



life.augmented



TouchGFX AL Development Introduction

Agenda

1 Introduction

2 TouchGFX Abstraction Layer
Introduction

3 Abstraction Layer Architecture

4 TouchGFX Generator

Introduction



life.augmented

Goal of this presentation

- Getting to know the TouchGFX Abstraction Layer and TouchGFX Generator
 - Introduction to TouchGFX AL
 - Abstraction Layer Architecture
 - TouchGFX Generator

Target audience: Developers new to TouchGFX AL development

Further reading

- You will find a lot of help afterwards in the TouchGFX documentation site:

<http://support.touchgfx.com/>

- Slides in this presentation will refer to relevant documentation pages. Links will be in the lower right-hand corner of the slides
- A good place to start reading following this presentation is:

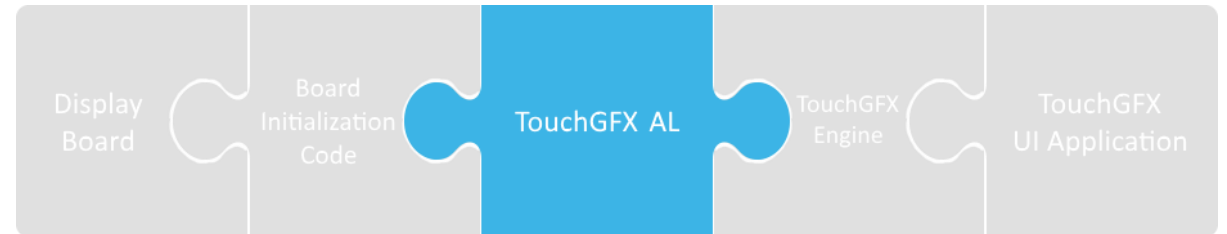
[TouchGFX AL Development - Introduction](#)

TouchGFX AL Development

Main Activities:



Main components:



In this presentation we will focus on the TouchGFX AL development activity and the TouchGFX Generator development workflow

Abstraction Layer Introduction

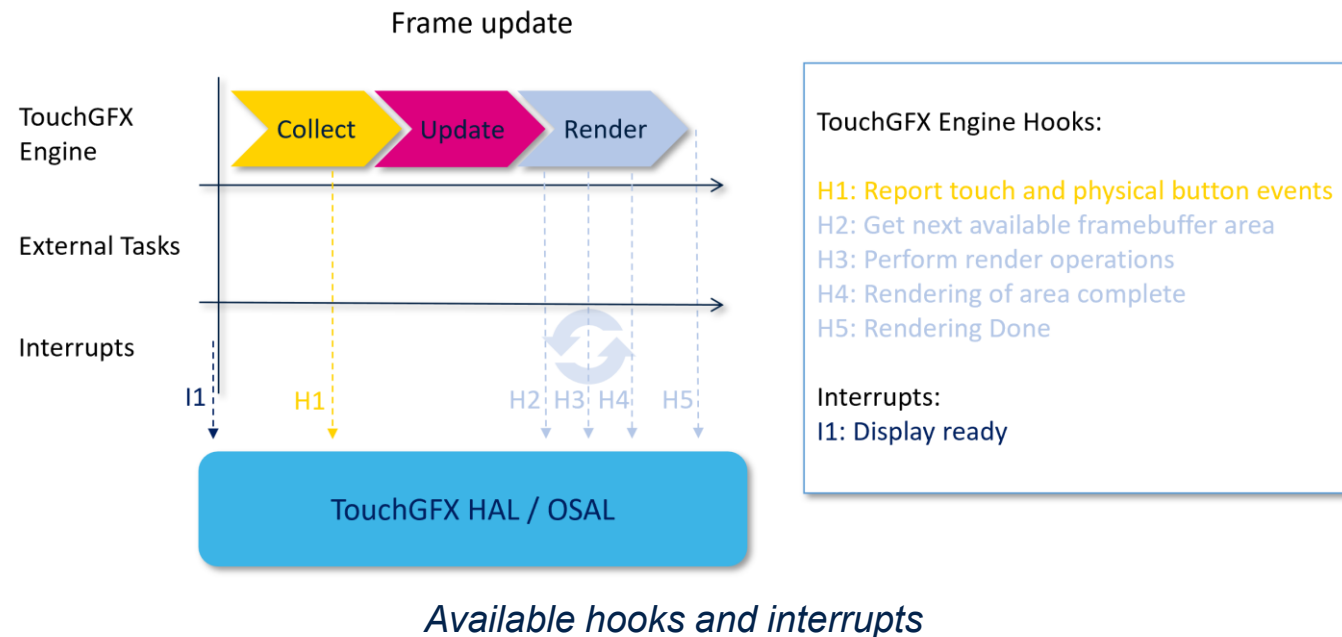
Responsibilities of the TouchGFX Engine

A TouchGFX Abstraction Layer consists of:

- Hardware Abstraction Layer (HAL)
- Operating System Abstraction Layer (OSAL)

Main responsibilities of the Engine:

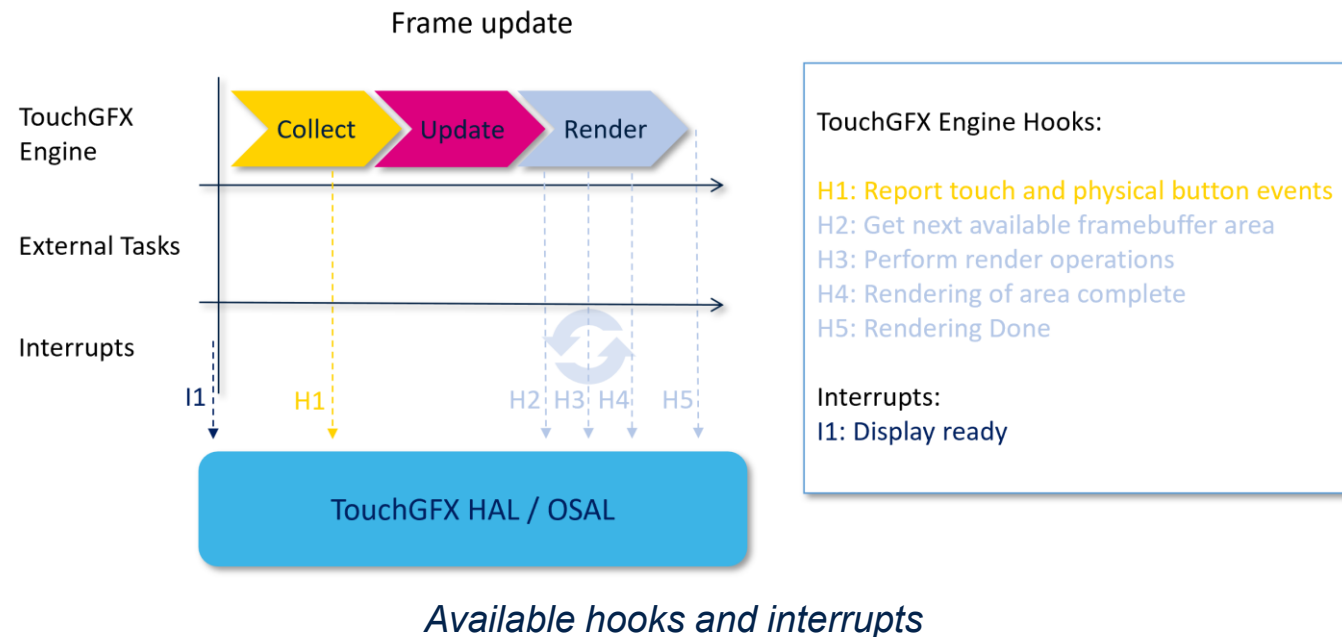
- Collect inputs
- Update the Scene Model
- Render to the framebuffer



Responsibilities of the Abstraction Layer

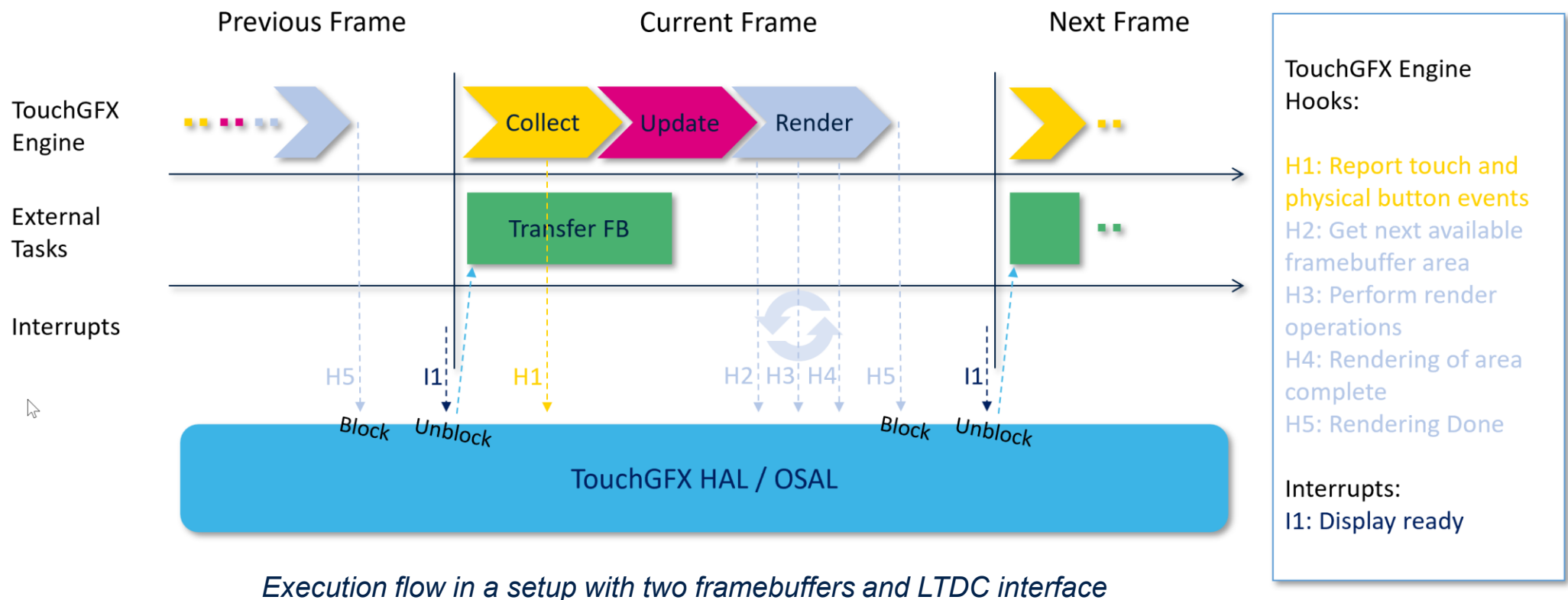
Responsibilities of the AL:

- Synchronize TouchGFX Engine main loop with display transfer
- Report touch and physical button events
- Synchronize framebuffer access
- Report the next available framebuffer area
- Perform render operations
- Handle framebuffer transfer to display



Responsibilities of the Abstraction Layer

Parallel RGB (LTDC) display interface example :



Abstraction Layer Architecture



Abstraction Layer Architecture

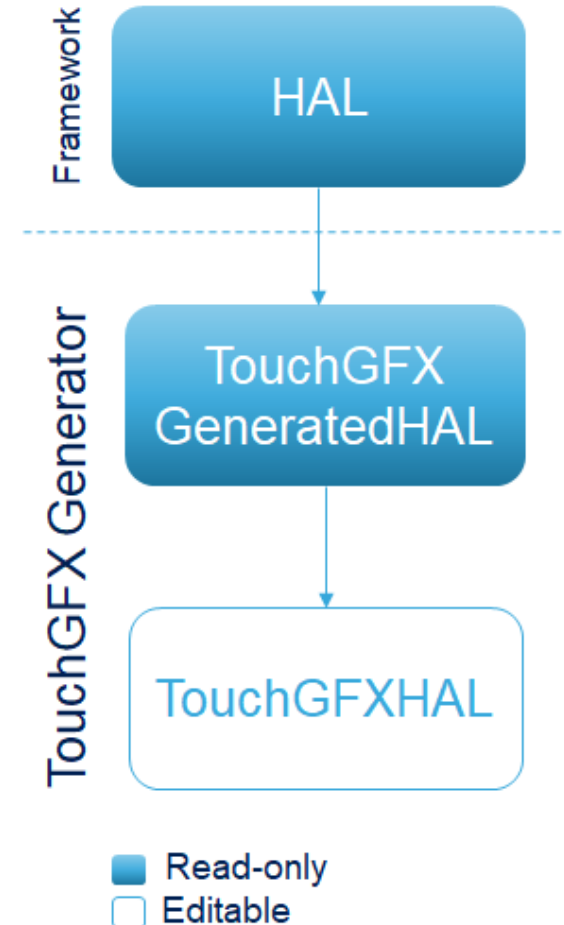
Responsibility reminder

Responsibility	Area	Description
Synchronize TouchGFX Engine Main Loop with display transfer	HAL/OSAL (required)	TouchGFX Engine Main Loop blocks after rendering. Display signals TouchGFX Engine when ready to process new frames.
Report Touch and Physical Events	HAL (optional)	If available, touch- and physical events (e.g. buttons) can be reported to TouchGFX Engine.
Synchronize Framebuffer access	OSAL (required)	Ensures that only one actor accesses a framebuffer.
Report next available Framebuffer area	HAL (required)	Report where TouchGFX Engine should render next. Depends on the framebuffer strategy.
Perform Render Operations	HAL (optional)	Implementation of additional rendering capabilities, e.g. DMA2D.
Handle Framebuffer transfer to display	HAL (required)	Depends on display interface type. For e.g. SPI/FMC developers initiate the transfer manually when asked to.

Responsibilities description table

Abstraction Layer Classes

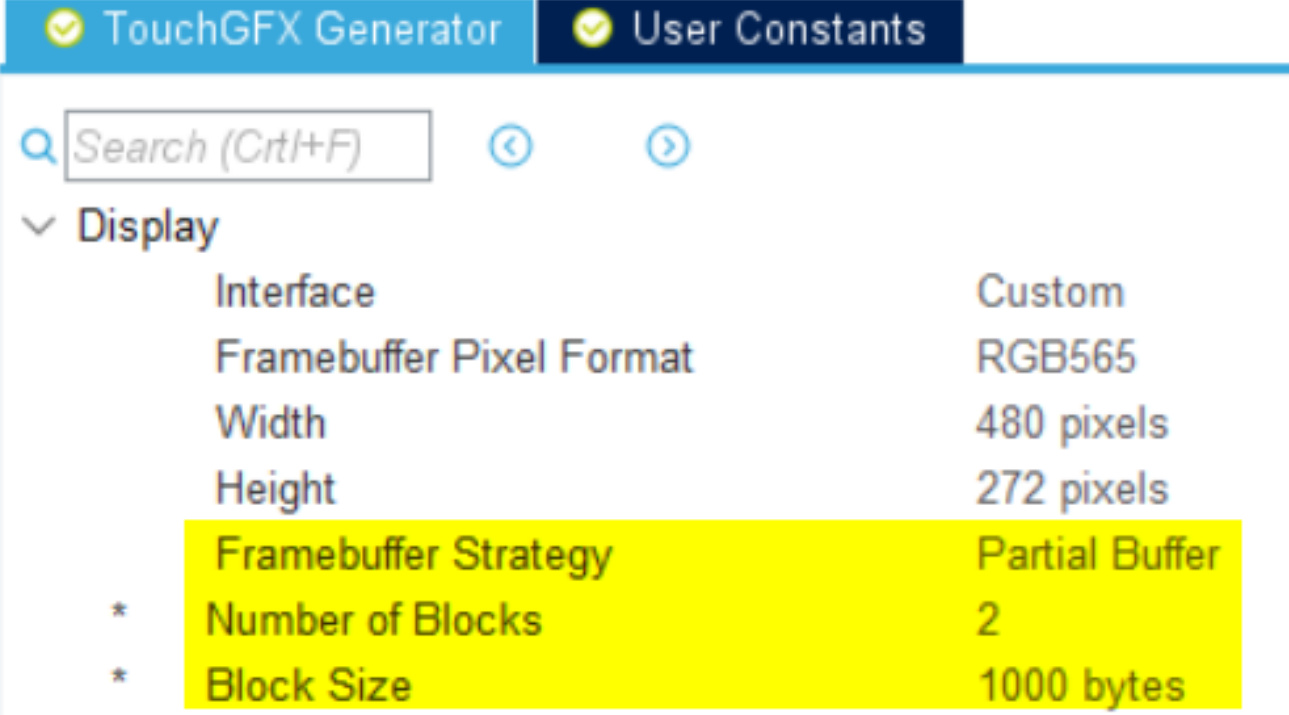
- HAL responsibilities implemented in subclasses of HAL based on STM32CubeMX MCU configuration
- OSAL automatically generated based on STM32CubeMX Middleware configuration
 - CMSIS RTOS V1 or V2
 - Other RTOS to be implement by the developer



Hierarchy of generated code

Advanced Topics

- Development steps documentation
- Scenarios
 - LTDC/Parallel RGB
 - FMC and SPI display interface
 - Framebuffer strategies
- Workshops



The screenshot shows the 'User Constants' tab in the TouchGFX Generator. A search bar at the top contains 'Search (Ctrl+F)'. The 'Display' section is expanded, showing a list of configuration parameters. The following parameters are highlighted in yellow:

Interface	Custom
Framebuffer Pixel Format	RGB565
Width	480 pixels
Height	272 pixels
Framebuffer Strategy	Partial Buffer
* Number of Blocks	2
* Block Size	1000 bytes

Partial framebuffer scenario

TouchGFX Generator

TouchGFX Generator

The TouchGFX Abstraction Layer is generated by TouchGFX Generator

- TouchGFX Generator is part of the X-CUBE-TouchGFX package
- When enabled, TouchGFX Generator creates a TouchGFX Abstraction Layer accordingly to the user settings

TouchGFX Generator

Dependencies group

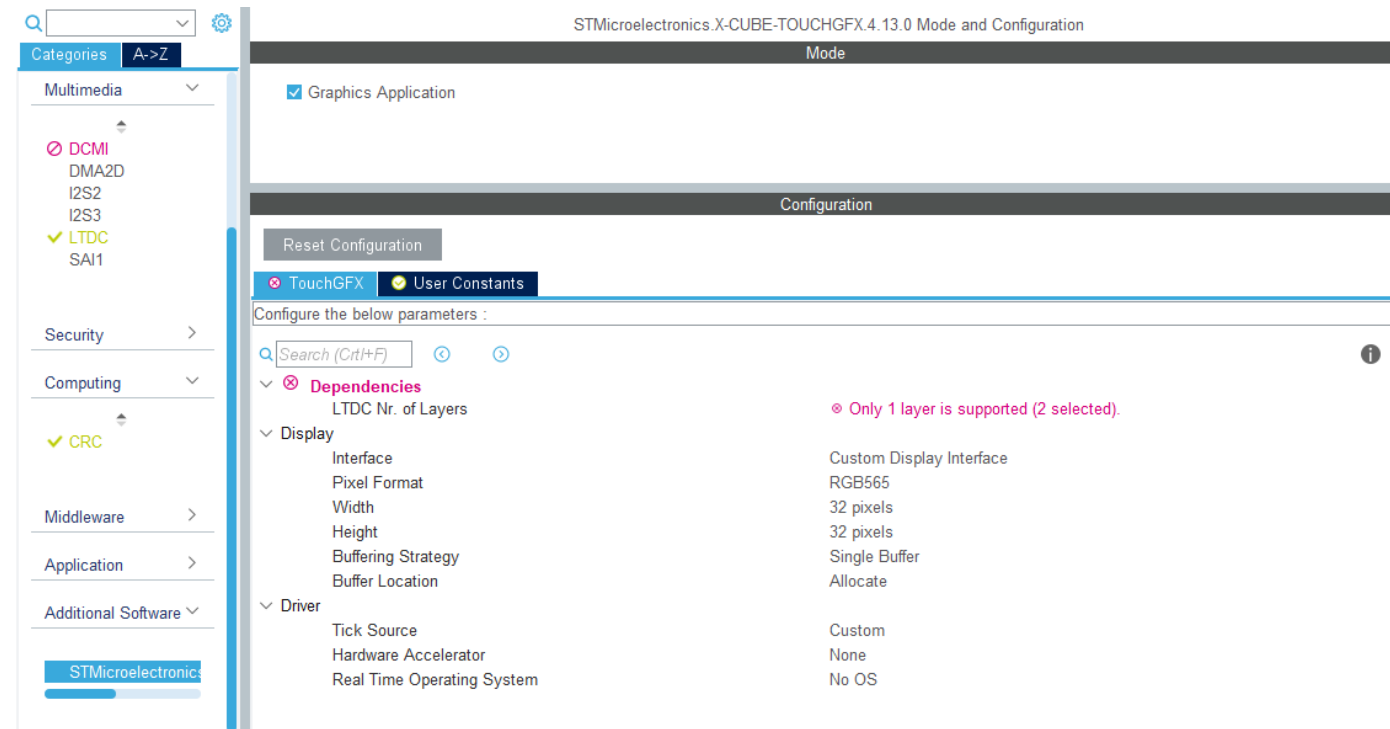
- List of warnings or information based on STM32CubeMX configuration

Display group

- Settings related to interface, format, dimensions and strategy

Driver group

- Settings related to driving and accelerating TouchGFX as well as selecting the OSAL

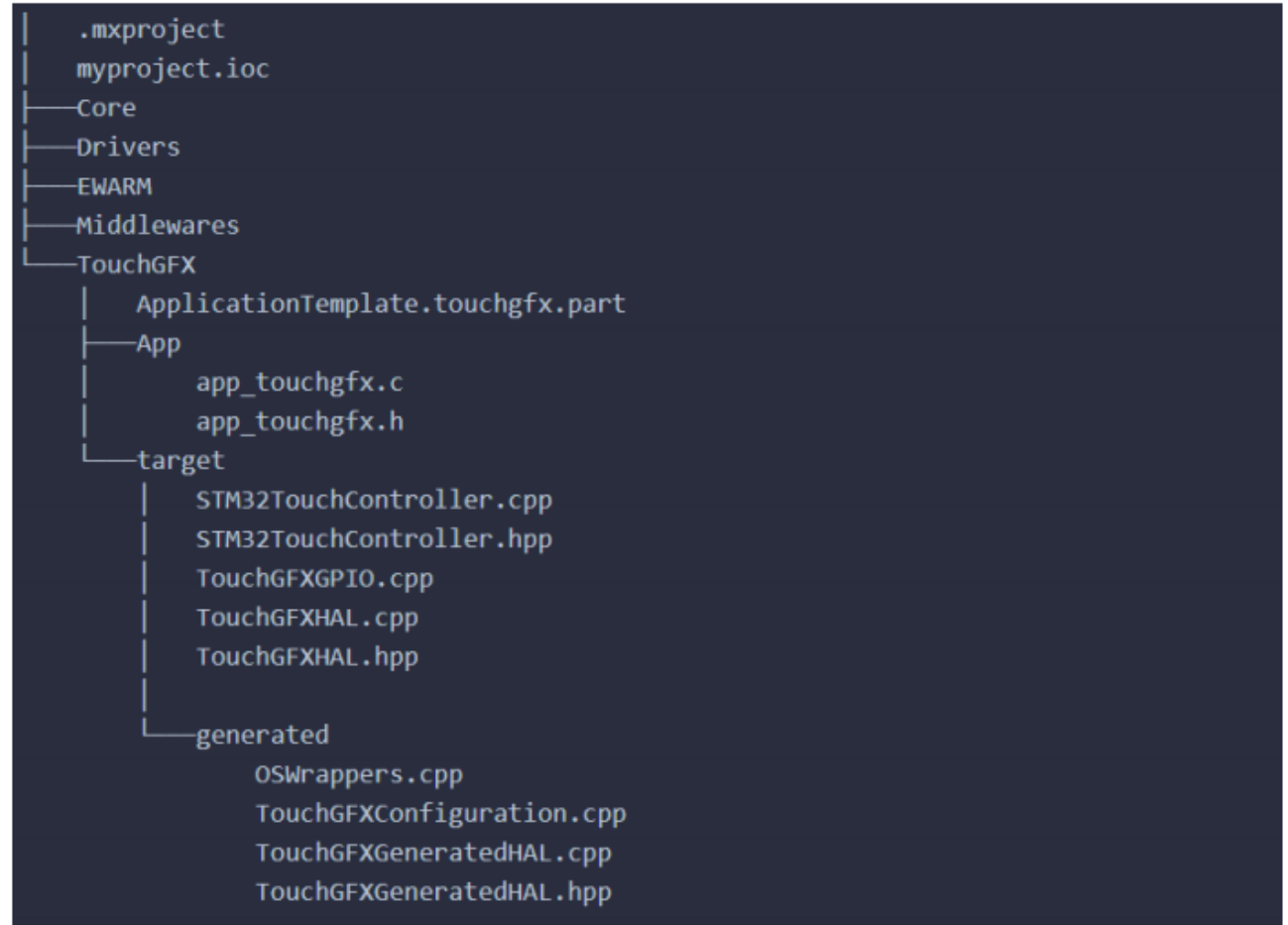


TouchGFX Generator user interface in STM32CubeMX

TouchGFX Generator Code Architecture

The generated Abstraction Layer is created inside the TouchGFX folder when generating from CubeMX

- User code to be implemented in files not under the */target/generated* folder
 - Dedicated sections are highlighted in the code



Generated folder architecture

TouchGFX Generator

- Next step: TouchGFX GUI development in TouchGFX Designer
- Updates to TouchGFX Generator configuration reflected in TouchGFX Designer
 - Pixel format, screen dimensions, ...
- Additional configuration code to be expected for custom hardware platforms
 - TouchGFX Generator configurations in the application templates of ST development kits can be used as source of inspiration

Further reading

- You will find a lot of help afterwards in the TouchGFX documentation site:

<http://support.touchgfx.com/>

- Slides in this presentation will refer to relevant documentation pages. Links will be in the lower right-hand corner of the slides

- A good place to start reading following this presentation is:

[TouchGFX AL Development – Introduction](#)

- Write your questions in the ST forum “Community” to get help from other users and ST employees

[STMicroelectronics Forum](#)

Thank you

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to www.st.com/trademarks.

All other product or service names are the property of their respective owners.



life.augmented