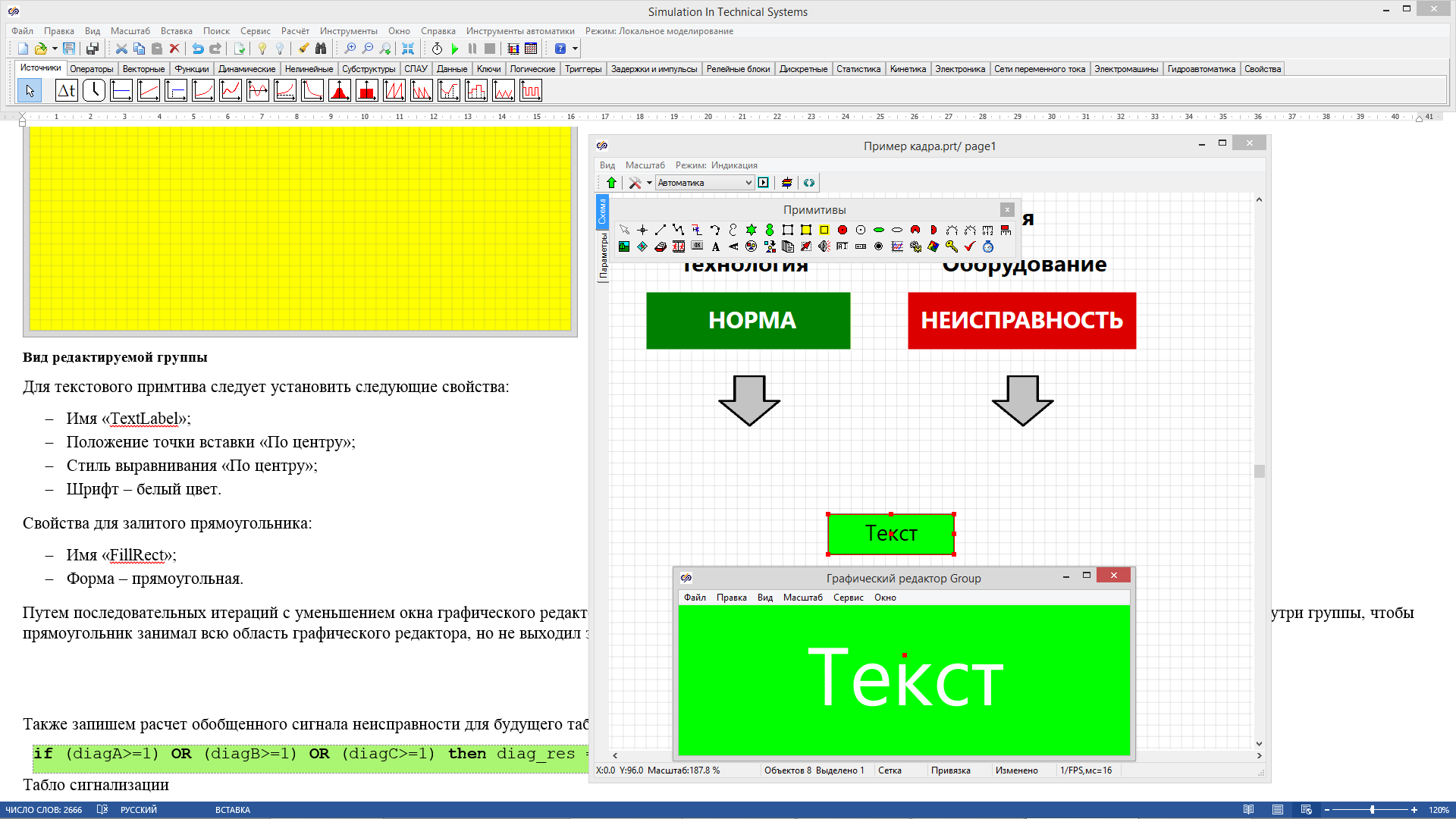
Для текстового примтива следует установить следующие свойства:

* Имя «TextLabel»;
* Положение точки вставки «По центру»;
* Стиль выравнивания «По центру»;
* Шрифт – белый цвет.

Свойства для залитого прямоугольника:

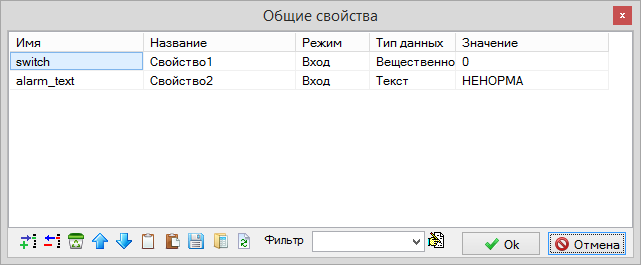
* Имя «FillRect»;
* Форма – прямоугольная.

Путем последовательных итераций с уменьшением окна графического редактора, его закрытием и повторным открытием следует добиться такого размещения объектов внутри группы, чтобы прямоугольник занимал всю область графического редактора, но не выходил за границы графической группы, а текст был в центре группы.



**Вид отформатированной группы в схемном окне и в графическом редакторе**

Наш виртуальный прибор должен быть расчитан на работу от одного внешнего сигнала, принимающего значения «0» - норма, «1» - предупредительная сигнализация и «2» - аварийная сигнализация. При этом должен меняться как цвет фонового прямоугольника, так и текст надписи. Для передачи значений внешнего сигнала нужно создать два глобальных свойства: первое для засветки табло и выбора варианта текста, а второе для вставки своего варианта текста для аварийного значения сигнала. Параметры глобальных свойств приведены на скриншоте ниже.

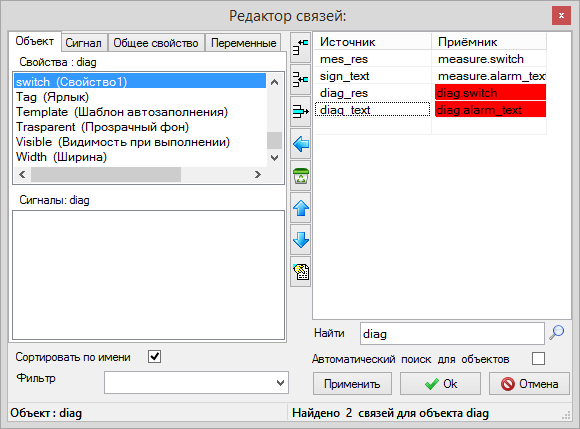


**Глобальные свойства для табло обобщенной сигнализации**

Описание связей внутри группы и логику работы мы реализуем в скрипте группы.

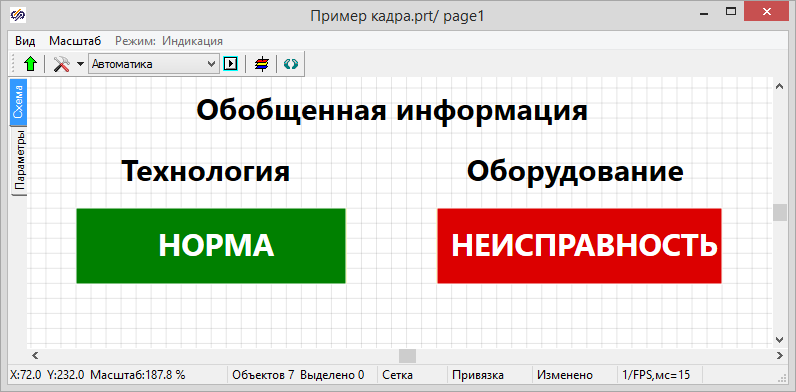
|  |
| --- |
| **if** (switch = 0)  **then**  **begin**  fillrect.color = **rgbtocolor**(0, 128, 0) //зеленый  textlabel.text = "НОРМА"  **end**  **else** **if** (switch = 1)  **then**  **begin**  fillrect.color = **rgbtocolor**(255, 235, 0) //желтый  textlabel.text = "ПС"  **end**  **else if** (switch = 2)  **then**  **begin**  fillrect.color = **rgbtocolor**(220, 0, 0) //красный  textlabel.text = alarm\_text  **end;** |

Теперь можно закончить редактирование группы и описать связи в схемном окне проекта. В нашем случае это можно сделать с помощью редактора связей. Редактор связей на уровне схемного окна проекта можно вызвать из меню Главного Окна: **«Сервис» → «Связи…»**.



**Связи для двух табло обобщенной сигнализации на кадре page1**

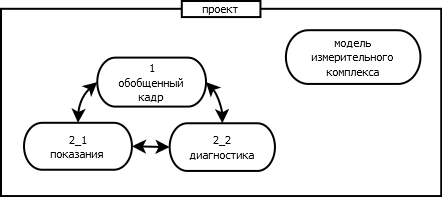
Кадр можно дополнить поясняющими подписями.



**Вид кадра обобщенной сигнализации page1**

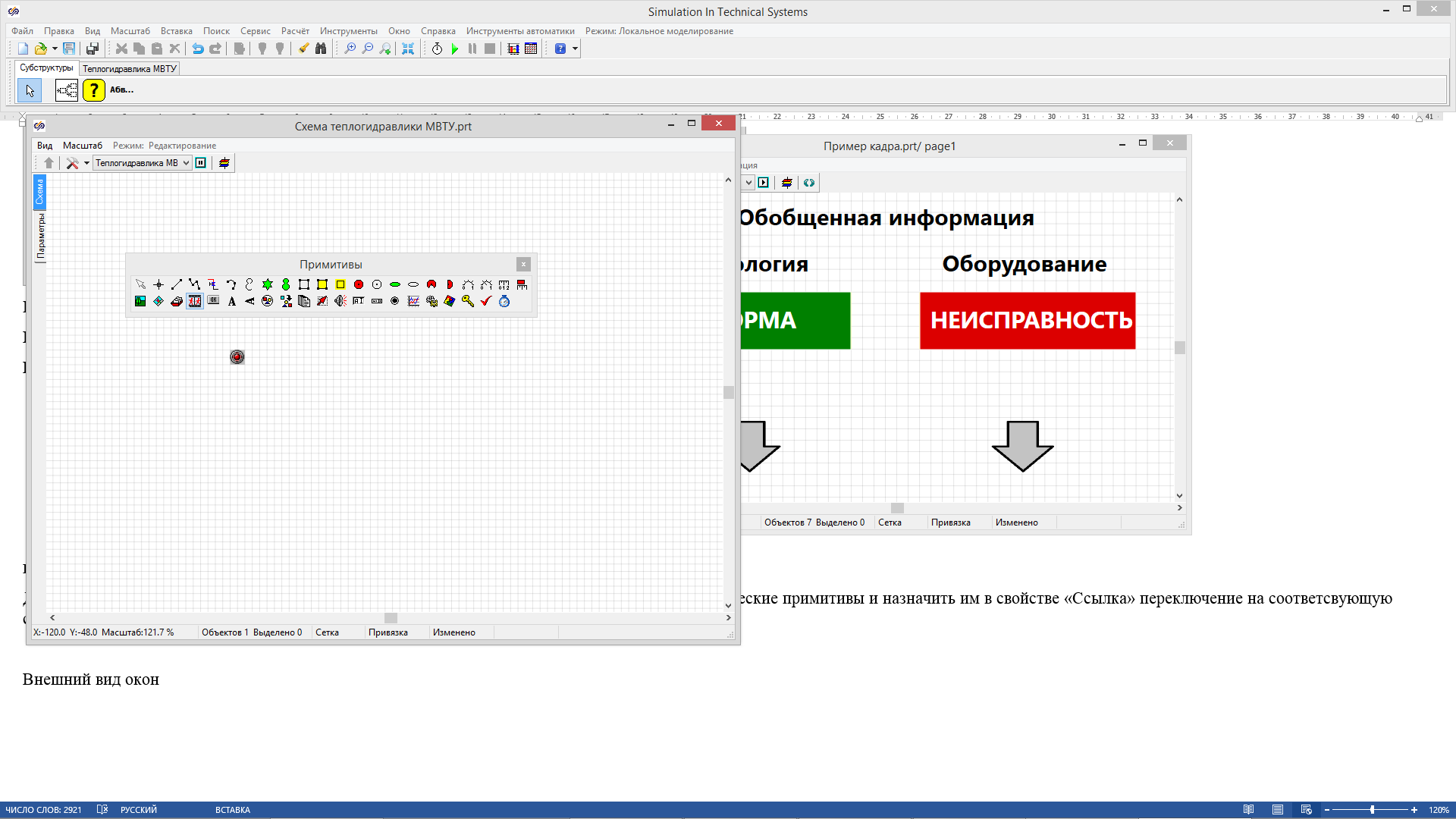
**Навигация**

По условию задачи нам нужно организовать переключение между кадрами проекта так, как это показано на схеме ниже.



**Навигация в создаваемом проекте**

Для осуществления переходов достаточно добавить на кадры любые логические блоки или графические примитивы и назначить им в свойстве «Ссылка» переключение на соответсвующую страницу проекта. Воспользуемся для этих целей графическим примитивом «Набор изображений / ImageList».



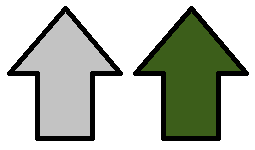
**Вид примитива «Набор изображений / ImageList» по умолчанию**

Данный примитив примечателен тем, что в качестве отображаемого изображения ему можно указать набор картинок одинакового размера, объединенных в одном файле, а также указать число этих картинок. Затем пользователь может, указывая индекс требуемой картинки, выбирать то изображение, которое должно отображаться в примитиве в данный момент.

В первую очередь добавим кнопки перехода в кадр верхнего уровня. Для этого добавим в страницу проекта page1 один графический примитив «Набор изображений / ImageList» и зададим ему слудющие свойства:

* Имя «ImageList» (по умолчанию);
* Растровое изображение – exitarrows.png (см. ниже);
* Прозрачность «Да» (по умолчанию);
* Прозрачный цвет – белый (по умолчанию);
* Число кадров «2» (по умолчанию);
* Индекс кадра «0» (по умолчанию).

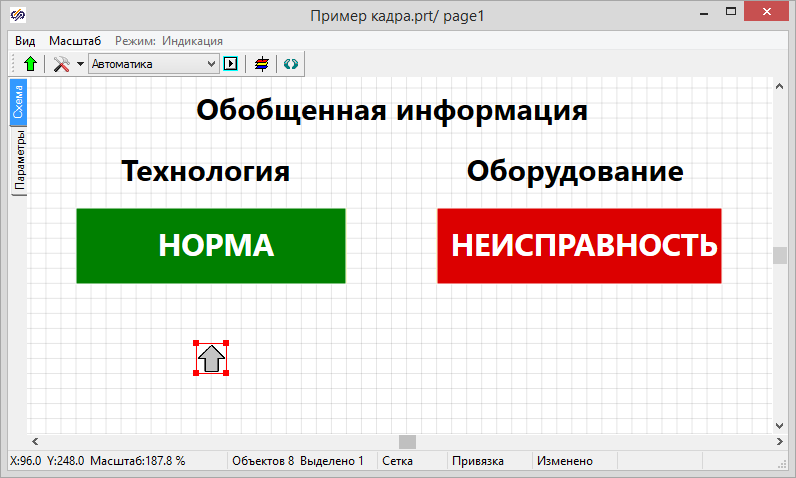
В качестве изображения для данного примитива укажем файл exitarrows.png.



**Изображение для кнопки перехода между кадрами**

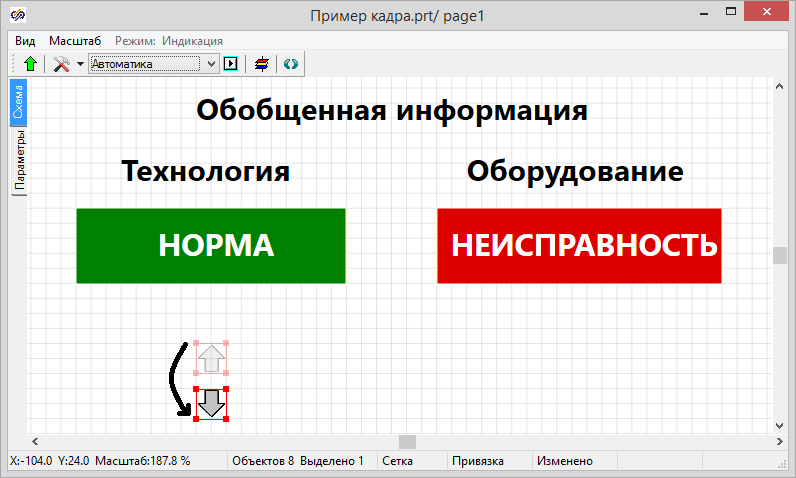
Данное изображение достаточно простое и может быть изготовлено самостоятельно. Главным условием корректного отображения картинок в примитиве SimInTech является равный размер «кадров», поскольку именно так примитив автоматически поделит исходное изображение. Несмотря на то, что по умолчанию в форме добавления растрового изображения предлагается указать файл формата .bmp, можно также использовать файлы форматов .png .jpg.

После добаввления изображения в страницу page1 должно получиться следующее изображение.



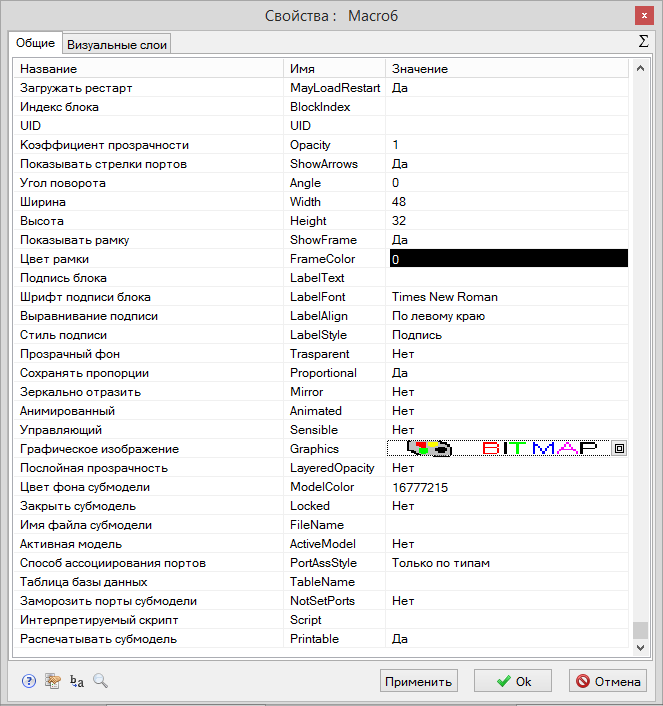
**Промежуточный вид обобщенного кадра при создании кнопки**

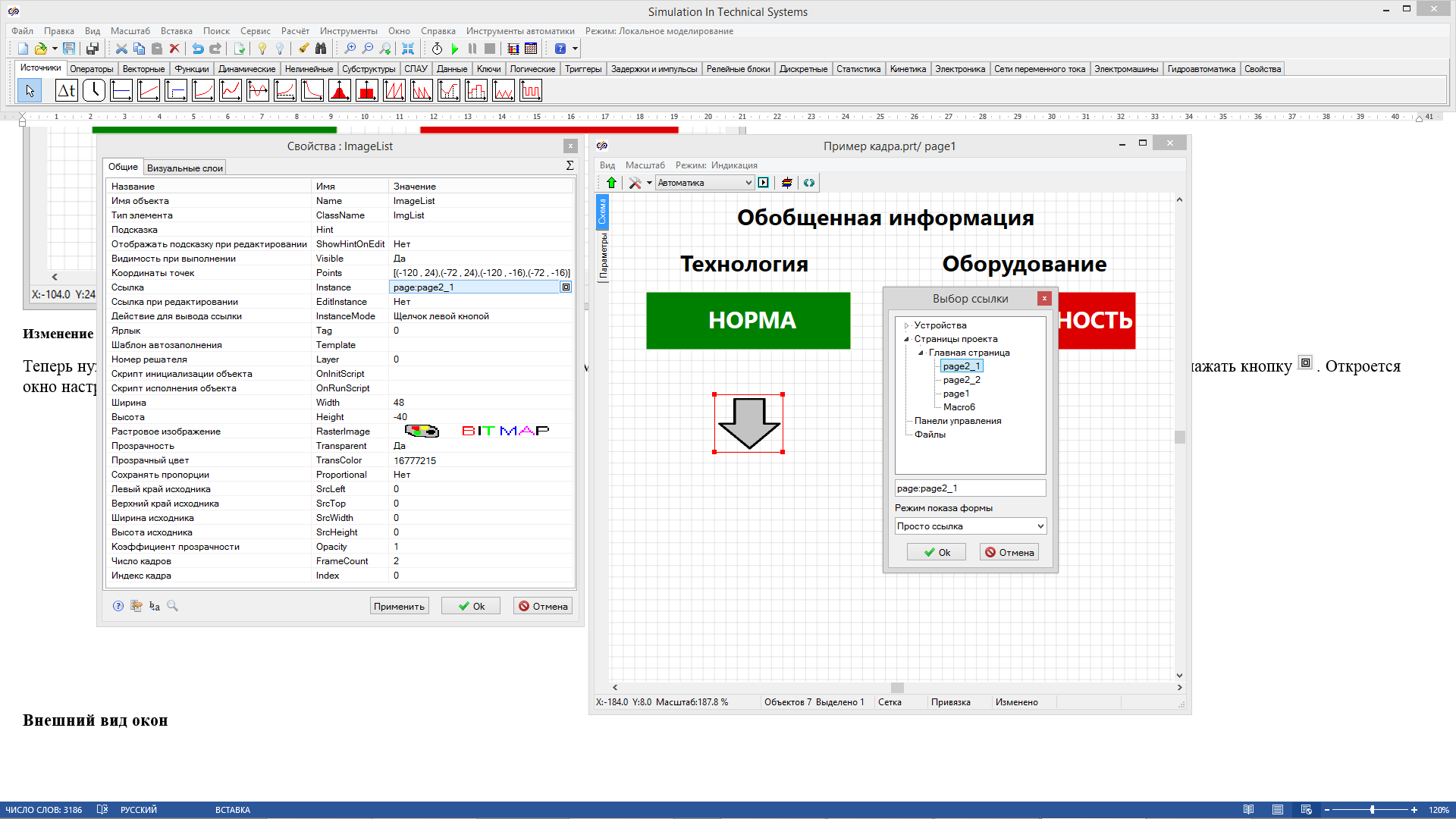
Поскольку данная кнопка будет использоваться для перехода на кадры более низкого уровня, логично перевернуть ее изображение. Это можно сделать просто переместив любой из верхних маркеров вниз, под нижние.



**Изменение направления стрелки**

Для удобства можно увеличить размер будущей кнопки.

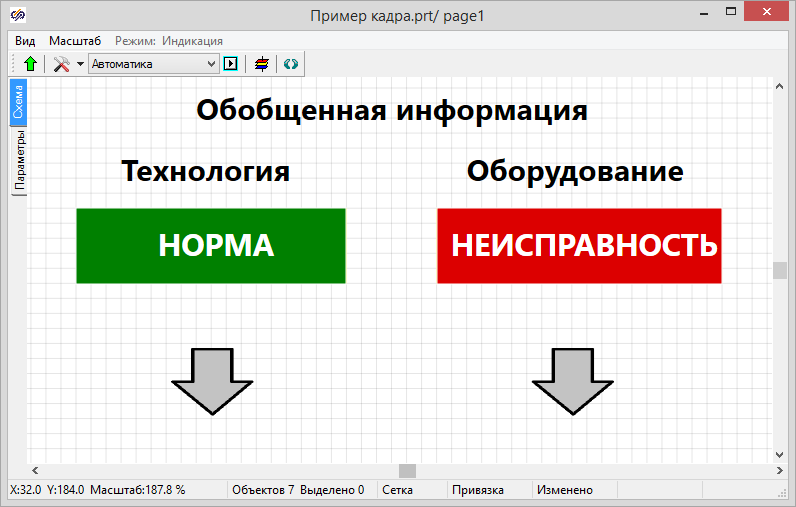
Теперь для осуществления собственно возможности перехода нужно указать параметры ссылки для данного примитива. Для этого нужно в поле свойства «Ссылка» нажать кнопку . Откроется окно настройки ссылок с отображением структуры текущего проекта. Нас интересует раздел «Страницы проекта», подраздел «page2\_1». Режим показа формы: «просто ссылка».



**Настройка ссылки для перехода на другой кадр**

Также в свойстве «Действие для вывода ссылки / InstanceMode» укажем «Щелчок левой кнопке». В этом случае переход на кадр page2\_1 будет происходить по клику ЛКМ на изоражении примитива.

Теперь можно скопировать полностью настроенный примитив и поместить копию под вторым табло, изменив в свойствах копии ссылку с page2\_1 на page2\_2.



**Вид обобщенного кадра с добавленными кнопками перехода на детальные кадры**

Для того, чтобы обратить внимание пользователя на интерактивность наших кнопок, мы можем менять изображение внутри примитива при попадани курсора мыши в границы примитива. Для этого нужно в скрипте текущего кадра написать следующий код.

|  |
| --- |
| p = **getmousepos**; //координаты курсора мыши  c = ImageList.Points[3]; //левый верхний угол кнопки  s = ImageList.Points[2]; //правый нижний угол кнопки  px = **xcoord**(p); //координата X мыши  py = **ycoord**(p); //координата Y мыши  cx = **xcoord**(c); //координата X л.в. угла кнопки  cy = **ycoord**(c); //координата Y л.в. угла кнопки  sx = **xcoord**(s); //координата X п.н. угла кнопки  sy = **ycoord**(s); //координата Y п.н. угла кнопки  **if** (px > cx) **AND** (px < sx) **AND** (py > cy) **AND** (py < sy) //проверка на попадание курсора мыши в пределы примитива  **then** ImageList.Index = 1  **else** ImageList.Index = 0; //задание индекса изображения по умолчанию для кнопки |

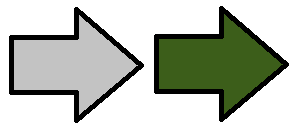
В данном коде сначала происходит запись в переменную p координаты точки (пара координат X и Y), в которой находится курсор мыши, с помощью функции getmousepos. Затем в переменные c и s записываются пары коодинат противополжных углов примитива. Для того чтобы сравнить значения отдельных координат их нужно выделить из получившихся переменных с помощью функций xcoord и ycoord. После того, как получены отдельные переменные со значениями X и Y всех интересующих нас точек, происходит проверка на попадание координат курсора мыши в диапазон координат нашего примитива-кнопки. При положительном результате индекс изображения, отображаемого внутри примитива должен измениться с «0» на «1». В нашем примере это означает то, что должна отобразиться закрашенная стрелка. Если же условие не выполнится, то индекс будет задан равным «0», что соответствует изображению незалитой стрелки. Для проверки работы кода нужно запустить расчет и провести курсором мыши над примитивом ImageList. Если все сделано верно, при пересечении курсором прямоугольника примитива стрелка исзменится с незалитой на залитую, а при выходе курсора из этой зоны стрелка вновь изменится на незалитую. Для второй кнопки нужно повторить весь приведенный выше код, учитывая необходимость создания своего набора переменных и уникальное имя примитива ImageList1. Пример полного кода скрипта для обобщенного кадра приводится ниже.

|  |
| --- |
| p = **getmousepos**; //координаты курсора мыши  c = ImageList.Points[3]; //левый верхний угол кнопки  s = ImageList.Points[2]; //правый нижний угол кнопки  c1 = ImageList1.Points[3]; //левый верхний угол кнопки1  s1 = ImageList1.Points[2]; //правый нижний угол кнопки1  px = **xcoord**(p); //координата X мыши  py = **ycoord**(p); //координата Y мыши  cx = **xcoord**(c); //координата X л.в. угла кнопки  cy = **ycoord**(c); //координата Y л.в. угла кнопки  sx = **xcoord**(s); //координата X п.н. угла кнопки  sy = **ycoord**(s); //координата Y п.н. угла кнопки  c1x = **xcoord**(c1); //координата X л.в. угла кнопки1  c1y = **ycoord**(c1); //координата Y л.в. угла кнопки1  s1x = **xcoord**(s1); //координата X п.н. угла кнопки1  s1y = **ycoord**(s1); //координата Y п.н. угла кнопки1  **if** (px > cx) **AND** (px < sx) **AND** (py > cy) **AND** (py < sy) //проверка на попадание курсора мыши в пределы примитива  **then** ImageList.Index = 1  **else** ImageList.Index = 0; //задание индекса изображения по умолчанию для кнопки  **if** (px > c1x) **AND** (px < s1x) **AND** (py > c1y) **AND** (py < s1y) //проверка на попадание курсора мыши в пределы примитива1  **then** ImageList1.Index = 1  **else** ImageList1.Index = 0; //задание индекса изображения по умолчанию для кнопки1 |

**Кадр измеряемых показаний**

В данном кадре также должны присутствовать две кнопки – одна для перехода на кадр верхнего уровня, вторая для перехода на кадр диагностики. Для первой кнопки можно скопировать уже готовый примитив из обобщенного кадра, не забыв перевернуть изображение так, чтобы стрелка указывала вверх. Также нужно будет внести соответствующие изменения в условие сравнения координат вдоль оси Y. Если раньше происходила проверка на достоверность условия cy < py < sy, то после преобразования необходимо проверять условие sy < py < cy. Кроме того в свойстве «Ссылка» примитива ImageList нужно указать страницу проекта «page1».

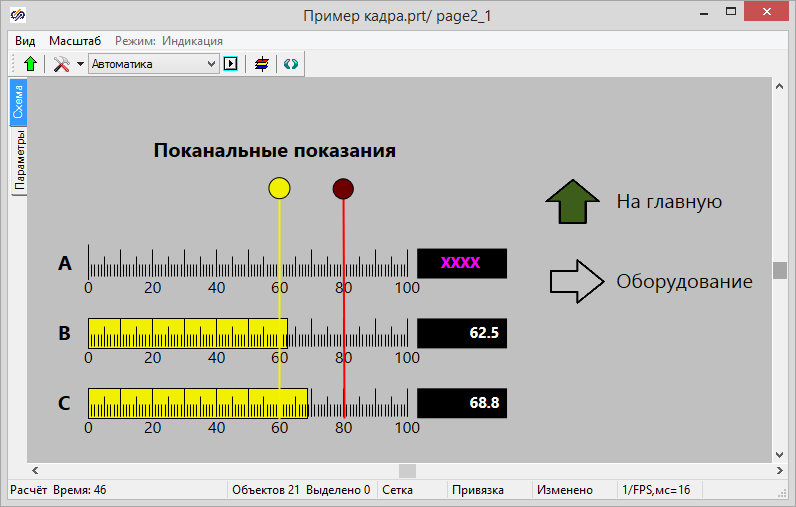
Для создания второй кнопки логично было бы изобразить стрелку, указывающую в горизонтальном направлении, т.к. по нашей структуре кадр диагностики и кадр отображения показаний принадлежат одному уровню. Но используемымые нами примитивы «Набор изображений» не поддерживают возможность вращения изображения. По этому для второй кнопки нужно взять другое изображение стрелки.



**Рисунок для кнопки перехода на кадр того же уровня**

Имя второго примитива «ImageList1», ссылка «page2\_2». Все остальные атрибуты второго примитива должны соответствовать атрибутам первого. Сопутствующий код для страницы page2\_1 выглядит следующим образом.

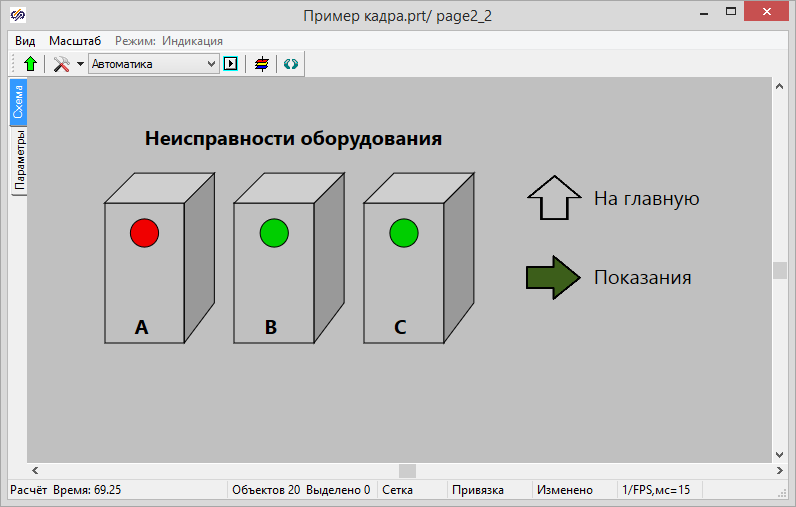
|  |
| --- |
| p = **getmousepos**;  c = ImageList.Points[3];  s = ImageList.Points[2];  c1 = ImageList1.Points[3];  s1 = ImageList1.Points[2];  px = **xcoord**(p);  py = **ycoord**(p);  cx = **xcoord**(c);  cy = **ycoord**(c);  sx = **xcoord**(s);  sy = **ycoord**(s);  c1x = **xcoord**(c1);  c1y = **ycoord**(c1);  s1x = **xcoord**(s1);  s1y = **ycoord**(s1);    **if** (px > cx) **AND** (px < sx) **AND** (py < cy) **AND** (py > sy)  **then** ImageList.Index = 1  **else** ImageList.Index = 0;  **if** (px > c1x) **AND** (px < s1x) **AND** (py < c1y) **AND** (py > s1y)  **then** ImageList1.Index = 1  **else** ImageList1.Index = 0; |



**Вид кадра показаний с кнопками навигации**

**Кадр диагностики**

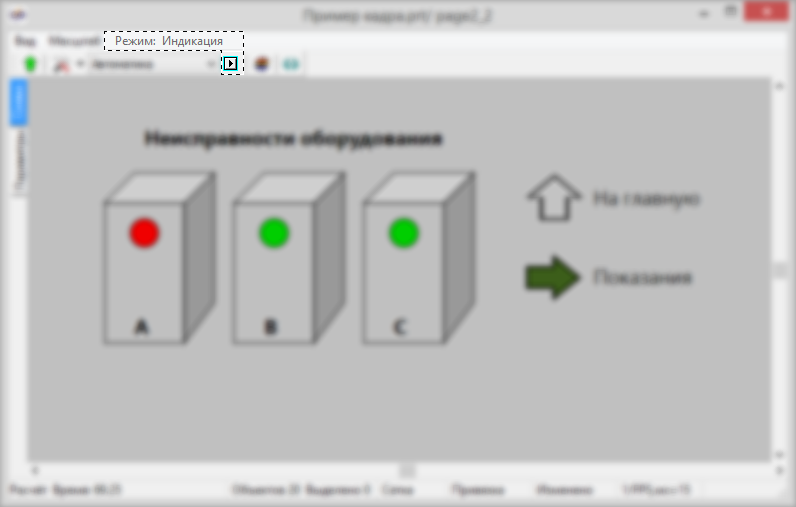
Для кадра диагностики целесообразно скопировать обе кнопки и сопутствующий им код из кадра показаний без каких-либо изменений за исключением замены ссылки в примитиве ImageList1 с «page2\_2» на «page2\_1».



**Вид кадра диагностики с кнопками навигации**

**Внешний вид окон**

В целях минимизации возможностей имеющихся у пользователя для изменения внешнего вида кадров можно воспользоваться рядом опций. Однако прежде, чем вносить изменения стоит убедиться в том, что все страницы проекта, являющиеся кадрами, переведены в режим «Индикация».

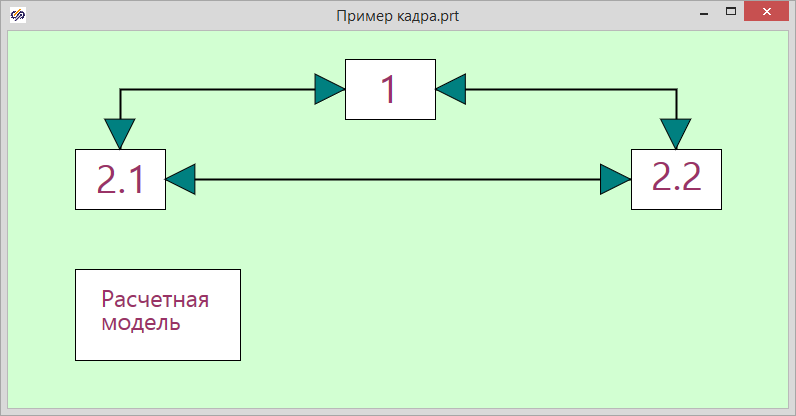


**Режим страницы проекта «Индикация»**

В меню «Вид» схемного окна проекта можно отключить следующие опции:

* Дополнительно:
  + Блокировать масштабирование и перемещение;
* Полосы прокрутки;
* Строка состояния;
* Закладки режима редактора;
* Панели инструментов:
  + Слои.

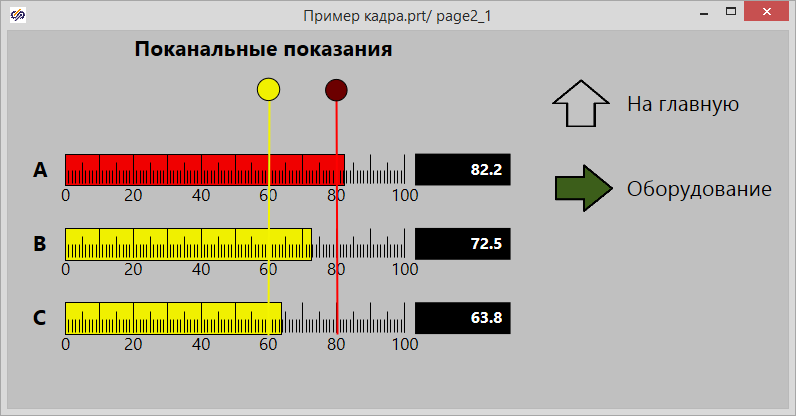
Также в контекстном меню схемного окна можно включить опцию «Скрыть меню окна». Данную опцию нужно применить отдельно для каждой страницы проекта.



**Внешний вид главной страницы проекта**



**Кадр обобщенной сигнализации**



**Кадр отображения измеряемых показаний**



**Кадр диагностики**