# TouchGFX

## Описание элементов:

Виджеты:

Все наименования виджетов начинаются с типа виджета для лучшей идентификации:

* BTN – кнопка
* BOX – рамка с подложкой
* Value – текстовое поле, в которое передается значение, содержит Wildcard
* Label – текстовое поле
* State – текстовое поле для отображения состояния (будет удалено в будущих версиях)
* IMG – изображение
* ANIM – анимация

Interactions:

GoTo… - переход на экран

## Список экранов

### Home

Данные для отображения:

* Температура ядра, Text ID: ValueCoreT1, ValueCoreT2
* Оставшееся время программы: ValuePRGTimeLeft
* Кнопка «Старт»: BTNStart, объект FlexButton
* Кнопка «Стоп»: BTNStop, объект FlexButton

Interactions:

* GoToVisualization – переход на страницу Visualization
* GoToSettings1

Планы на доработку:

* Смена состояния кнопок Старт и Стоп для исключения состояния, когда нажаты обе кнопки.

### Visualization

Данные для отображения:

* Состояние тэнов 1-4. На данный момент реализовано отображением текстовых полей StateHeat1 … StateHeat4 и запуском анимации AnimHeat12 и AnimHeat34 при включении одно из пары тэнов.
* Состояние вентиляторов 1-4. На данный момент реализовано отображением текстовых полей StateFan1 … StateFan4 и запуском анимации AnimFan12 и AnimFan34 при включении одно из пары вентиляторов.
* Данные по влажности в камере: ValueDefrosterH
* Данные по температуре в камере от двух датчиков: ValueDefrosterT1, ValueDefrosterT2
* Данные по температуре в рыбе от двух датчиков: ValueCoreT1Small, ValueCoreT2Small (Text ID с припиской Small, т.к. Typography используется Small, 10px)

Interactions:

* GoToHome
* GoToSettingsMenu

Планы на доработку:

* Отказ от отображения текстовых полей, вместо них сделать анимации с двумя парами вентиляторов и двумя парами тэнов. Вентиляторы начинают вращаться, а тэны меняют цвет при включении.

### SettingsMenu

*Данные для отображения:*

* Нет

*Interactions:*

* GoToHome
* GoToSettings1\_1 (стрелка в углу экрана)
* GoToSettings1
* GoToSettings2
* GoToSettings3
* GoToSettings4

*Планы на доработку:*

* Не планируется

### Settings1

*Данные для отображения:*

* Использование датчиков температуры в теле продукта. Для этого используется Toggle Button BTNCoreTSensorsState. По умолчанию стоит значение ON.
* Температура в теле продукта. Задается через **int CoreTSet.** Описание ниже в списке переменных

*Interactions:*

* GoToHome
* GoToSettings2
* GoToSettings4
* GoToSettingsMenu
* BTNCoreTSetIncreaseClicked – вызов виртуальной функции BTNCoreTSetIncreaseClicked, отвечающей за увеличение значения CoreTSet и ограничения её в 8 градусов.
* BTNCoreTSetDecreaseClicked – аналогично Increse, устанавливает ограничение температуры не ниже 0 градусов.

*Планы на доработку:*

Возможно отключение датчика по одному и отображение их значений. Надо думать.

### Settings2

*Данные для отображения:*

Состояние ворот указывается кнопкой BTNGateControl. OFF = закрыто.

Клапан воды для форсунок указывается кнопкой BTNSprayControl. OFF = закрыто

*Interactions:*

* GoToHome
* GoToSettings1
* GoToSettings3
* GoToSettingsMenu
* BTNGateManualClicked - вызов виртуальной функции BTNGateManualClicked, отображающей кнопку для открытия и закрытия ворот.
* BTNSprayManualClicked - вызов виртуальной функции BTNSprayManualClicked, отображающей кнопку для открытия и закрытия ворот и отключающий форсунки.

*Планы на доработку:*

* Нет

### Settings3

*Данные для отображения:*

* Состояние вентиляторов 1, 2: BTNFanControl1, BTNFanControl2
* Состояние тэнов 1, 2: BTNHeatControl1, BTNHeatControl2

*Interactions:*

* GoToHome
* GoToSettings1
* GoToSettings3
* GoToSettingsMenu
* BTNManual1Clicked – вызов виртуальной функции BTNManual1Clicked. Включение отображения кнопки управления вентиляторами BTNFanControl1. При отключении ручного управления выключается тэн и вентилятор группы 1. Скрываются кнопки управления группы 1.
* BTNFanControl1Clicked - вызов виртуальной функции BTNFanControl1Clicked. Включение отображения кнопки управления тэном BTNHeatControl1. При отключении вентилятора выключается тэн. Скрывается кнопка управления тэном.

*Планы на доработку:*

* Нет

### Settings4

*Аналогично Settings3.*

## Список переменных

1. **int CoreTSet** – задание температуры ядра. Отображается и изменяется на экране Settings1

Инициализируется в файле: Project\_Defrost-Anatoly\TouchGFX\gui\include\gui\settings1\_screen\Settings1View.hpp

Изменение значения на экране происходит функциями BTNCoreTSetIncrease, BTNCoreTSetDecrease. Также функциями происходит ограничения изменения значения от 0 до 8 градусов цельсия

Функции описаны в файле:

Project\_Defrost-Anatoly\TouchGFX\gui\src\settings1\_screen\Settings1View.cpp

1. int DefrosterT1 – температура в дефростере, датчик №1.
2. Settings5

*Данные для отображения:*

Экран программирования датчиков. Выбирается тип датчика через всплывающее окно. Отображается текущие скорость и адрес датчика. Через всплывающие окна выбираются новые скорость и адрес датчика. Присутствует кнопка записи для отправки новых значений в датчик.

*Interactions:*

* GoToHome
* GoToSettings1
* GoToSettings3
* GoToSettingsMenu
* BTNSensorTypeClicked – вызов виртуальной функции, отображение скроллового окна на экране. Подробнее смотри ниже.

*Планы на доработку:*

## Работа скроллового окна

1. Инструкция: <https://support.touchgfx.com/docs/development/ui-development/ui-components/containers/scroll-wheel#user-code>
2. В TouchGFX необходимо создать контейнеры с элементами списка и если требуется, то контейнер, который будет описывать активный элемент.
3. При создании ScrollList на экране в TouchGFX указывается количество элементов и «элемент по умолчанию» и контейнеры для отображения данных.
4. .cpp файлы контейнеров находятся по адресу: Defrost\TouchGFX\gui\src\containers

.hpp файлы контейнеров находятся по адресу: Defrost\TouchGFX\gui\include\gui\containers

1. В .cpp прописывается код для обновления содержимого текстового поля ScrollItem в самом контейнере:

void ScrollItemContainer::updateScrollItem(int16\_t value)

{

Unicode::snprintf(ScrollItemBuffer, SCROLLITEM\_SIZE, "%d", value);

ScrollItem.invalidate();

}

В .hpp прописывается функция: void updateScrollItem(int16\_t value);

1. Аналогично для ScrollSelectedItemContainer:

.cpp:

**void** ScrollSelectedItemContainer::**updateScrollSelectedItem**(int16\_t value)

{

Unicode::snprintf(ScrollSelectedItemBuffer, SCROLLSELECTEDITEM\_SIZE, "%d", value);

ScrollSelectedItem.invalidate();

}

.hpp:

void updateScrollSelectedItem(int16\_t value);

1. В файлах экранов уже идет обращение к конкретному «скроллу». Может быть несколько скроллов, которые будут использовать одни и теже контейнеры для вывода содержимого (хоть текст, хоть картинку, смену цвета и вообще что хочешь).
2. Чтобы начать изменять данные скролла нужно взять код его функции в файле: \Defrost\TouchGFX\generated\gui\_generated\include\gui\_generated\settings5\_screen\ Settings5ViewBase.hpp
3. В файле .cpp экрана будет следующее (на примере selected item, где кроме обновления содержимого «центрального» контейнера еще записывается переменная int SetSpeed, которая при нажатии кнопки «записать» отобразится на кнопке выбора скорости и может быть записана в датчик. Почему-то прошёл сдвиг в updateScrollSelectedItem и записи переменной на -1):

**void** **Settings5View::scrollSensorSpeedNewUpdateCenterItem**(ScrollSelectedItemContainer& item, int16\_t itemIndex)

{

**switch** (itemIndex)

{

**case** 0:

item.updateScrollSelectedItem(2400);

SetSpeed = 115200;

**break**;

**case** 1:

item.updateScrollSelectedItem(4800);

SetSpeed = 2400;

**break**;

**case** 2:

item.updateScrollSelectedItem(9600);

SetSpeed = 4800;

**break**;

**case** 3:

item.updateScrollSelectedItem(19200);

SetSpeed = 9600;

**break**;

**case** 4:

item.updateScrollSelectedItem(38400);

SetSpeed = 19200;

**break**;

**case** 5:

item.updateScrollSelectedItem(115200);

SetSpeed = 38400;

**break**;

}

}

1. В файле .hpp экрана:

**virtual** **void** **scrollSensorSpeedNewUpdateItem**(ScrollItemContainer& item, int16\_t itemIndex);

**virtual** **void** **scrollSensorSpeedNewUpdateCenterItem**(ScrollSelectedItemContainer& item, int16\_t itemIndex);

## Вызов функции С++ из С

Если Вы хотите вызвать функцию - члена класса (включая функции virtual) из C, то понадобится реализовать простую обертку. Например:

// Код на языке C++:

**class** **C**

{

// ...

**virtual** **double** f(**int**);

};

// Функция-обертка над функцией f(int):

**extern** "C" **double** call\_C\_f(C\* p, **int** i)

{

**return** p->f(i);

}

Теперь функция C::f() может быть вызвана следующим образом:

// Код на языке C:

**double** **call\_C\_f**(**struct** C\* p, **int** i);

**void** **ccc**(**struct** C\* p, **int** i)

{

**double** d = call\_C\_f(p,i);

// ...

}

# MySQL

## Базовые настройки сервера и базы данных

TCP/IP соединение, Port 3306, X Protocol Port 33060

Root Password: 1234

Users (Name / Password / Role):

Admin / 1234 / DB Admin

+

pip install mysql-connector-python