life.augmented

Optimizing flash memory usage in graphical interfaces

STM32

Flash requirements in GUIs

Graphical applications require nonvolatile memory to store assets, fonts, and GUI code.

These requirements are increasing, thereby increasing the use of flash memory. Typical examples:

- High resolution displays, true color depth (24 bit)
- Attractive UI designs, color schemes
- Large applications / many different screens / graphical elements
- Applications with many languages, several sizes of the same font

However, additional requirements may constrain the use of flash memory for running graphical operations. Such as:

- The availability of external NOR flash but limited space
- The use of only internal flash to save BOM or to achieve a simpler PCB
- The need to limit total memory usage to save programming time/cost

Bitmap (BMP) compression

- **L8** image format (max 256 colors per image)
- Compression of L8 images (**L8 RLE** Run-length encoding)
- Compression of RGB565 / ARGB8888 (**C-RGB**)

Bitmap graphics–tricks

- Tiled images (one full image build of copies of one small tile)

How to optimize flash usage in GFX applications?

Using vector-based graphics (geometric shapes)

Vector shapes (box etc.)
SVG (scalable vector graphics image format)
Vector fonts (**VG fonts**)

Suitable assets to optimize flash usage and achieve good performance

- Only one layer vector graphic
- Simple SVGs (SVG tiny 1.2)
- Assets that can be replicated as shapes (boxes, circles.)

Supported with hardware acceleration in STM32 MCUs
(ChromART, NeoChrom GPU)

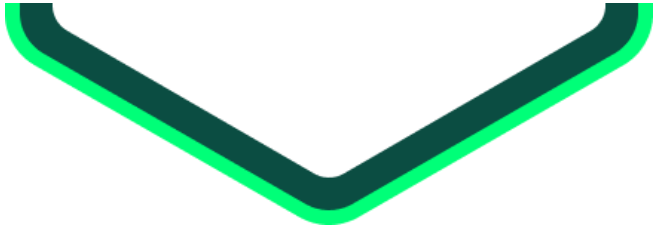
+

All build-in features in TouchGFX software

life.augmented

Examples: how to save flash usage

Button + icon 122 x 112 px		
BMP:	75.4 KB	*100%*
L8:	19.3 KB	*25.6%*
L8 RLE:	2.55 KB	*3.4%*
C-RGB:	4.53 KB	*6.0%*
SVG:	3.01 KB	*4.1%*



Gauge part 150 x 436 px		
BMP:	231.6 KB	*100%*
L8:	65.6 KB	*25.1%*
L8 RLE:	4.66 KB	*1.8%*
C-RGB:	10.5 KB	*4.0%*
SVG:	0.686 KB	*0.27%*

All fonts used (all screens of e-bike)		
BMP:	720 KB	*100%*
VG fonts:	190 KB	*26.4%*

Select the best method and achieve significant flash savings

Main introduction video

life.augmented

STM32U5G9J-DK2 demo & technical insights

life.augmented

Watch the full demo video

One-chip solution for graphics

Use TouchGFX built-in features to develop advanced GUIs requiring **low flash usage**

Bitmap compression:
- L8 image format
- Compression of L8

Vector operations:
- SVG
- VG fonts

TouchGFX reference app running on STM32U5G9 proven advanced GUI with only internal memory.

Full project available in TouchGFXDesigner under Board specific demo

Full demo – Savings:
- Bitmap implementation: 10.5 MB (Assets: 10159k, App: 391k)
- Flash-limiting implementation: 800 KB (Assets: 407k, App: 391k)
- **Flash savings: 92%**

E-bike demo

MCU: STM32U5G9

Display: RGB 800x480

- Only internal flash used (4 MB available)
- Only internal RAM used (3 MB available)
- All fonts are vector fonts (21 different sizes)
  - Takes up 190 KB of flash memory
    - Would be 720 KB as bitmap fonts
- All images are compressed in L8 RLE format
  - 166 KB of compressed images
  - The assets are compatible (max 64 colors)
  - Cached when needed in scalable or rotatable widgets (for example, Scalable Image)
- SVG
  - 52 KB of SVGs
  - Used when assets are needed in several sizes (scalability)
- 4 languages supported
  - English, Chinese, Japanese, Arabic
- Performance: 60-30 FPS
  - Render time will be longer when changing theme, since the complete screen requires rerendering

Memory details
NB: All internal MCU

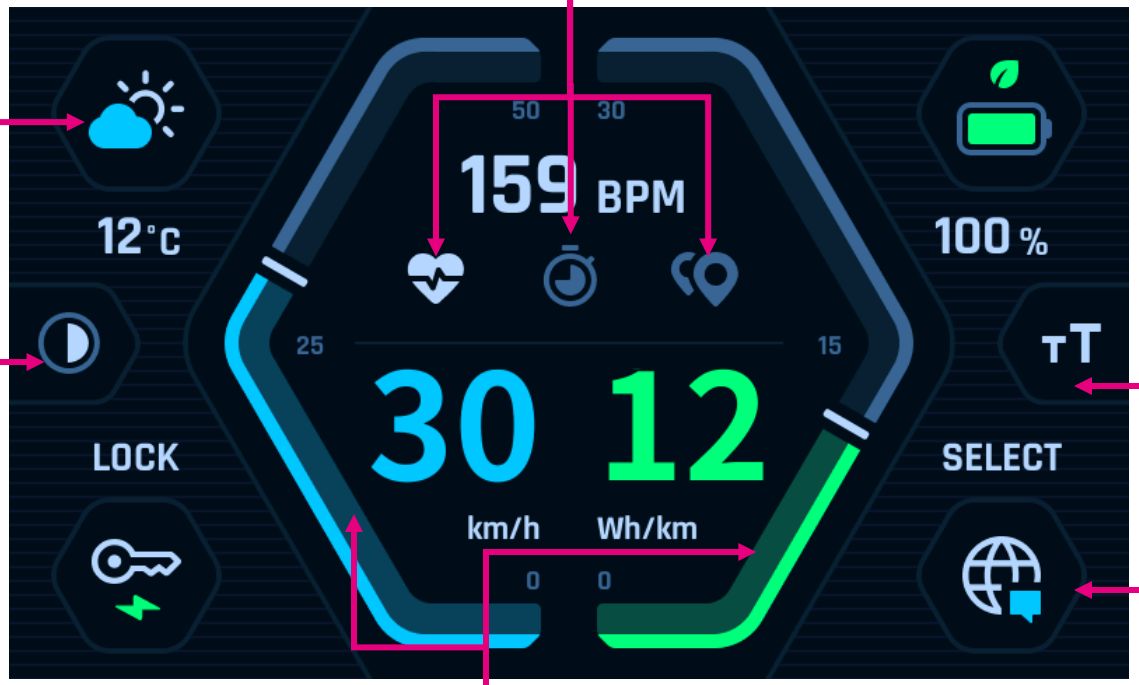
Description	*Used (KB)*
Flash memory	**798**
Demo application	391
Assets (images, SVG, fonts)	407
RAM	**2992**
Demo application	146
16-bit framebuffers	1536
Stencil buffer	389
Bitmap cache	921

Dashboard screen

Choose BPM, time or distance
The icons are L8 RLE

Buttons and icons are L8 RLE

2 colors (light/dark)



60 FPS but this result decreases when the gauge turns to orange, the theme is changed, or the language selector is extracted.

3 sizes

4 languages

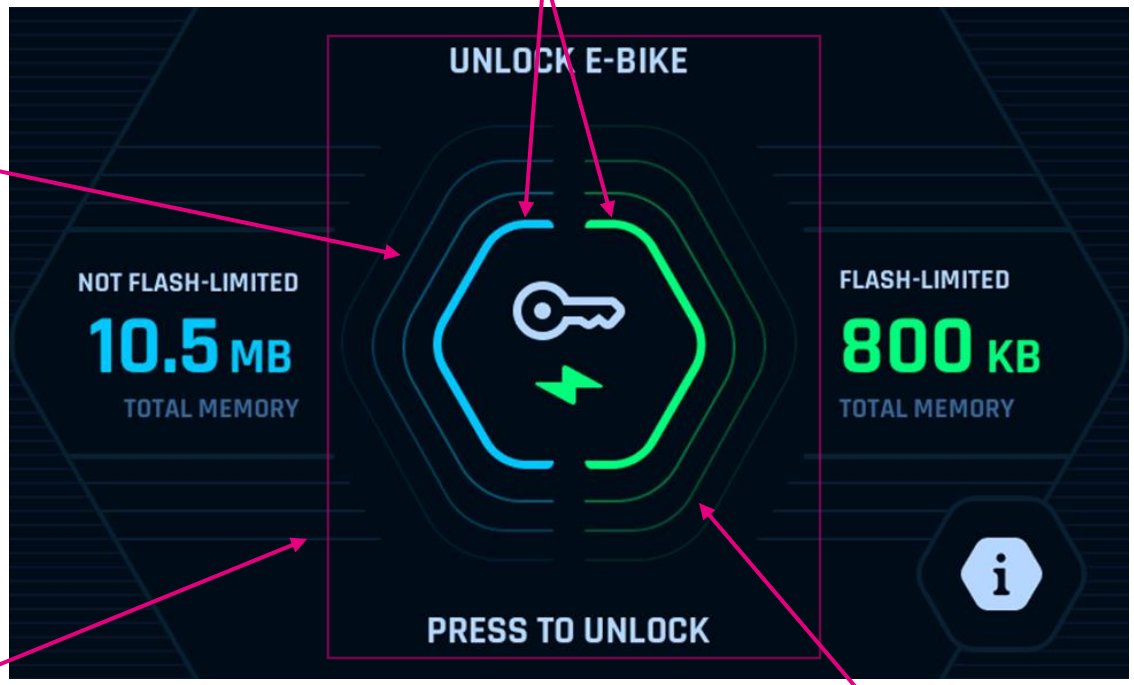
Mainly SVG gauges
(However, the vertical foreground gauges are L8 RLE – simpler to write to RAM)

Start screen

Scalable image
Normal image when done scaling
Decompressed to RAM to scale

60 FPS
Only 1 color (dark)

Ripples are compressed bitmaps



Go to the dashboard screen by pressing in this area

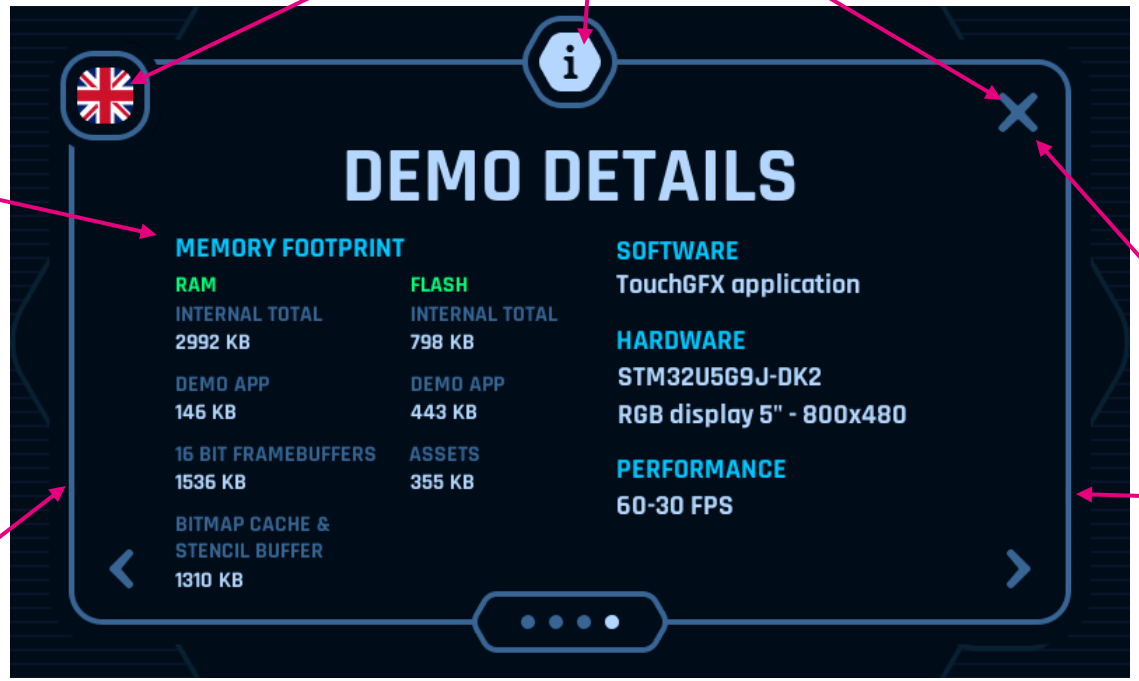
Ripples animate when idle
Animation done with alpha (FadeAnimator) and small movement (MoveAnimator)

Info boxes

All icons are RLE compressed images

60-30 FPS
Only available in dark theme

The content is partly cached to improve performance



Close to get back to start screen

Box built of SVG corners and boxes

SwipeContainer

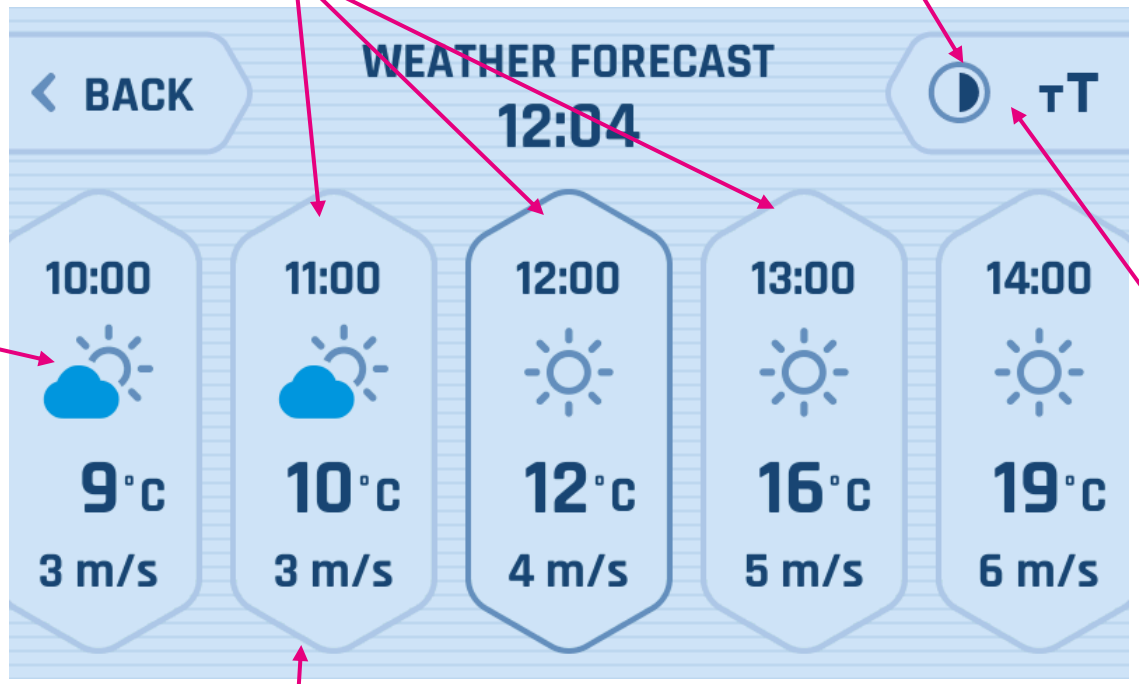
Weather screen

Weather elements are cached to RAM to improve performance

2 color themes (light/dark)

60-30 FPS when scrolling (Depends on scroll speed)

Vector icons
Just saved once and then scaled to fit all 3 sizes



3 sizes

Scroll Wheel to scroll content

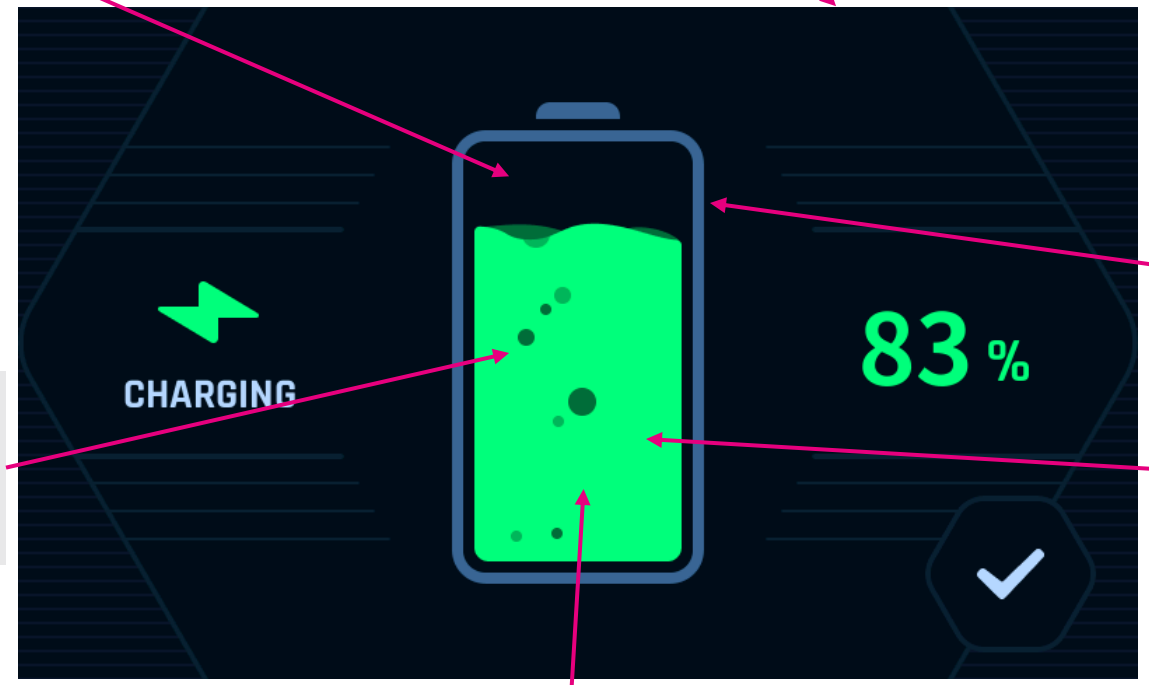
Vector boxes
Just saved once and then scaled to fit all 3 sizes

Battery screen

Wave is RLE compressed animated images

2 colors (light/dark)
Cannot be changed inside the screen

60 FPS



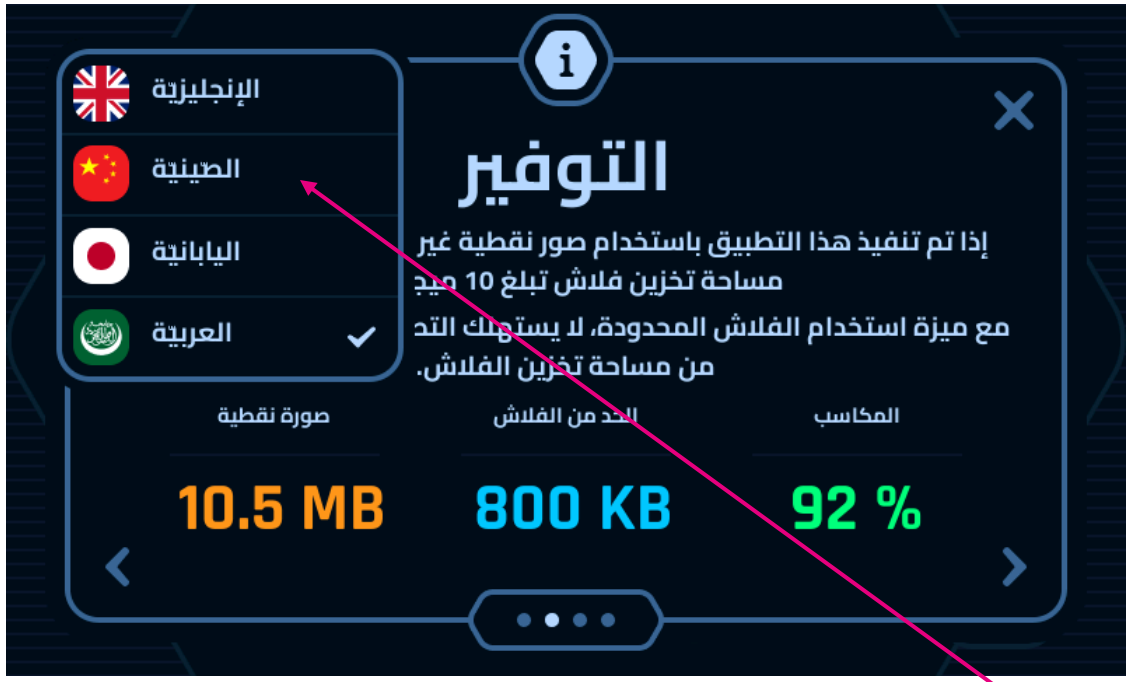
The frame is SVG

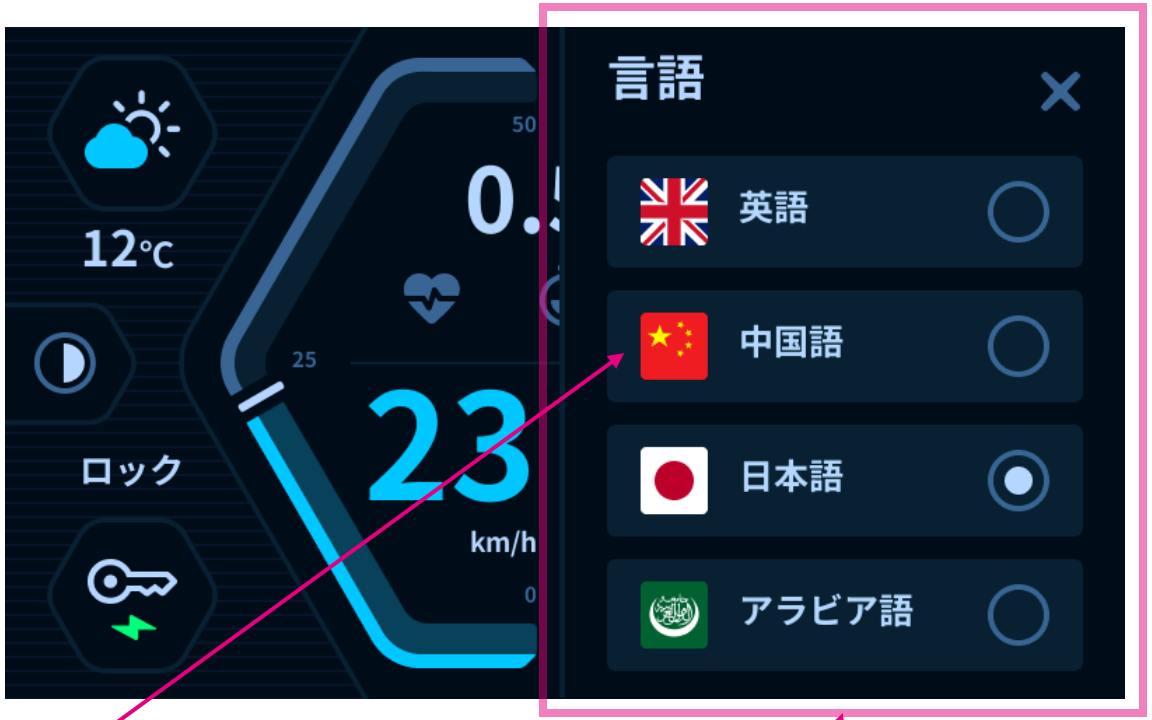
Bubbles are animated with MoveAnimators

The solid filled part is the box widget

Bubbles are individual shapes (can change color)

Languages and language selector





4 languages supported in the complete demo

Language can be changed from both the dashboard screens and the information boxes on the start screen

Language selector is cached to RAM to improve performance (60-30 FPS)

Conclusion
compression and vector graphics

Image compression

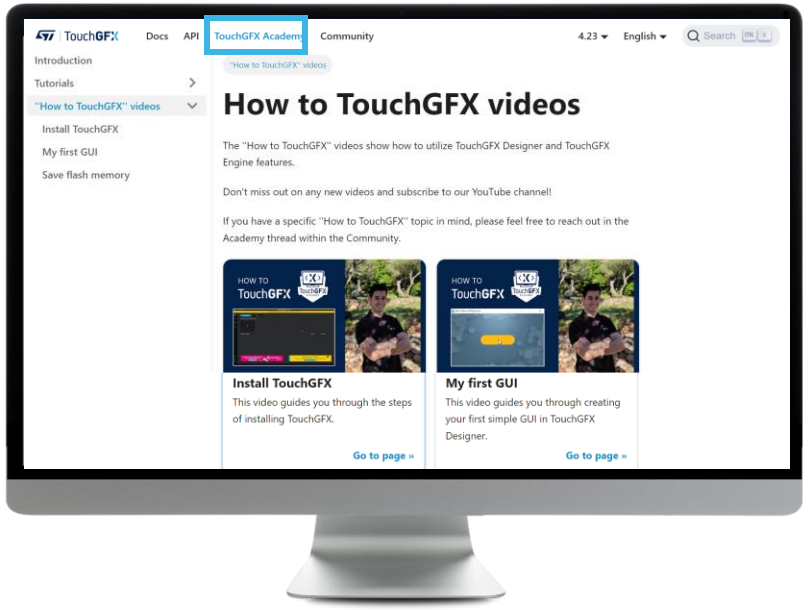
- Significant flash savings
  - HIGH compression rates – up to 90 to 98%
  - Less compression rate for complex images (many colors, gradients)
  - Use L8 + compression when possible (max 256 colors per image)
  - Easy and individual selection of images and type of compression.
- Lossless
- Can apply to all STM32 MCUs
- Not direct applicable for rotating and scaling widgets (caching required)
- Minor performance penalty
  - Optimized implementations secure minimum penalty. Cache of 512 bytes to utilize ChromART
  - Longer render time, depending on image complexity, MCU speed, flash access, hardware acceleration.
  - NB: performance gain in some cases due to lower data volume.

Vector graphics

- Significant flash savings when the application embeds:
  - Fonts in multiple sizes, and large fonts
  - Simple images used in several sizes, image scaling
  - Large simple images
- Can apply to STM32 MCUs with VG hardware acceleration and/or sufficient MCU resources (MHz), STM32U5F, STM32H7, STM32H5
- Significant performance penalty
  - Longer rendering time, depending on available resources (hardware acceleration and clock speed).

Learning center: TouchGFX Academy

NEW



The place to go to explore TouchGFX features and functionalities, showcased through practical examples.

How to TouchGFX: save flash memory on your STM32 GUI application

On documentation

YouTube

life.augmented

Our technology starts with You

Find out more at www.st.com

life.augmented