Embedded graphics on STM32F4

V0.1

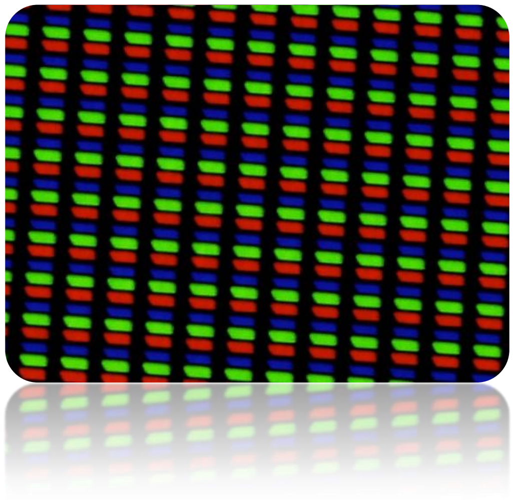
life.augmented

SILICA An Avnet Company

WWW.EMCU.IT

OBJECTIVES

- Getting familiar with basics of graphics
  - General LCD connection
  - Color representation
  - Layers
  - Transparency / alpha channels
  - CLUT
  - Color keying

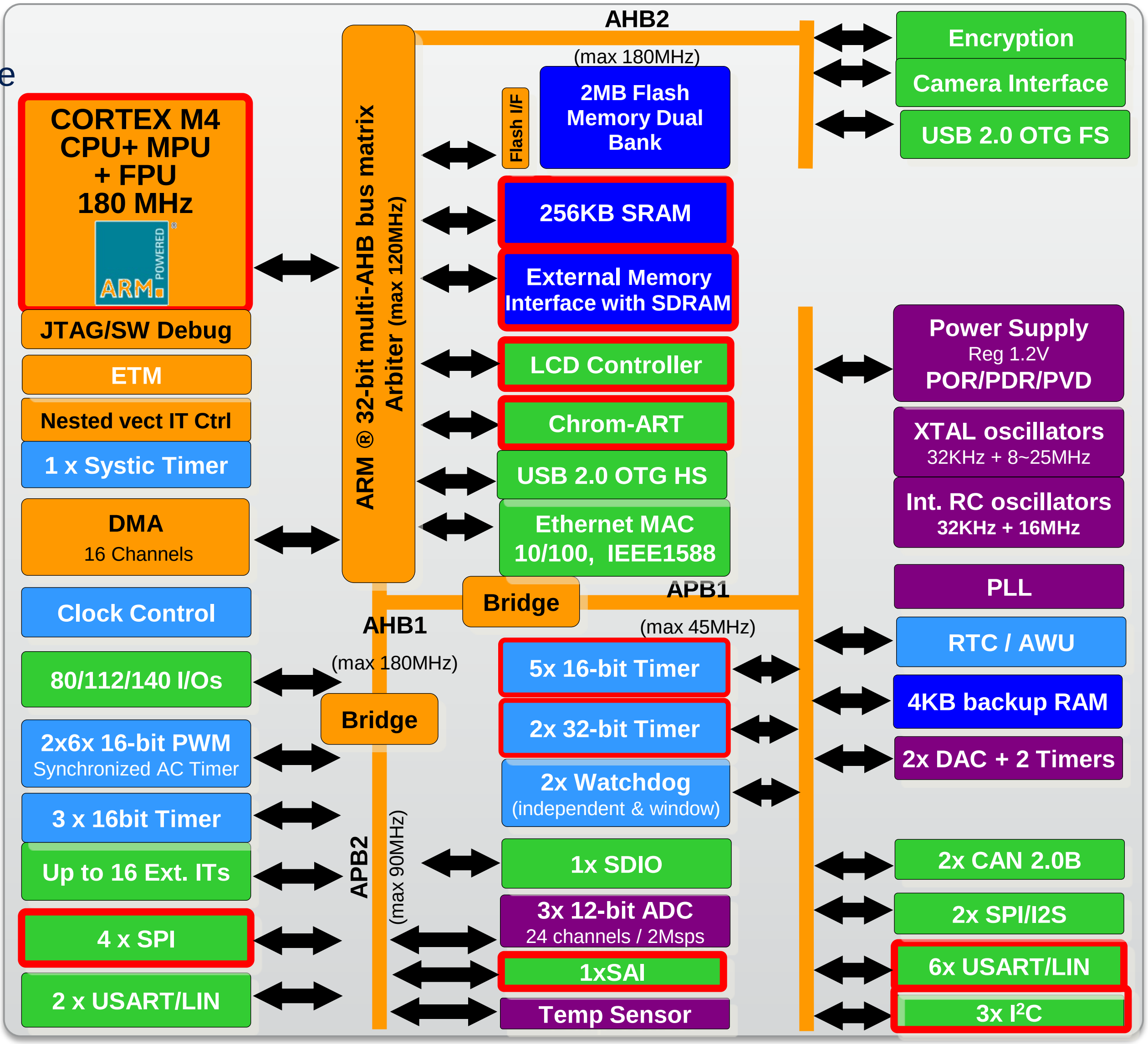


- Understand how you can benefit from STM32F4's HW acceleration
  - Usage of LTDC layer features
  - Offload CPU by using Chrom-ART
  - Hardware pixel format conversion

STM32F4x9 Block Diagram

- Fully compatible with F4x
- Up to 180MHz with over-drive mode
- Dual Bank 2 x 1MB Flash
- 256KB SRAM
- FMC with SDRAM + Support and 32-bit data
- Audio PLL + Serial Audio I/F
- LCD-TFT Controller
- Chrom – ART Accelerator
- Hash: supporting SHA-2 and GCM
- More serial com and more fast timers running at Fcpu
- 100 pins to 208 pins
- 1.71V-3.6V Supply

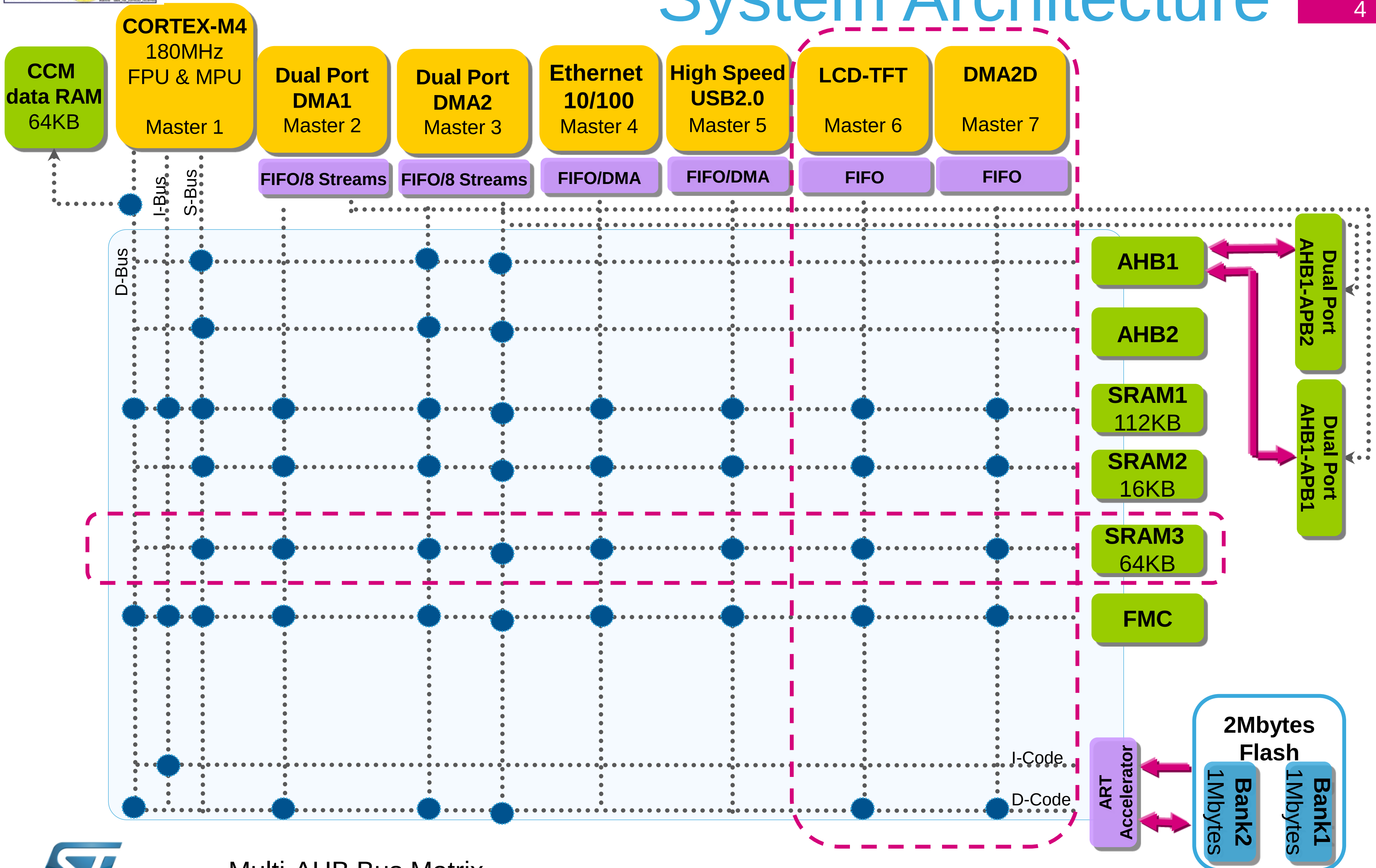
New IPs/Features/more IPs instances



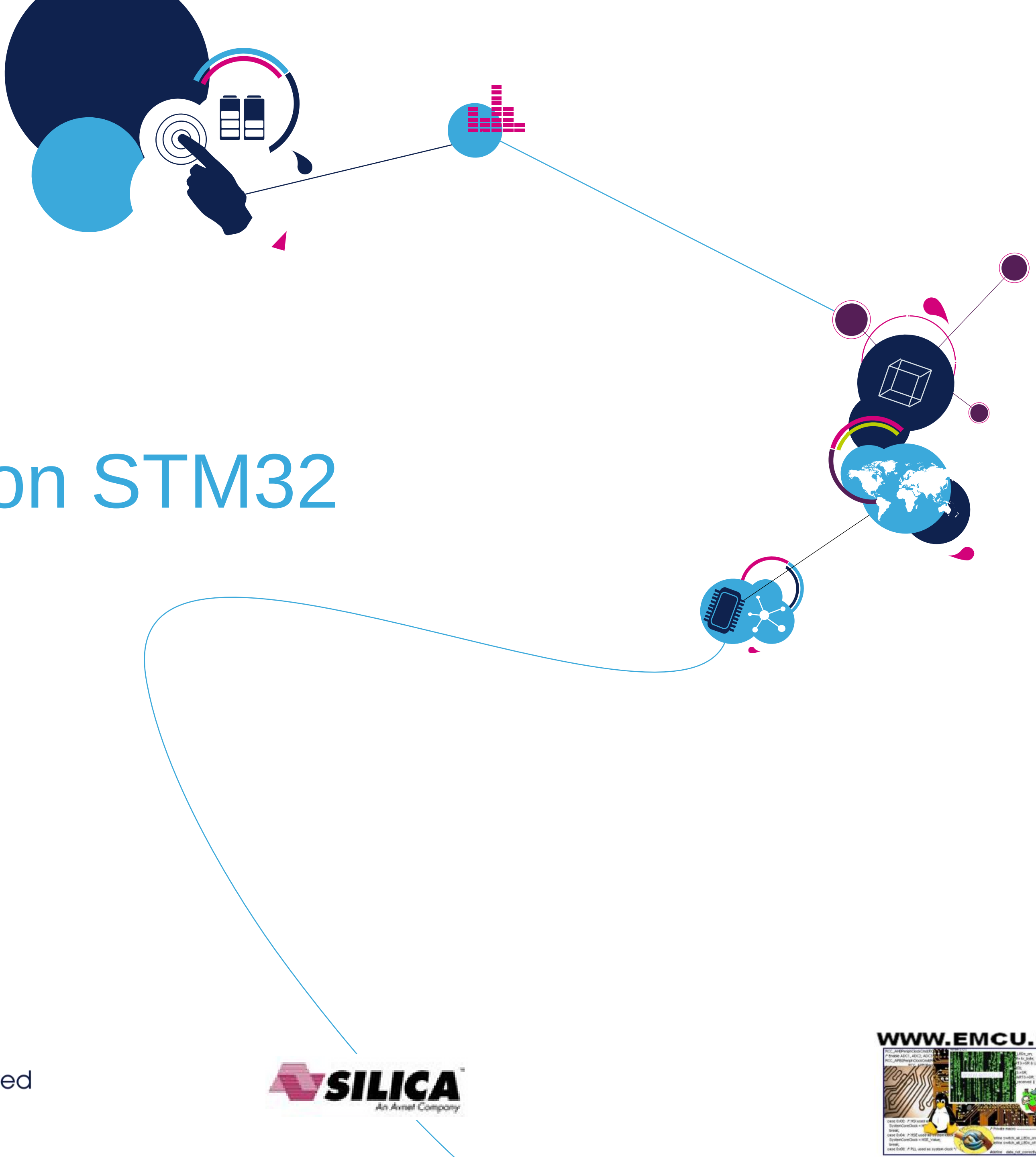
life.augmented

SILICA An Avnet Company

System Architecture



Multi-AHB Bus Matrix



Graphics on STM32

MCD Application Team

life.augmented

SILICA An Avnet Company

WWW.EMCU.IT

Image quality needs

- iPhone has completely changed the required level of graphical quality
  - Bigger display size
  - Better display resolution
  - Perfect icons
  - State of the art animations (coverflow…etc…)
- User wants the same experience than on iPhone
  - Huge resource needs on microcontroller side
  - Hardware acceleration becomes mandatory
  - Content design/creation is a key aspect
- But no need for a gaming machine
  - No need for real 3D : **2D graphics** is wide enough

Overview

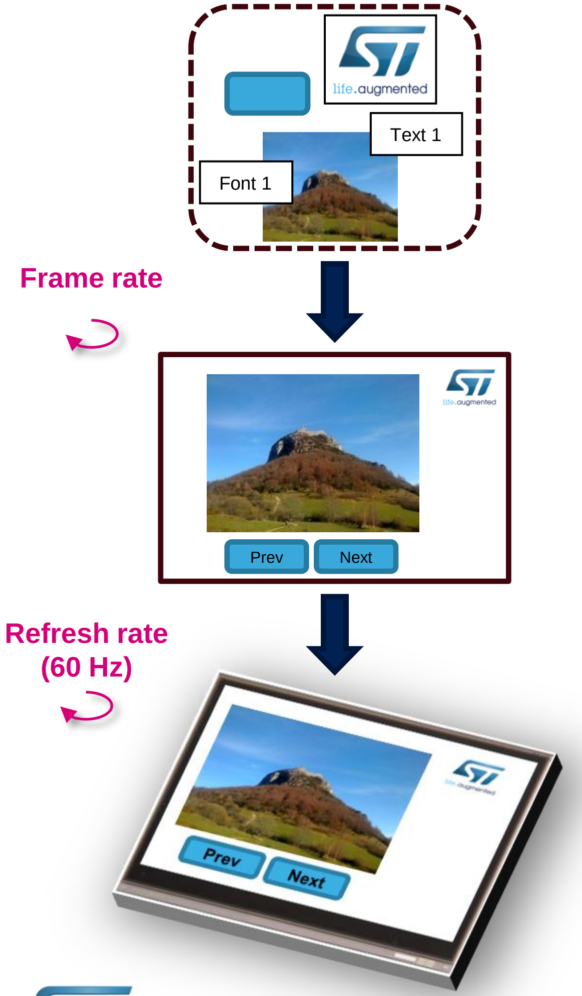
- **Past / Today**
  - Keyboard and/or Knob
  - Optional 7-segment display
  - Optional dot-matrix display (text or graphical)
  - Limited interactivity

- **Next generation**
  - **One TFT color display + touch sensing (virtual keyboard/knob)**
  - **More interactivity**
    - Integrated manual
    - Flash like animations for “how to”
    - Explicit diagnostic
    - …etc…

Typical embedded display needs

- **For home appliance**
  - Display size and form factor will be very different from an equipment to another
  - In most of them we shall not be above **6~7 inches**
  - **iPad 2** display is **132PPI** (pixels per inch)
  - **7" 4:3 display 800x600** represents **@142PPI**

From ones and zeros to image display



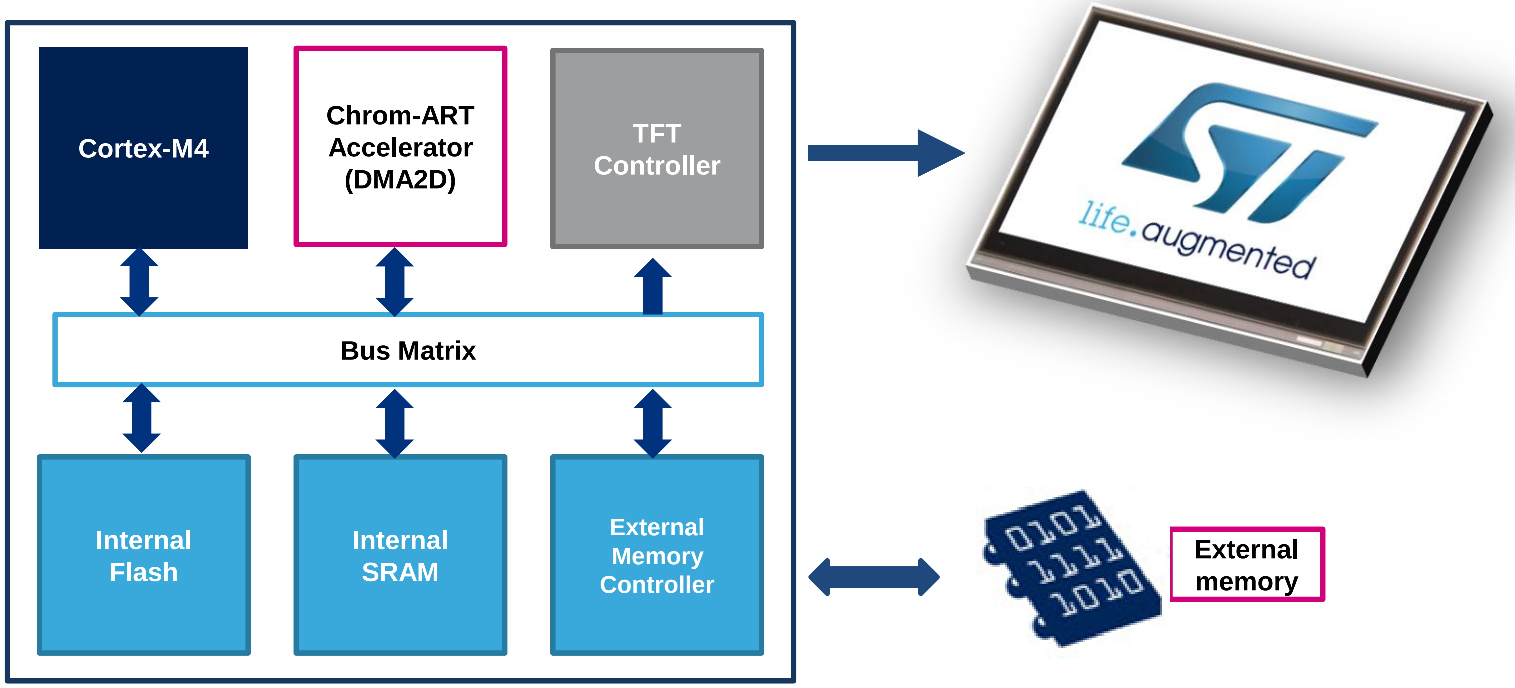
Step 1 : Create the content in a frame buffer

- The frame buffer is build by composing graphical primitive
- This operation is done by the **CPU** running a graphical library software
- It can be **accelerated** by a **dedicated hardware** used with the CPU through the graphical library
- More often the frame buffer can be updated, more fluent will be the animations (frames per second / fps)

Step 2 : Display the frame buffer

- The frame buffer is transferred to the display through a dedicated hardware interface
- Graphical oriented microcontroller are offering a **TFT controller** to drive directly the display
- The frame buffer must be sent at 60fps to have a perfect image color and stability (independently from the animation fps)

STM32F4x9 Architecture



- **TFT controller** allows the interfacing
- **Chrome-ART (**DMA2D) provides a **true HW graphical acceleration**
- **DMA2D offloads the CPU** for operations like rectangle filling, rectangle copy (with or without pixel format conversion), and image blending
- **DMA2D goes faster than the CPU** for the equivalent operation

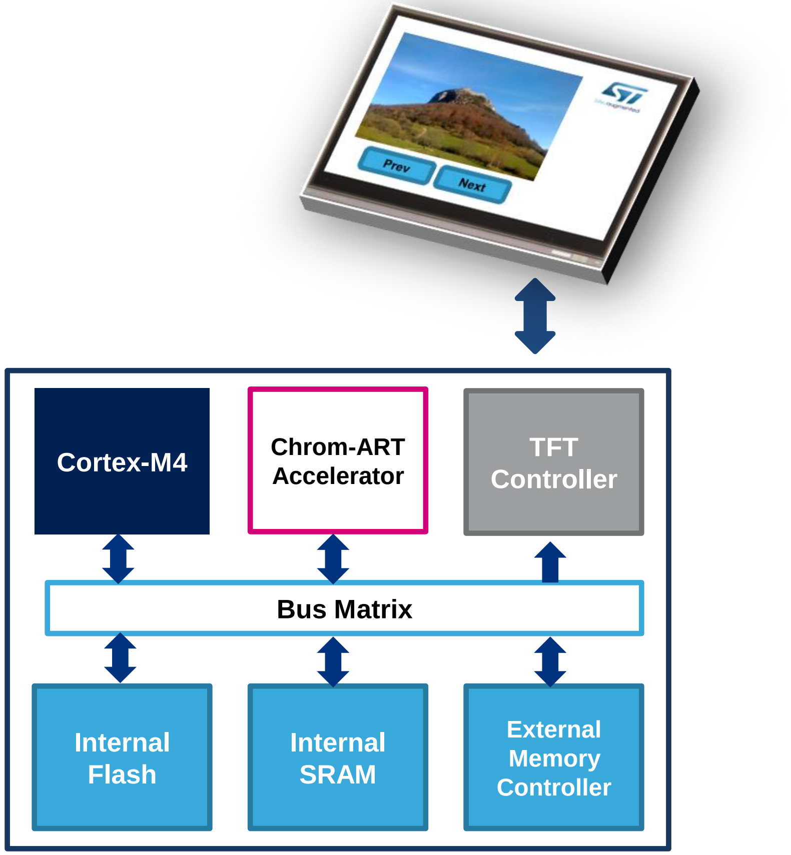
Implementation examples and resources requirements

Single Chip MCU

- Internal Flash up to 2MB
- Internal SRAM up to 256KB
  - Frame buffer in internal SRAM
    - 16-bit QVGA (320 x 240 ~154 KB)
    - 8-bit WQVGA (400 x 240 ~97 KB)
- Package **LQFP 100pin**

Cost saving + Graphic Acceleration

16-bit QVGA, 8-bit WQVGA

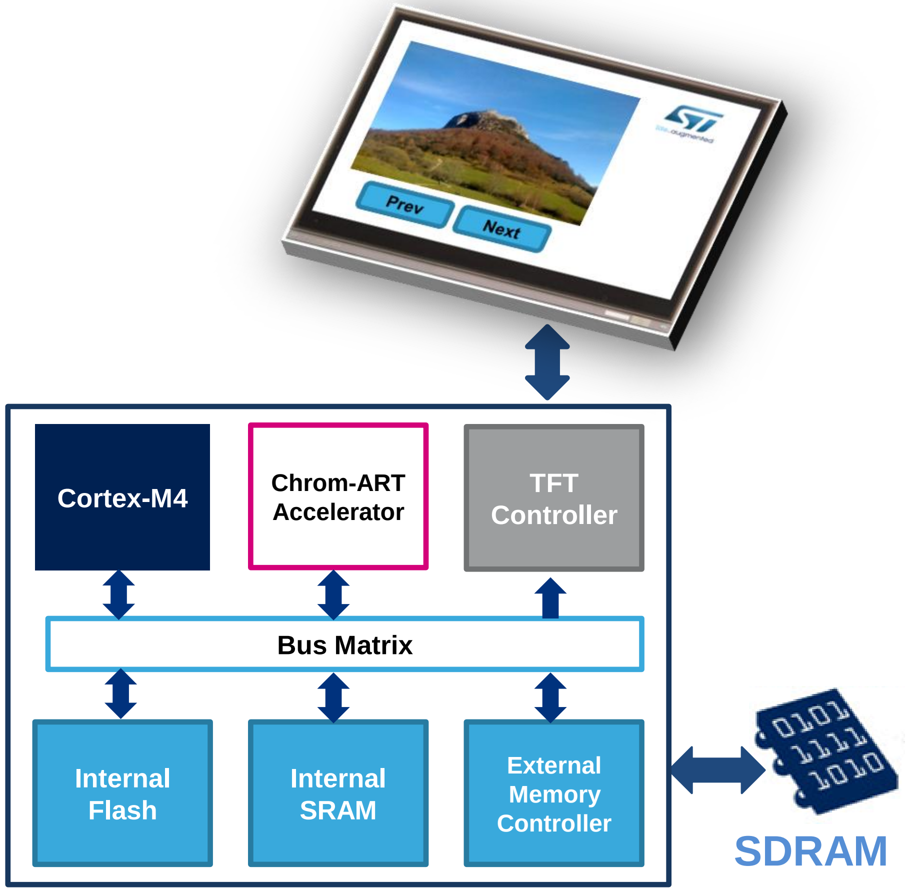


MCU with External Memory

- Internal Flash up to 2MB
- Internal SRAM up to 256KB
- External Memory for frame buffer
  - 16-bit or 32-bit SDRAM / SRAM
- Package: **LQFP 144pin,** up to 208.

Unique Graphical Capability and Flexible architecture

Up to SVGA (800x600)



STM32F4x9

LCD-TFT Display Controller (LTDC)

MCD Application Team

life.augmented

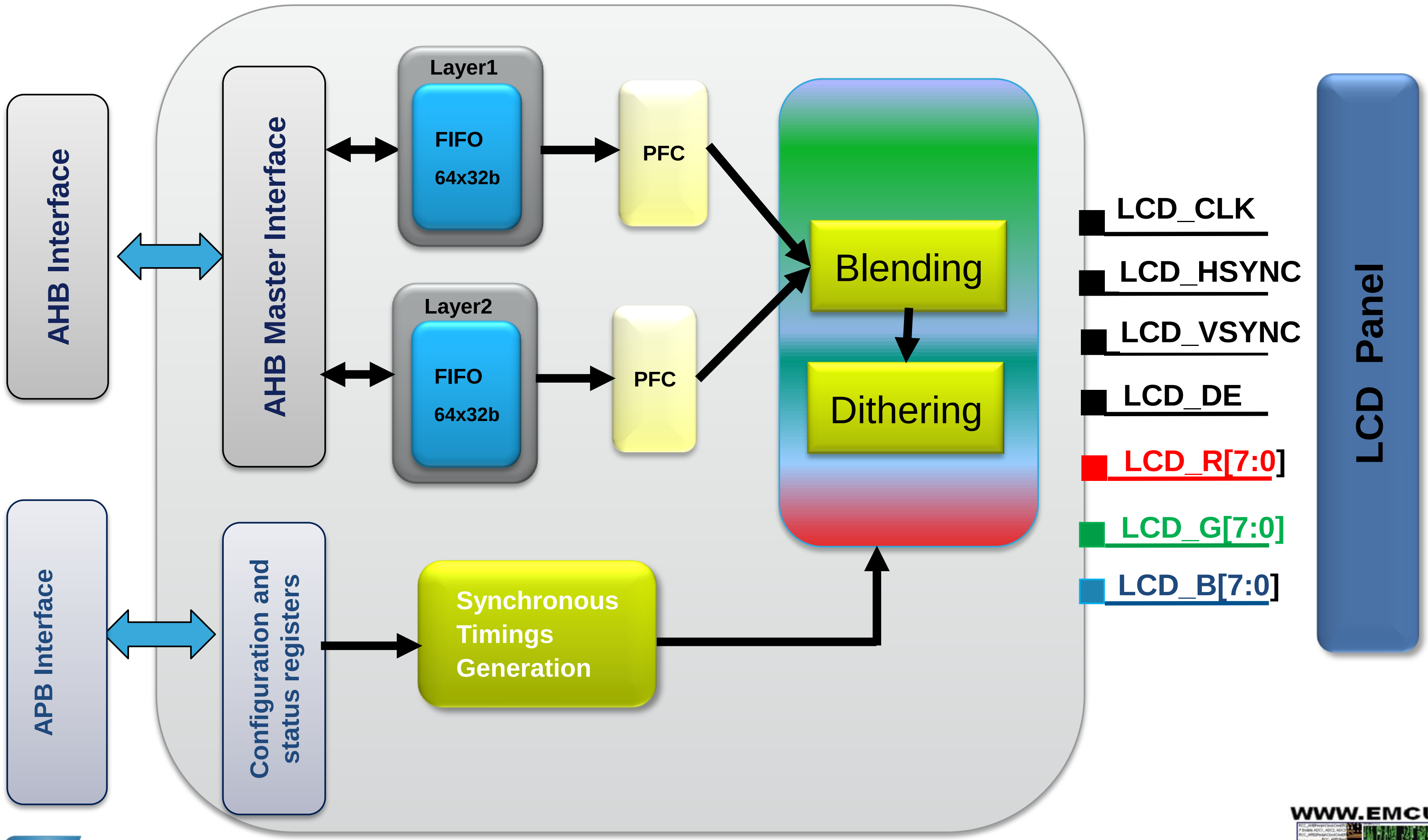
SILICA An Avnet Company

WWW.EMCU.IT

Benefits

- AHB bus Master
  - Can access any part of address space → ext. memory, internal SRAM, Flash.
- Flexible programmable display parameters
  - Display control signals
  - Flexible color format
- Multi-Layer Support
  - Windowing
  - Blending
  - Flexible programmable parameters for each Layer
- On chip memory or External memory can be used as Frame buffer

LCD-TFT architecture



LCD Memory Requirements

- **Frame buffer size**
  - The panel size and bits per pixel determine the amount of memory needed to hold the graphic buffer.
  - Memory Requirement (KB) = (bpp * Width * Height)/8

Panel Resolution	Total Pixel	bpp (bit per pixel)	Required memory (KB)
320x240 (QVGA)	76.8K	16bpp	153.6
		8bpp	76.8
480x272 (WQVGA)	130.5K	16bbp	261.12
640x480 (VGA)	307.2K	16bbp	614.4

- In many cases, more memory is needed. E.g. double buffering: one graphic buffer to store the current image while a second buffer used to prepare the next image.

LCD-TFT Signals

- The LCD-TFT IO pins not used by the application can be used for other purposes.

LCD-TFT Signals	Description
LCD_CLK	Pixel Clock output
LCD_HSYNC	Horizontal Synchronization
LCD_VSYNC	Vertical Synchronization
LCD_DE	Data Enable
LCD_R[7:0]	8-Bits Red data
LCD_G[7:0]	8-Bits Green data
LCD_B[7:0]	8-Bits Blue data

LTDC Main Features - (1/3)

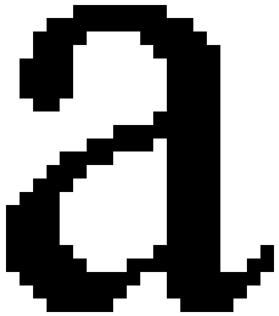
- 24-bit RGB Parallel Pixel Output; 8 bits-per-pixel (RGB888)
- AHB 32-Bit master with burst access of 16 words to any system memory
  - Dedicated FIFO per Layer (depth of 64 word)
- Programmable timings to adapt to targeted display panel.
  - HSYNC width, VSYNC width, VBP, HBP, VFP, HFP
- Programmable Polarity of:
  - HSYNC, VSYNC, Data Enable
  - Pixel clock
- Supports only TFT (no STN)

LTDC Main Features - (2/3)

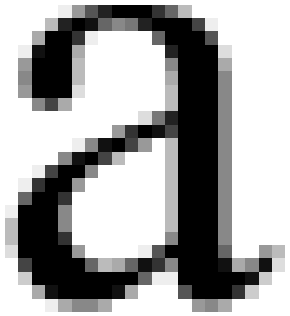
- Programmable display Size
  - Supports up to 800 x 600 (SVGA)
- Programmable Background color
  - 24-bit RGB, used for blending with bottom layer.
- Multi-Layer Support with blending, 2 layers + solid color background
- Dithering (2-bits per color channel (2,2,2 for RGB))
  - Combination of adjacent pixels to simulate the desired shade
  - Dithering is applicable for 18bit color displays, to simulate 24bit colors.
- Newly programmed values can be loaded immediately at run time or during Vertical blanking
- 2 Interrupts generated on 4 event Flags

Alpha Channel Usage

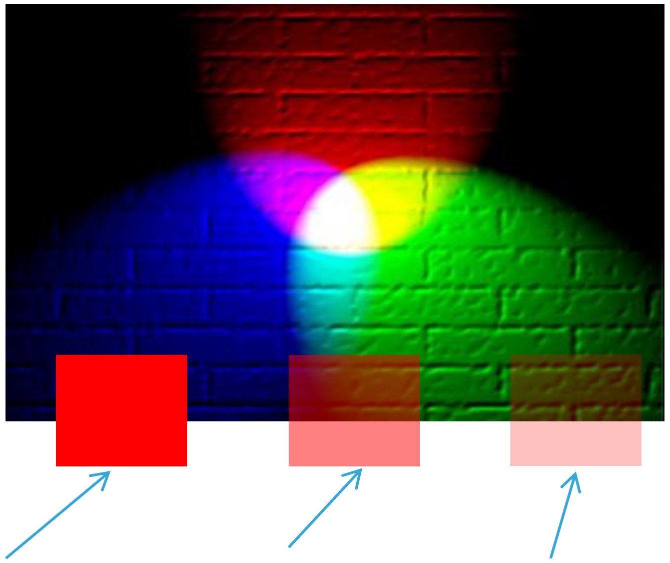
- The Alpha channel represent the transparency of a pixel
- It’s **mandatory** as soon as you are manipulating bitmaps with non square edges or for anti-aliased font support



Aliased



Anti-aliased

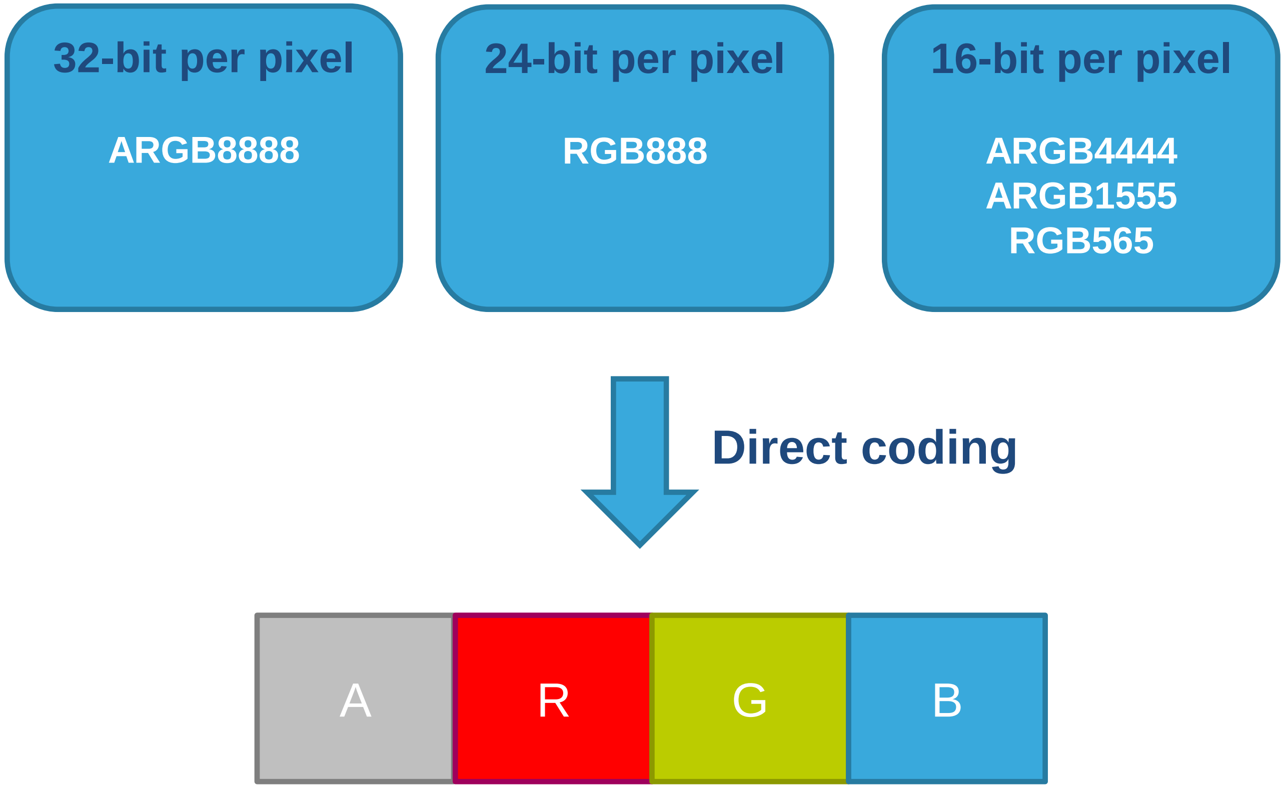


- **0xFF = opaque**
- 0x00 = transparent

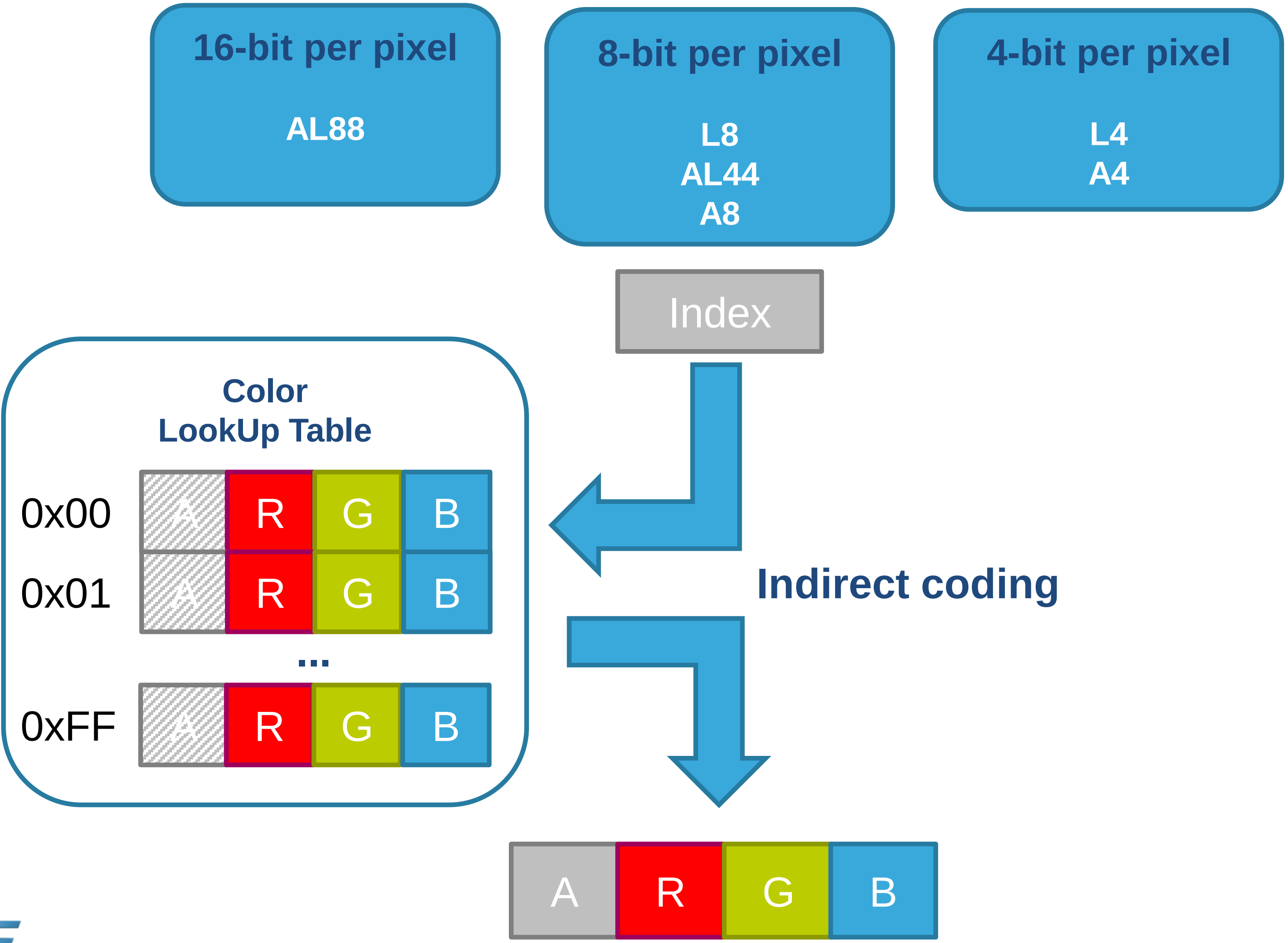
0xFFFF0000 0xAAFF0000 0x30FF0000

Box filled with color in ARGB8888 format

PFC - Direct color Mode



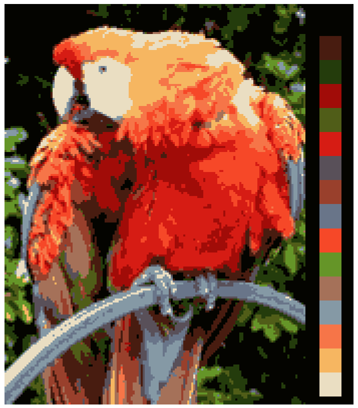
PFC - Indirect color Mode



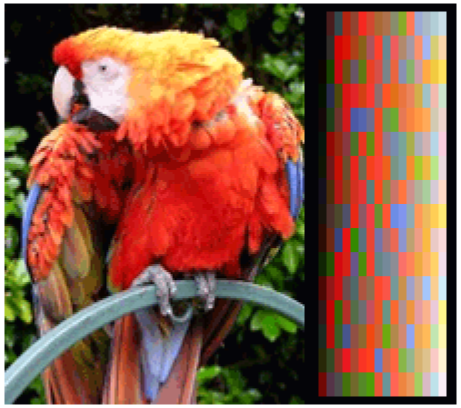
CLUT- Palletized

- The R, G and B values and their own respective address are programmed through the **LTDC_LxCLUTWR** register.
  - L8 and AL88 input pixel format, the CLUT has to be loaded by 256 colors
  - AL44 input pixel format, the CLUT has to be loaded by 16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit.
    - L0 (indexed color 0), at address 0x00,
    - L1, at address 0x11….
    - L2, at address 0x22
    - L15, at address 0xFF

Indexed 16 – L4



Indexed 256 - L8

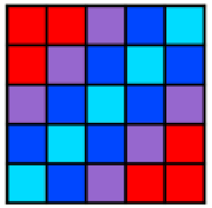


RGB888

Source - wikipedia.org

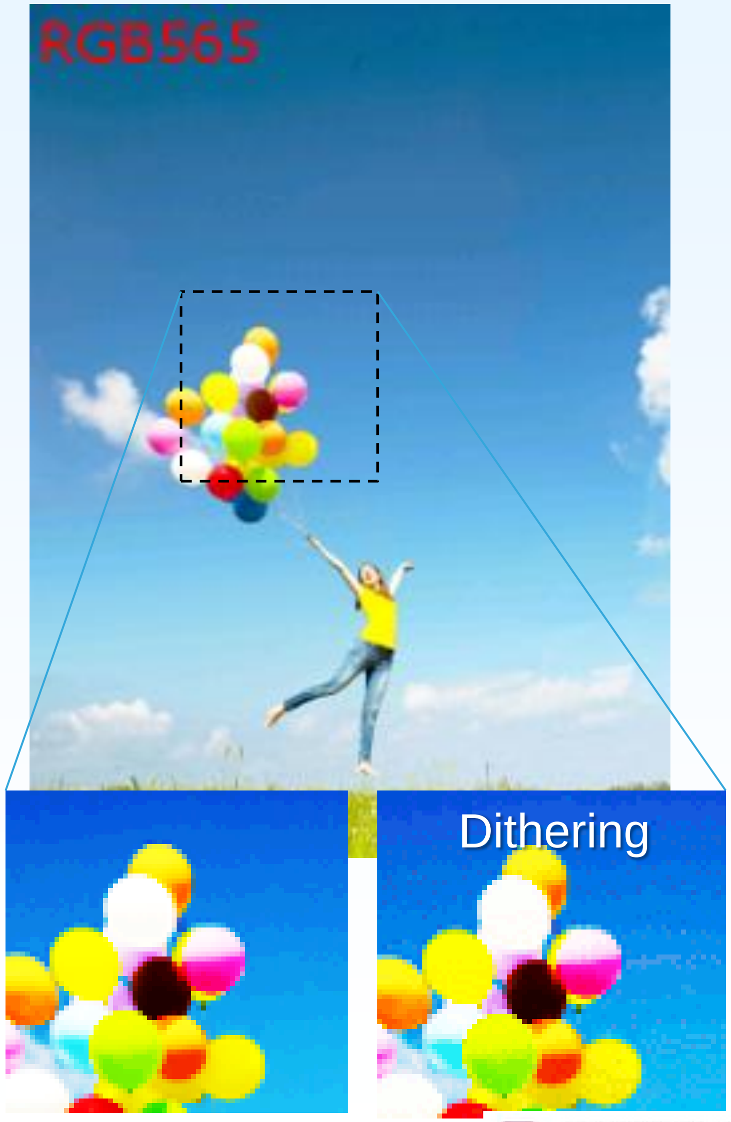
0	0	1	2	3
0	1	2	3	2
1	2	3	2	1
2	3	2	1	0
3	2	1	0	0

0 =	🟥
1 =	🟪
2 =	🟦
3 =	🟦



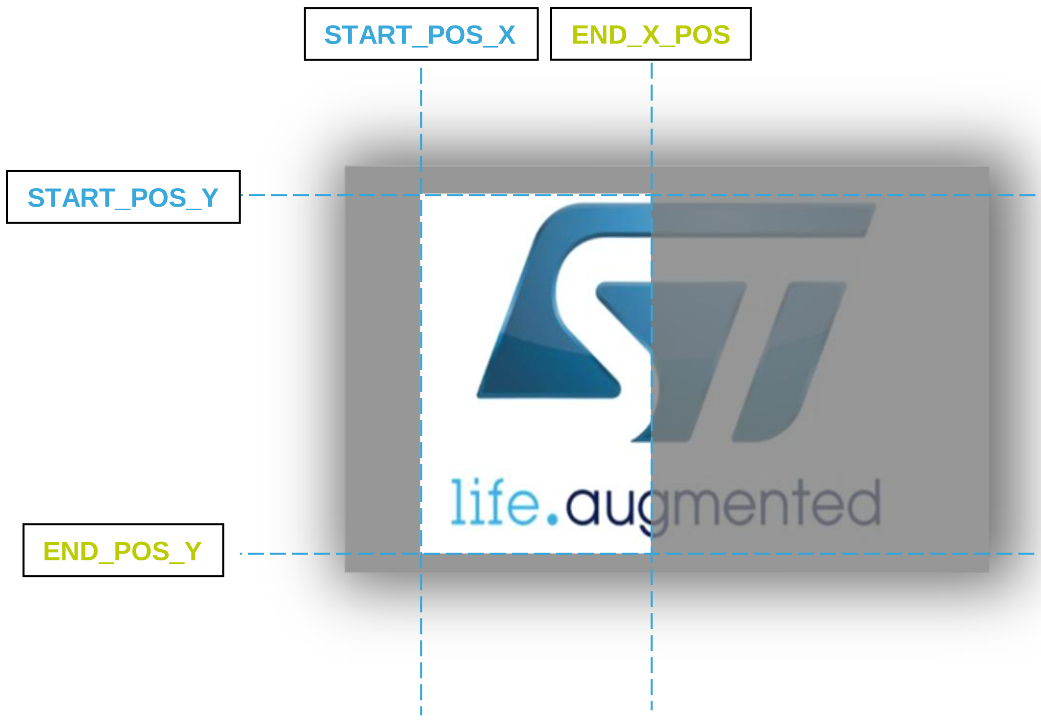
Demo – Color Formats

- Size of pictures displayed
  - RGB 888 - 230kb – needed for pictures with uniform color transitions (like blue sky on the picture)
  - RGB565 - 154kb – good trade off between size and quality
  - L8 - 78kb – for small icons it is usually good enough, without significant quality decrease
- Picture quality
  - Difference is mostly visible in the color transitions (like the blue sky)
  - Dithering is improving the quality, but it make worse details recognition
- STM32F429I-Discovery kit display
  - 18bit interface, it is not possible to display 24bpp (RGB888)
  - Difference between direct RGB888 and RGB565 is not visible
  - HW dithering is improving the quality of RGB888
  - Indexed L8 format can be improved by dithering implemented on PC side



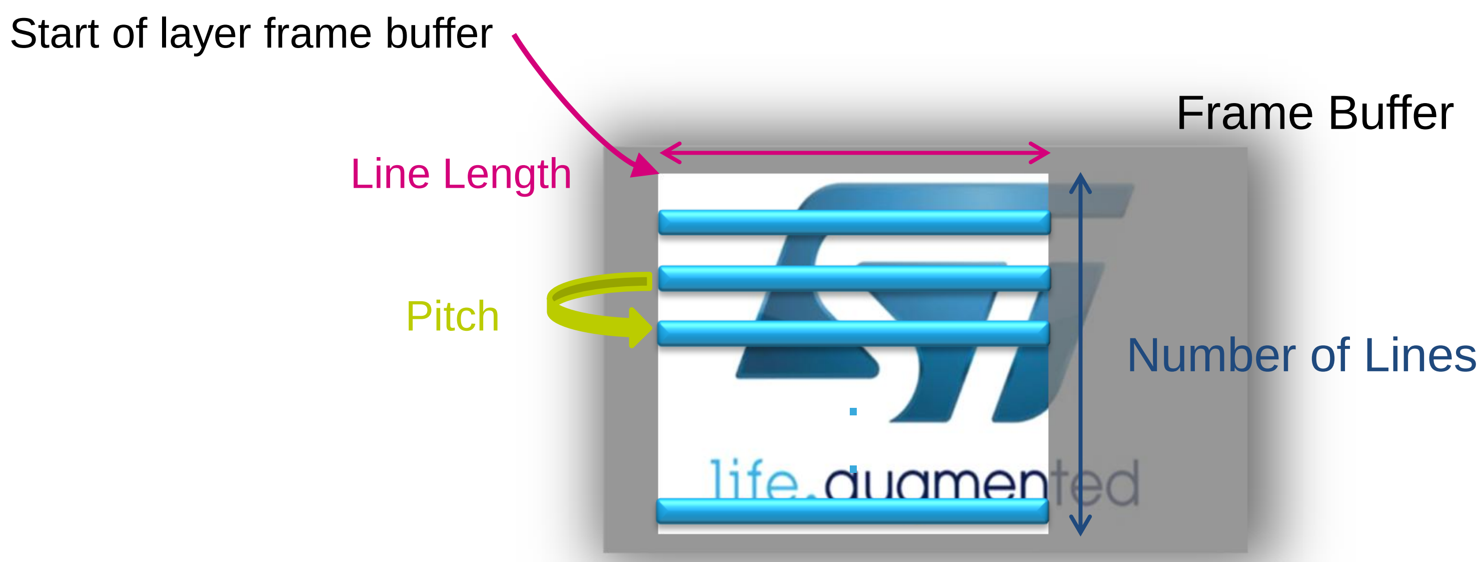
Layer Programmable Parameters: Window

- Window position and size
  - The first and last visible Pixel are configured in the **LTDC_LxWHPCR** register.
  - The first and last visible line are configured in the **LTDC_LxWVPCR**



Layer Programmable Parameters: Color Frame Buffer

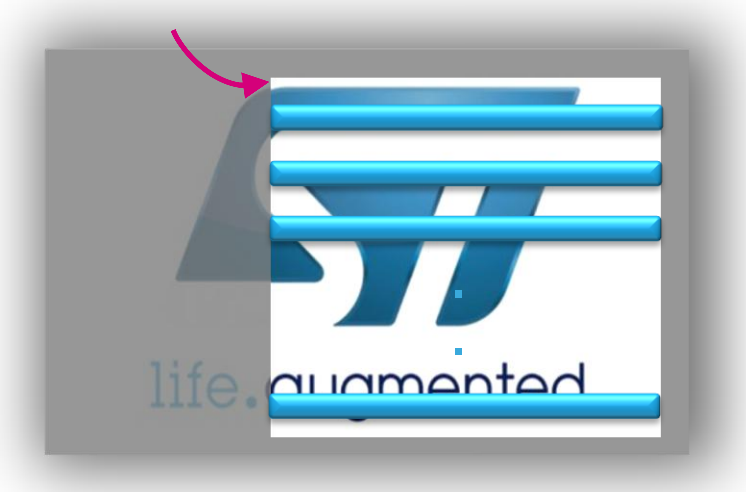
- The frame buffer size, line length and the number of lines settings must be correctly configured :
  - If it is set to less bytes than required by the layer, FIFO underrun error will be set.
  - If it is set to more bytes than actually required by the layer, the useless data read is discarded. The useless data is not displayed.
  - **Pitch =** start of one line and the beginning of the next line in bytes



Layer Programmable Parameters: Color Frame Buffer - effects

- Simply changing the start of the layer frame buffer you can scroll the picture over the layer
  - **The picture stored in the memory must be bigger than actual layer size**

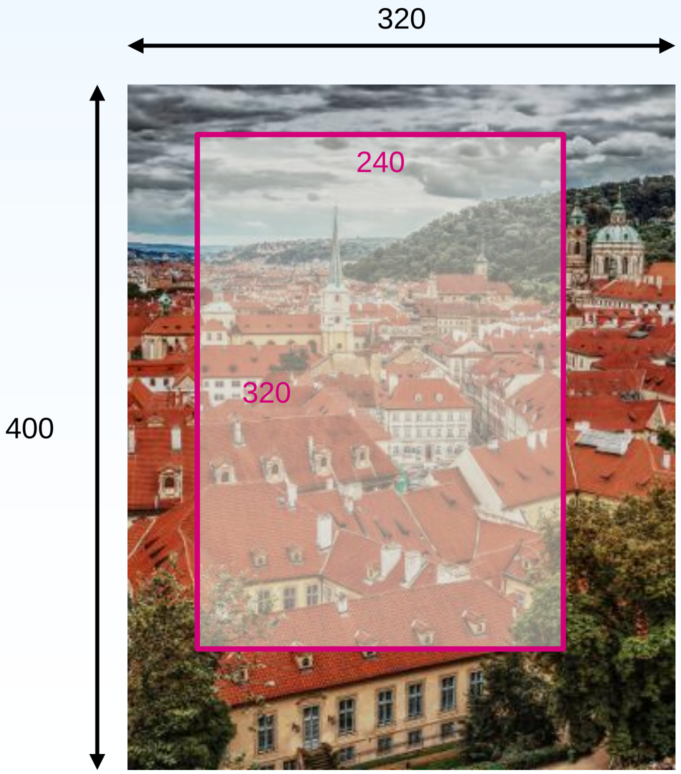
life.augmented



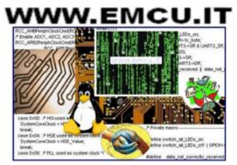
Demo - Moving layer

```
LTDC_Layer_InitStruct.LTDC_CFBStartAdress =
(uint32_t)Image + 3 * (offsetH + (offsetV * Image_width));
```

- Bytes per pixel
- Horizontal scroll
- Vertical scroll
- Base address of image in memory

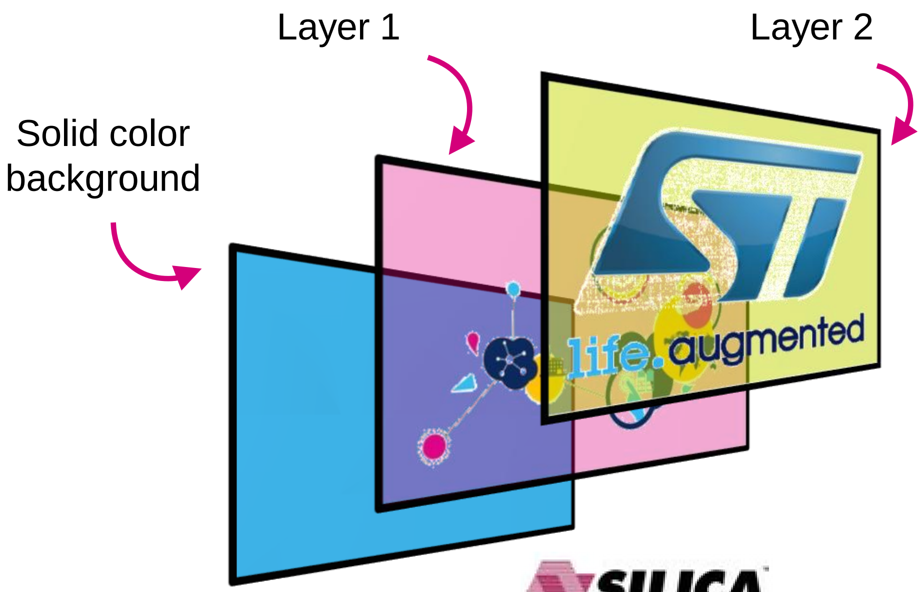


- Just by changing the layer buffer address you can move the visible layer over a bitmap which is bigger than the layer.



Layer Programmable Parameters: Multi-layer blending

- Blending is always active using alpha value
- Blending factors are configured through the **LTDC_LxBFCR** register
  - Fixed Constant alpha
  - Alpha pixel multiplied by the Constant Alpha
- The blending order is fixed and it is bottom up.
- Programmable Background Color for the bottom layer



Blending – Example 1

- Layer 1 blending with background
  - Background is black
  - Layer 2 is disabled.

100%

75%

50%

25%

0%

Blending – Example 2

- Layer 1 and Layer 2 blending
  - Background color is black
  - Layer 1 Constant Alpha set to 100 %
  - Layer 2 Constant Alpha is set to:

100%

75 %

50 %

25 %

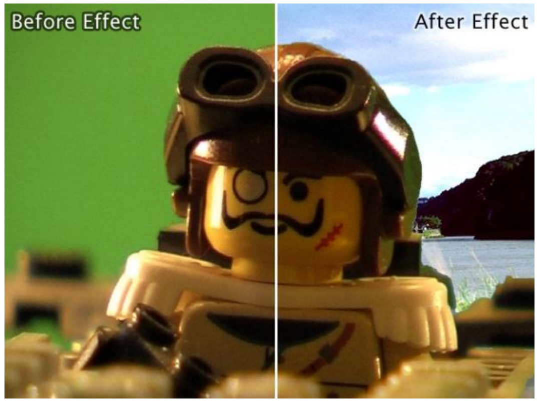
0 %

Demo – Two Layers

- Example is using two layers
  - Layer 1 – static background picture
  - Layer 2 – moving ST logo with alpha channel
- ST logo is moved using layer 2 parameters only, no load of CPU
- Blending is done directly by Constant Alpha parameter, no load of CPU, done by LTDC controller directly

Layer Programmable Parameters: Color Keying

- Transparent color (RGB) can be defined for each layer in the LTDC_LxCKCR register
- When the Color Keying is enabled, the current pixels is compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.
- Color Keying can be enabled on the fly for each layer in the **LTDC_LxCR** regiser



Chrom-ART Accelerator™

STM32F4 Graphic Accelerator (DMA2D)

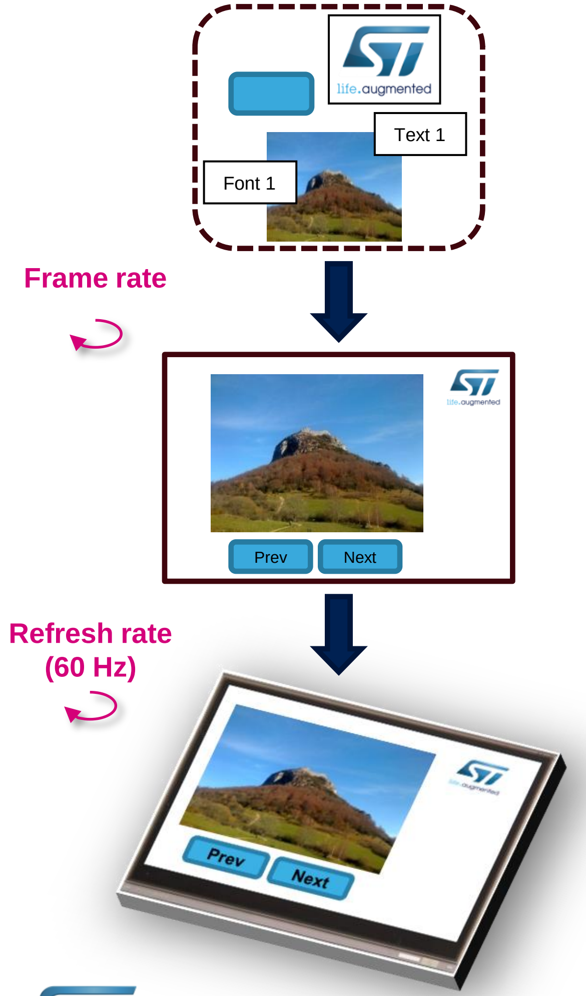
MMS-MCD Applications

life.augmented

SILICA An Avnet Company

WWW.EMCU.IT

Overview



Step 1 : Create the content in a frame buffer

- The frame buffer is build by composing graphical primitive
- This operation is done by the **CPU** running a graphical library software
- It can be **accelerated** by a **dedicated hardware** used with the CPU through the graphical library
- More often the frame buffer can be updated, more fluent will be the animations (animation fps)

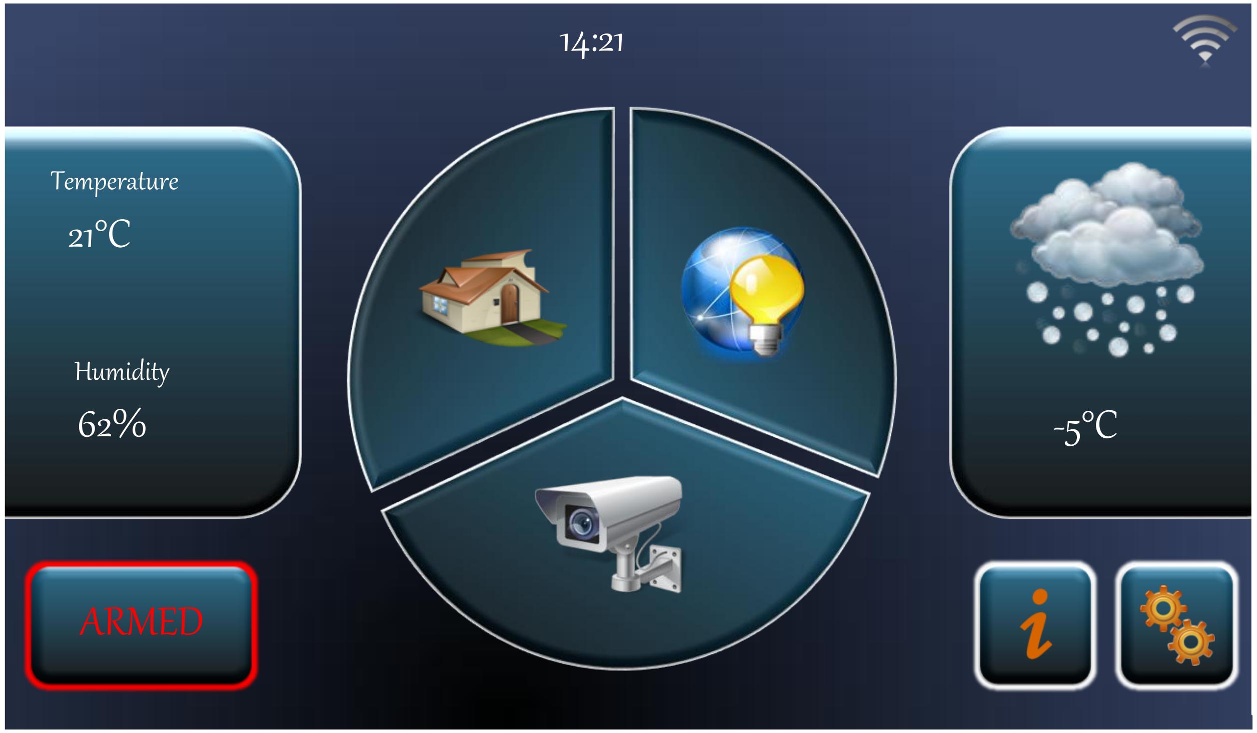
Step 2 : Display the frame buffer

- The frame buffer is transferred to the display through a dedicated hardware interface
- Graphical oriented microcontroller are offering a **TFT controller** to drive directly the display
- The frame buffer must be sent at 60fps to have a perfect image color and stability (independently from the animation fps)

Graphical content creation

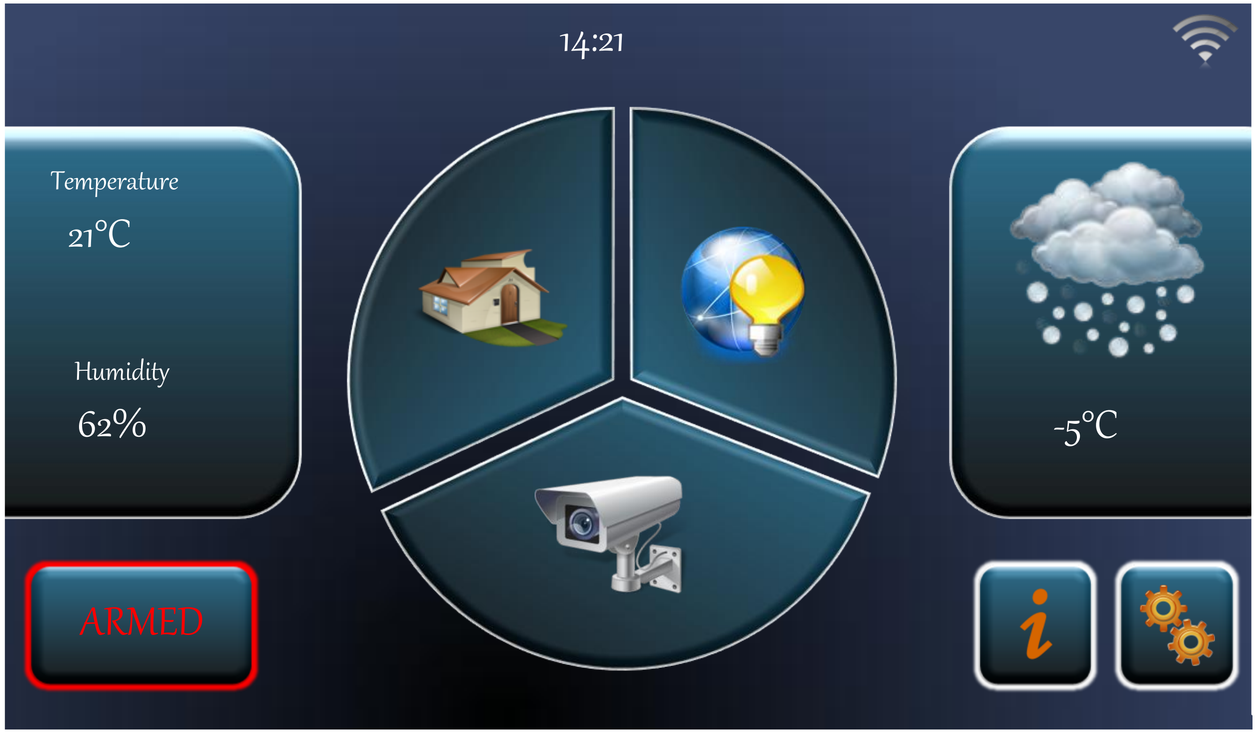
Creating something « cool »

- How the frame buffer is generated for creating a "cool" graphical interface to be displayed through the TFT controller ?



Frame buffer construction

- The frame buffer is generated drawing successively all the graphic objects



Frame buffer generation needs

- The resulting frame buffer is an **uncompressed bitmap** of the size of the screen.

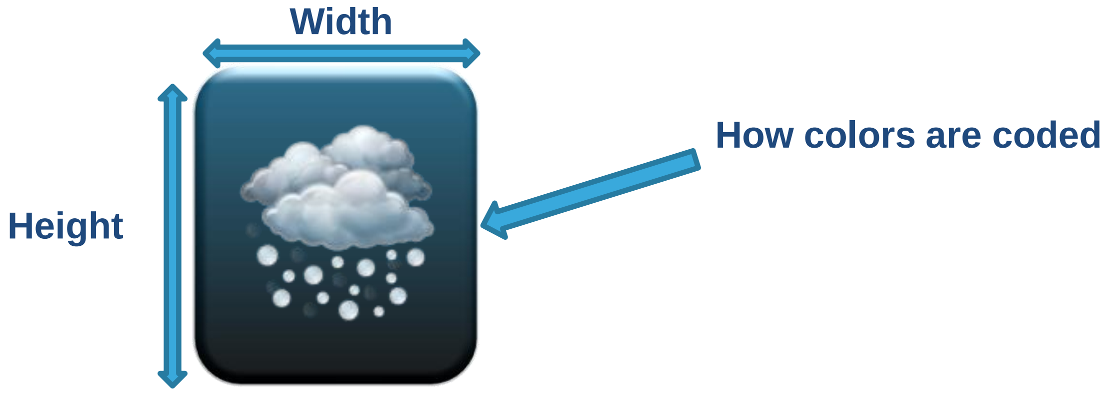
Temperature
21°C
Humidity
62%
14:21
ARMED
-5°C

- Each object to be drawn can be
  - A **Bitmap** with its own color coding (different from the final one), compressed or not
  - A **Vector Description** (a line, a circle with a texture...etc....)
  - (A **Text**)

Bitmaps

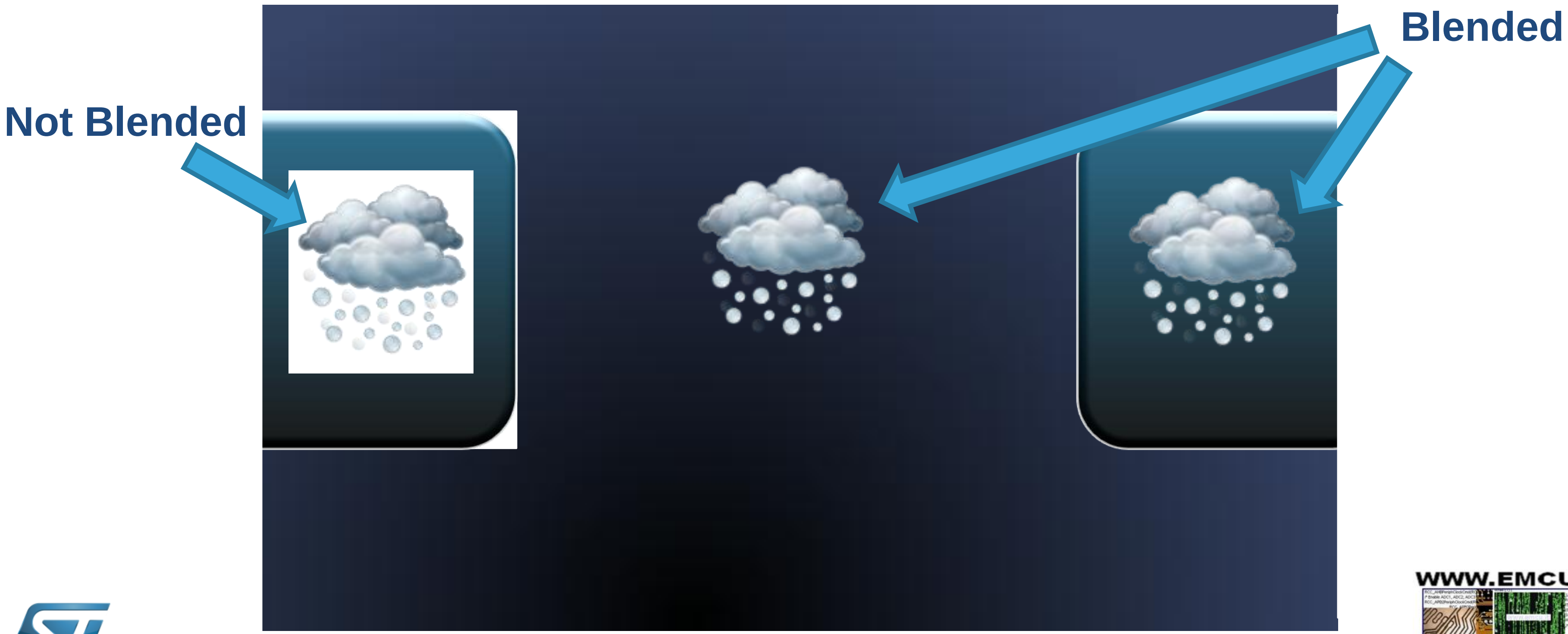
Overview

- Bitmap are an array representation of an image
- It shall have the **at least** following properties
  - Width (in pixel)
  - Height (in pixel)
  - Color mode (direct, indirect, ARGB8888, RGB565...etc...)
  - Color Look Up Table (optional)



Blending

- Blending consist in drawing an image onto another respecting the transparency information
- As a consequence blending implies to read 2 sources, then blend then write to the destination



Back to our « cool » interface

Background

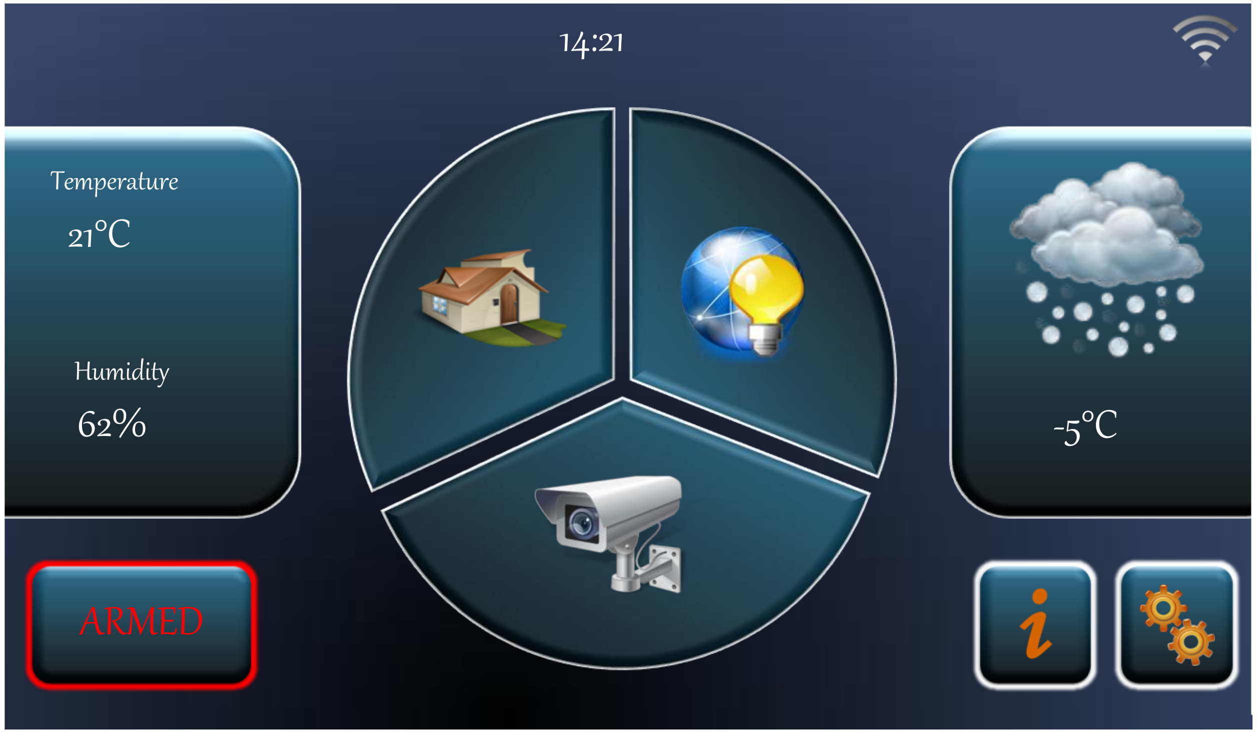
Almost Uniform
L8 mode

Button

Round shape
Gradient
ARGB8888 mode

Icon

Complex shape
Many colors
ARGB8888 mode

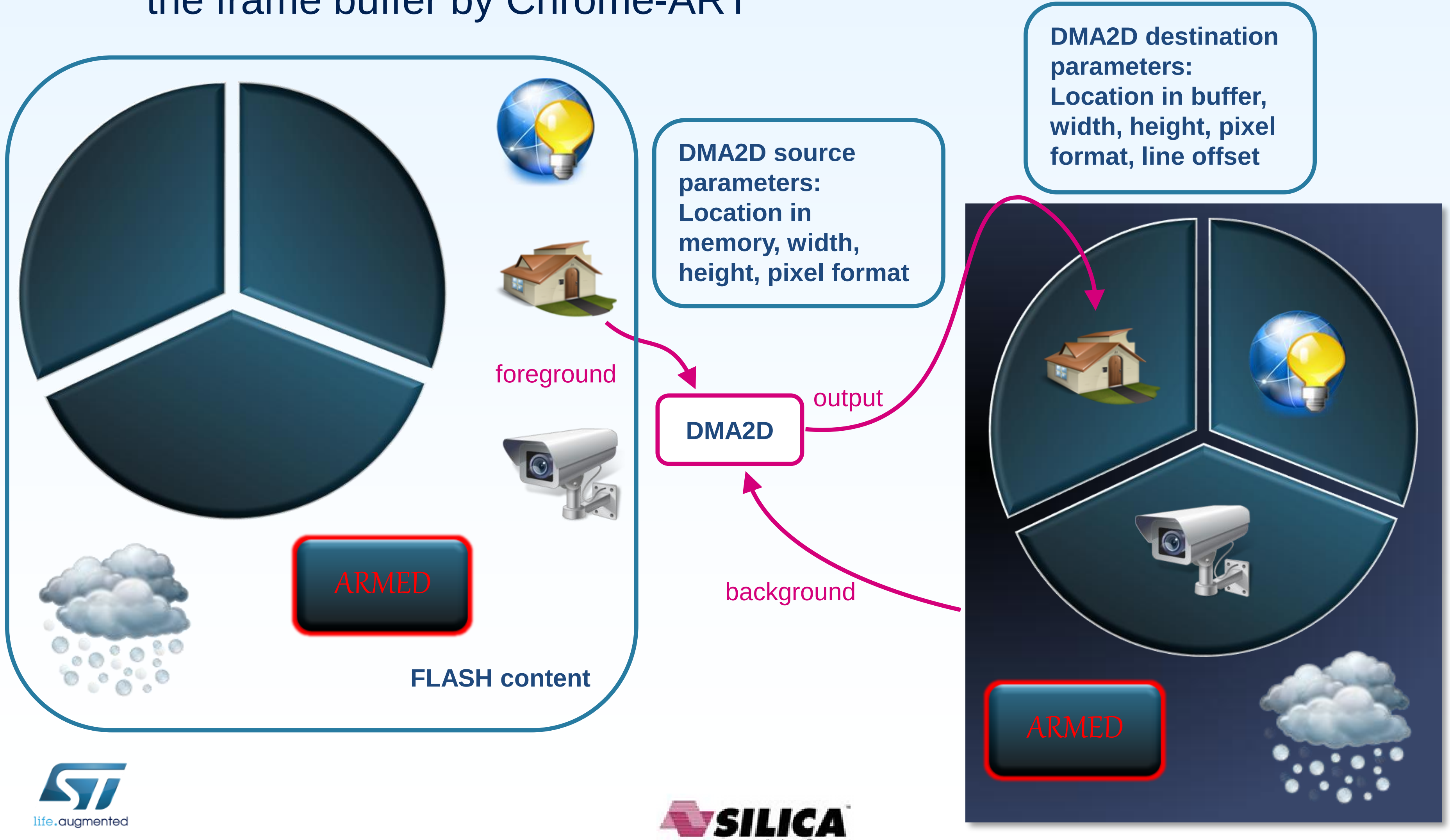


Fonts

Specific management with A8 or A4 mode

Demo – Content Creation

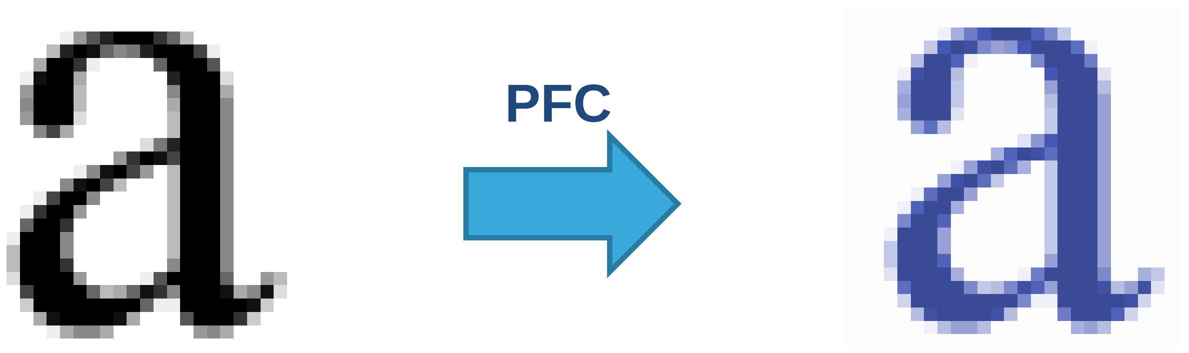
- Moving successively graphical elements from internal FLASH into the frame buffer by Chrome-ART



DMA2D destination parameters: Location in buffer, width, height, pixel format, line offset
DMA2D source parameters: Location in memory, width, height, pixel format
foreground
output
DMA2D
background
ARMED
FLASH content
ARMED

Font management

- Bitmap fonts are managed only using alpha channel (transparency)



ROMed character bitmap (A8 or A4)

Generated character bitmap with color information (ARGB8888, ARGB1555, ARGB4444) or (RGB888, RGB565)

Summary

- To generate my 24bpp frame buffer I will have to
  - Copy background from the ROM to the frame buffer with PFC L8 to RGB888
  - Copy buttons & icons from the ROM to the frame buffer with blending
  - Copy characters from the ROM to the frame buffer with PFC A4 to ARGB8888 and blending

Temperature
21°C
Humidity
62%
ARMED
14:21
-5°C

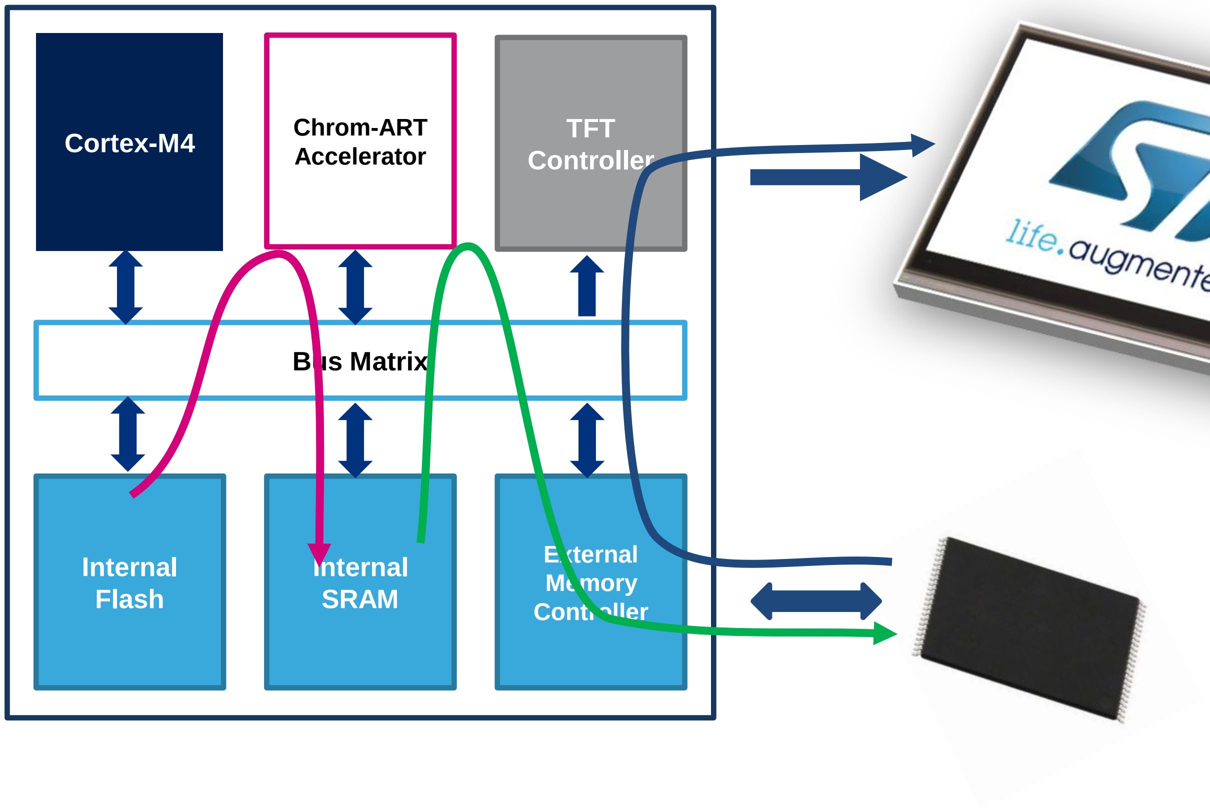
- Many copy operations with pixel conversion. Can be done by CPU, but it's very time consuming → HW acceleration helps.

Chrom-ART Accelerator (DMA2D)

Overview

- The Chrom-ART combines both a DMA2D and graphical oriented functionality for image blending and pixel format conversion.
- To offload the CPU of raw data copy, the Chrom-ART is able to copy a part of a graphic content into another part of a graphic content, or simply to fill an part of a graphic content with a specified color.
- In addition to raw data copy, additional functionality can be added such as image format conversion or image blending (image mixing with some transparency).

Typical data flow



→ Creation of an object in a memory device by the Chrom-ART

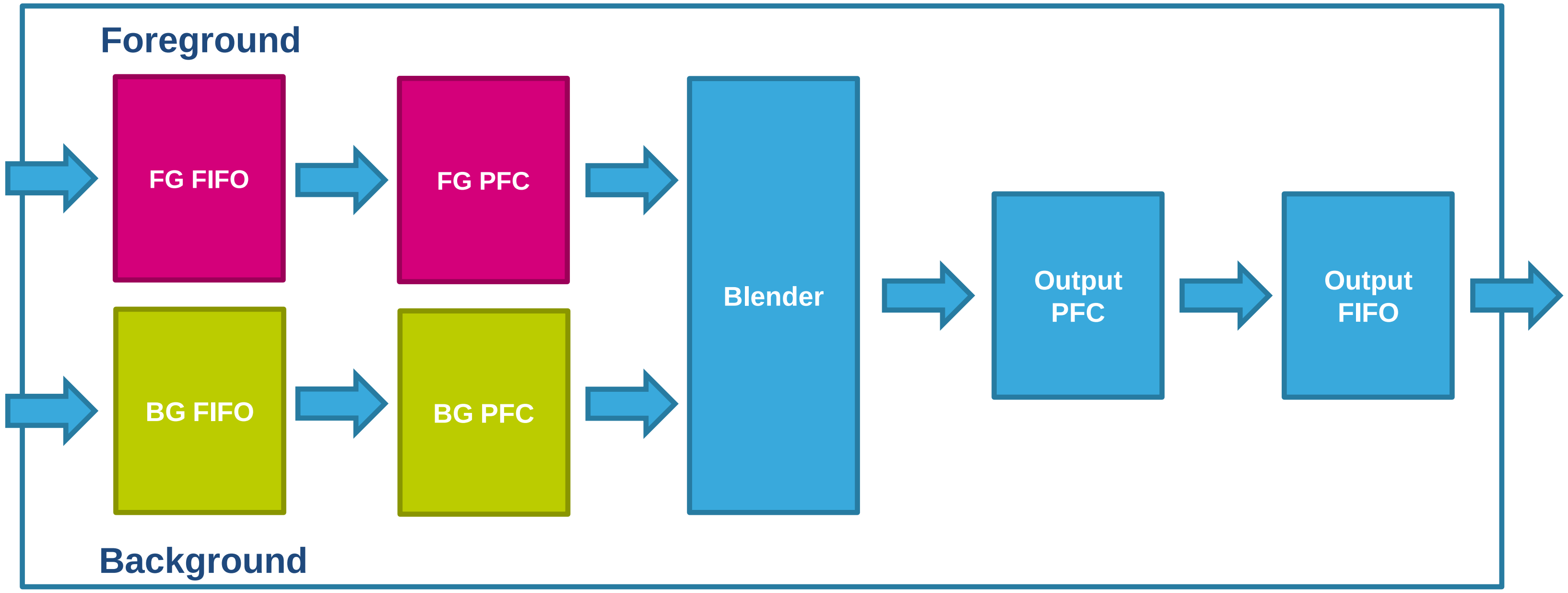
→ Update/Creation of the frame buffer in the external RAM by the Chrom-ART

→ TFT controller data flow

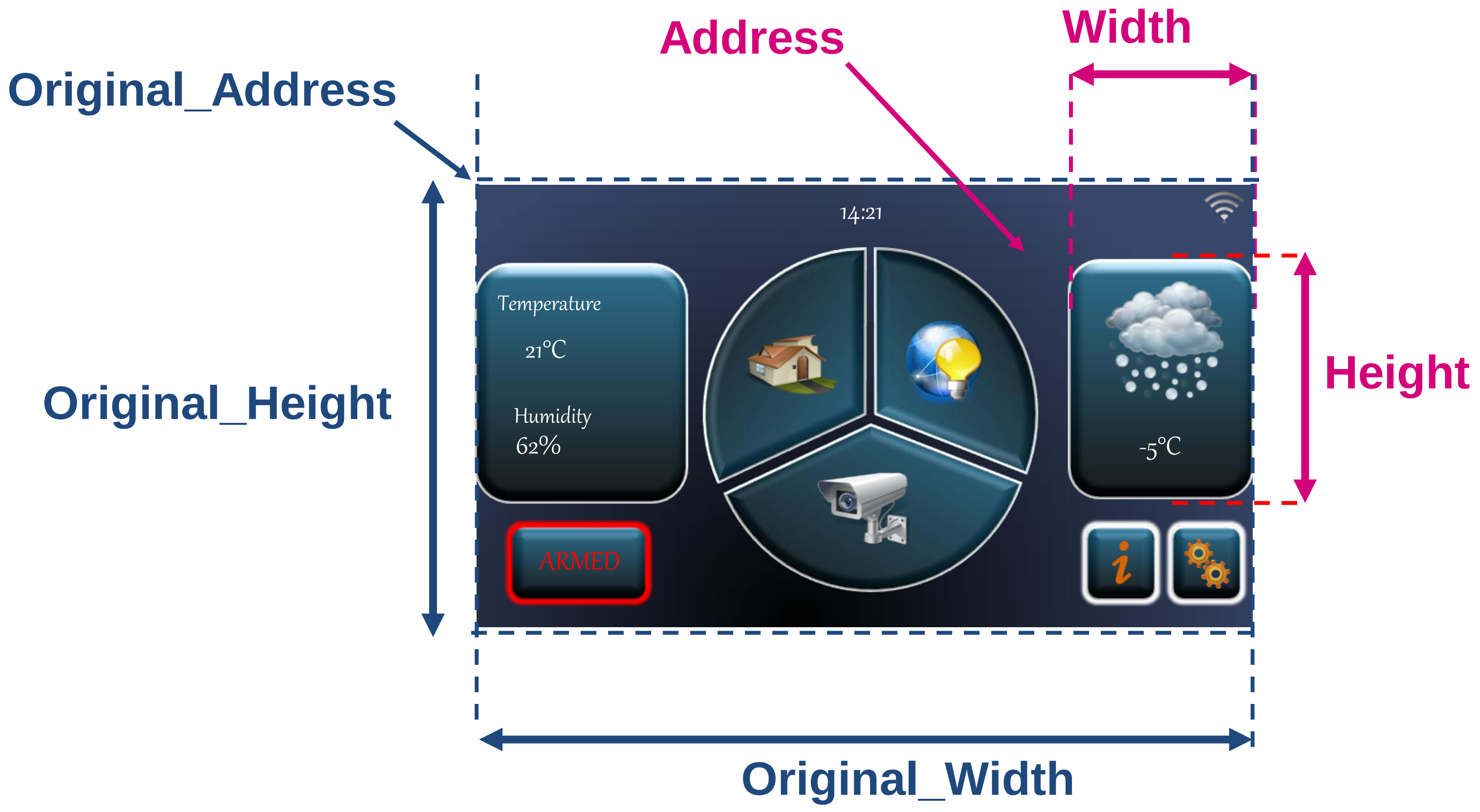
Chrom-ART features

- AHB bus master with burst access to any system memory
- Programmable bitmap height, width and address in the memory
- Programmable data format (from 4-bit indirect up to 32-bit direct)
- Dedicated memory for color lookup table (CLUT) independent from LTDC
- Programmable address destination and format
- Optional image format conversion from direct or indirect color mode to direct color mode
- Optional blending machine with programmable transparency factor and/or with native transparency channel between two independent image inputs to the destination image.

Chrom-ART pixel pipeline



Bitmap parameter



Address = **Original_Address** + (X + **Original_Width** * Y) * BPP
LineOffset = **Original_Width** - **Width**

CLUT management

- When an indirect color mode L8, AL44 or AL88 is used, the bitmap CLUT must be loaded into the Chrom-ART
- Each FG and BG has its own dedicated memory for CLUT.
- The CLUT can be loaded manually when no Chrom-ART operation are on going
- The CLUT can be loaded automatically configuring the CLUT format (24-bit or 32-bit), the CLUT size and the CLUT address and setting the CLUT Start bit

Supported operations

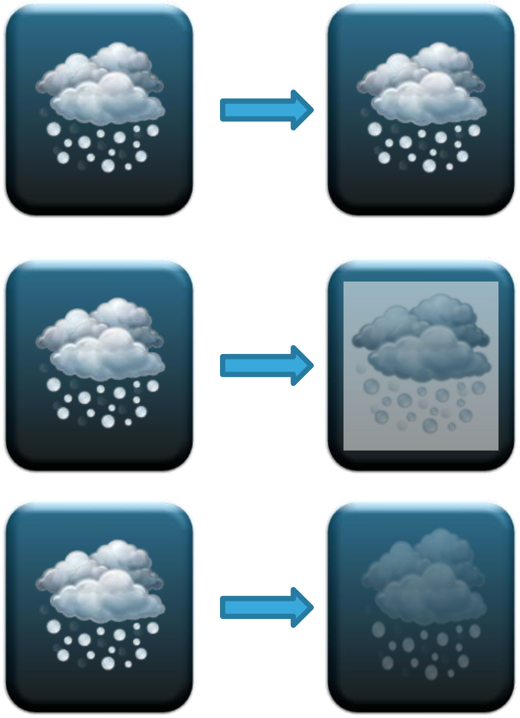
- 4 functional modes are supported
  - **Register to memory** : the destination bitmap is filled with the specified output color
  - **Memory to memory** : the data are fetch through the FG and are written to the destination bitmap without any color format modification (no PFC)
  - **Memory to memory with PFC** : the data are fetch through the FG and are written to the destination bitmap after being converted into the destination color format (PFC)
  - **Memory to memory with PFC and blending** : the data are fetched through the FG and the BG, are converted, are blended together and are written to the destination bitmap

- Once configured, the Chrom-ART is launched setting the Start bit

- The operation can be either suspended through the Suspend bit or aborted through the abort bit

Alpha modulation

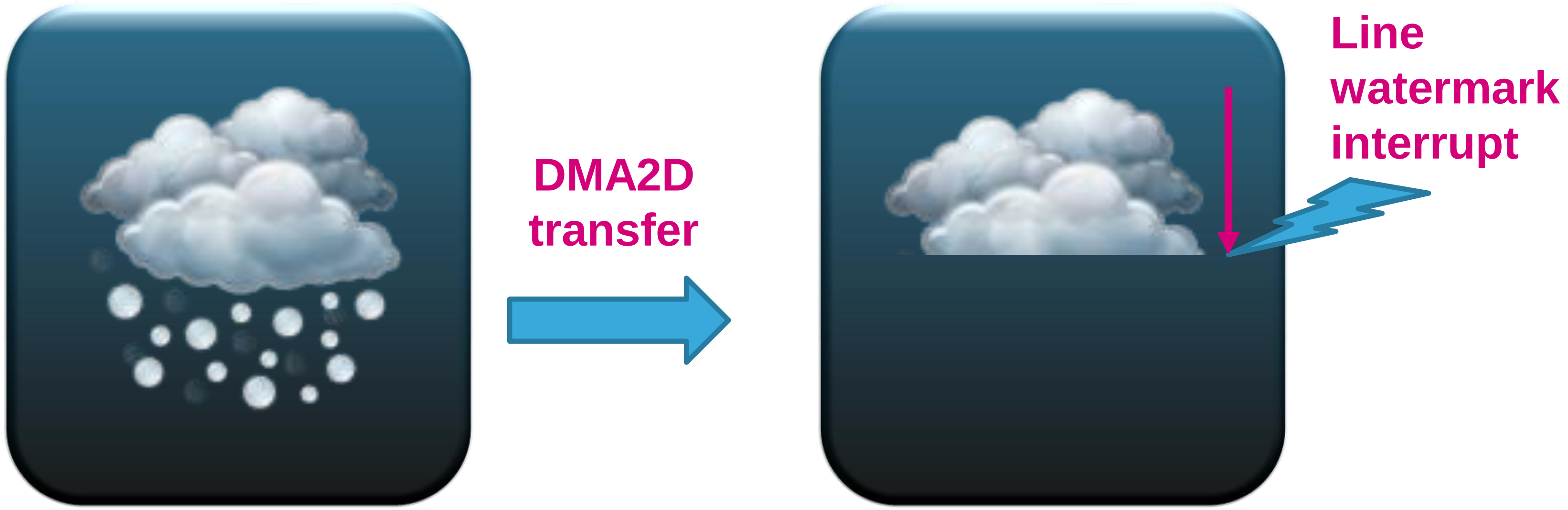
- When the PFC is activated the Alpha value of the pixel can be modified as follow
  - Kept as it is
  - Replaced by a fixed one
  - Replaced by the current one multiplied by the fixed one



- This allows to perform easily fade in/out animations

Line Watermark

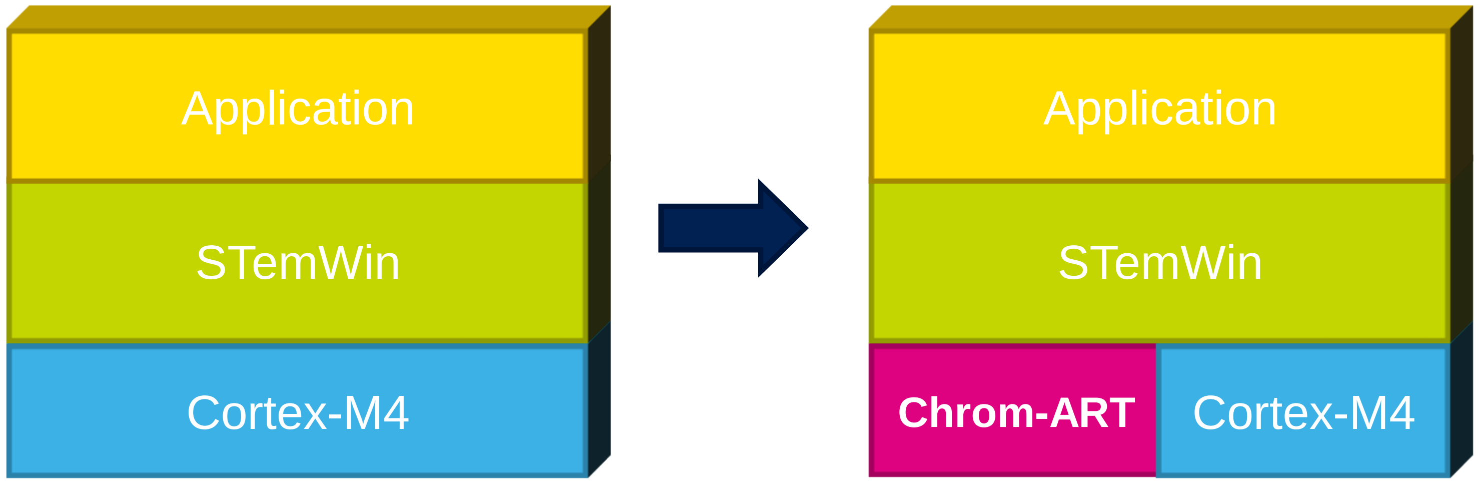
- For synchronization purpose, an interrupt can be generated when a specified line has been written by the Chrom-ART into the memory
- The line number is defined by the filed LW[15:0] of the DMA2D_LWR register



Interrupts

Interrupt	Flag	Enable	Clear	Description
Transfer Error	TERIF	TERIE	CTERIF	An AHB error occured during a Chrom-ART operation
Transfer Complete	TCIF	TCIE	CTCIF	The programmed operation is complete
Transfer Watermark	TWIF	TWIE	CTWIF	The Watermarked line has been written
CLUT Access Error	CAEIF	CAEIE	CCAEIF	An error has been generated accessing the CLUT (CPU tried to access the CLUT when the Chrom-ART is being active)
CLUT Transfer Complete	CTCIF	CTCIE	CCTCIF	The CLUT transfer is complete
Configuration Error	CEIF	CEIE	CCEIF	The Chrom-ART has been launched with an illegal configuration

Integration with Graphic Library



- **Chrom-ART** integration is **transparent for the application**
- The low-level drivers of the graphical stack are upgraded to directly use **Chrom-ART** for data transfer, pixel format conversion and blending
- **CPU load is decreased** and **graphical operations are faster**

STM32 Solution for Graphics

- STM32 offers a unique graphical capability in the Cortex-M based MCU perimeter
  - Real TFT Controller for optimum display control
  - External memory interface to connect both Flash for static content and SRAM or SDRAM for dynamic content and frame buffer
  - On-chip hardware acceleration deeply coupled with graphical library
  - Standard graphical library taking advantage of on-chip graphical acceleration

TouchGFX & STM32F4

Revolutionizing the User Experience of Embedded User Interfaces

STM32 F4

World leading MCU with LCD-TFT Controller and Chrom-ART Accelerator for highly efficient graphics rendering.

TouchGFX

A software framework from Draupner Graphics that enables high-end graphics on low-cost hardware (e.g. Cortex M).

life.augmented

SILICA An Avnet Company

DRAUPNER© GRAPHICS

Your Business Gains with TouchGFX & STM32

- A Product Differentiator:
  - Make your embedded product stand out by giving it a unique smartphone look and feel.
- A Value Creator:
  - Reach the full potential of your embedded product with an intuitive, responsive and consistent user interface.
- A Brand Booster:
  - Let your embedded product express your overall brand and visual identity.
- A Money Saver:
  - Achieve all of the above on low-cost hardware with TouchGFX & STM32

When to Use TouchGFX & STM32F4?

Whenever a **high-end GUI performance** brings value to your embedded product,

and, when one or more of the statements below apply:

- **Low unit cost** is essential for your business profit.
- **Low complexity** hardware and software make sense in your application.
- **Low power consumption** is important for the value of your application.

With TouchGFX & STM32F4 you get a high-performing GUI platform with plenty of CPU power to run the control application. No need for a complex and expensive application processor.

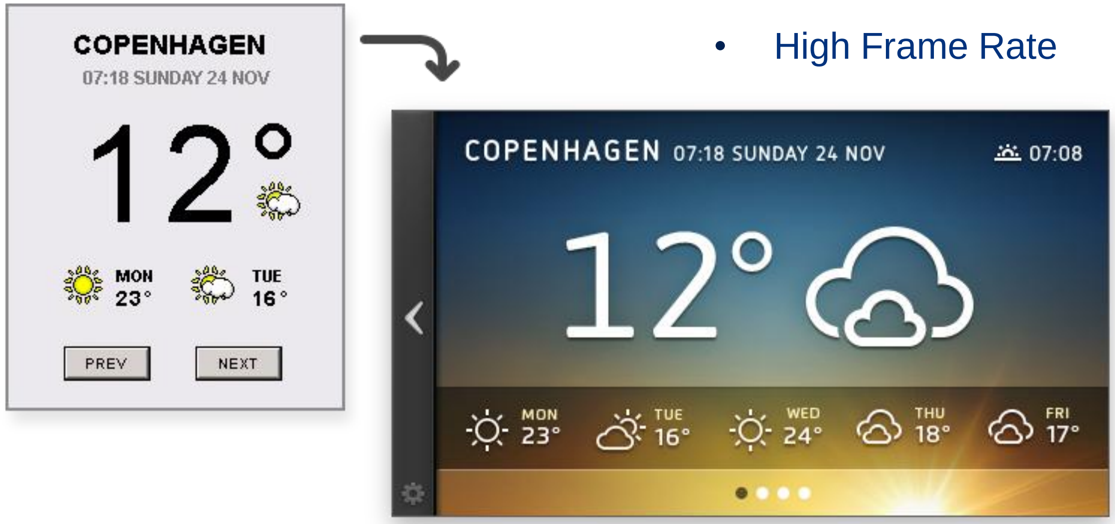
Differentiate Your Product with a Unique GUI

Modern touch displays and TouchGFX can give you:

- A unique and appealing look
- Better navigation
- Proper and safe operation
- More and better information
- Advanced features
- Cohesive brand across all platforms

Graphical features of TouchGFX:

- Transparency
- Alpha-blending
- Anti-Aliased Fonts and Kerning
- Touch Gestures
- Animations
- Full Screen Transitions
- High Frame Rate



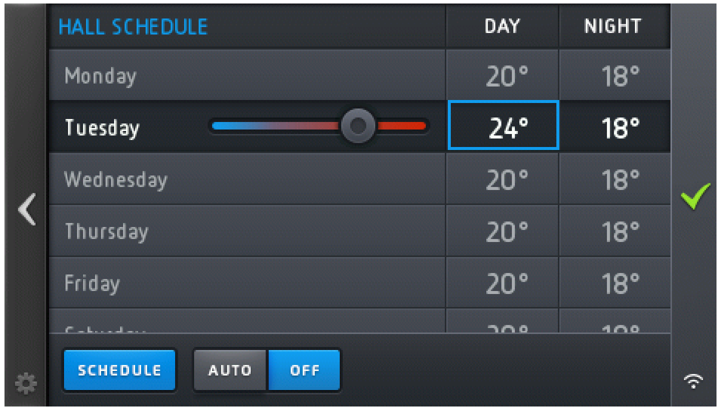
TouchGFX Hardware Setup

- Display Sizes up to 10”
  - QVGA: 320x240 (often 3.5”)
  - WQVGA: 480x272 (often 4.3”)
  - WVGA: 800x480 (often 7.0”)
  - WSVGA: 1024x600 (often 10”)
- Can be used on any MCU
  - Targeting Cortex M with TFT controller
  - MCU load - Typically less than 15%

- Memory Footprint
  - Internal RAM: 10 - 35 kB
  - Internal Flash: 20 kB (framework) + 10 - 100 kB (application)
  - External RAM: 50 kB - 3 MB (depending on display resolution and no. of framebuffers)
  - External Flash: 1 - 8 MB (graphics and texts)

TouchGFX Technology

- C++ Framework
  - TouchGFX application
  - TouchGFX core
  - Board-specific hardware abstraction layer
- Bitmap Graphics
  - Extensive use of DMA transfer (offloading the MCU)
- Advanced Rendering Algorithms
  - Visible Surface Determination and customized invalidation techniques minimize the number of pixels being drawn
- Easy and Automated Development
  - Multiple language support made easy
  - Bitmap and font converter
  - PC simulator



HALL SCHEDULE	DAY	NIGHT
Monday	20°	18°
Tuesday	24°	18°
Wednesday	20°	18°
Thursday	20°	18°
Friday	20°	18°

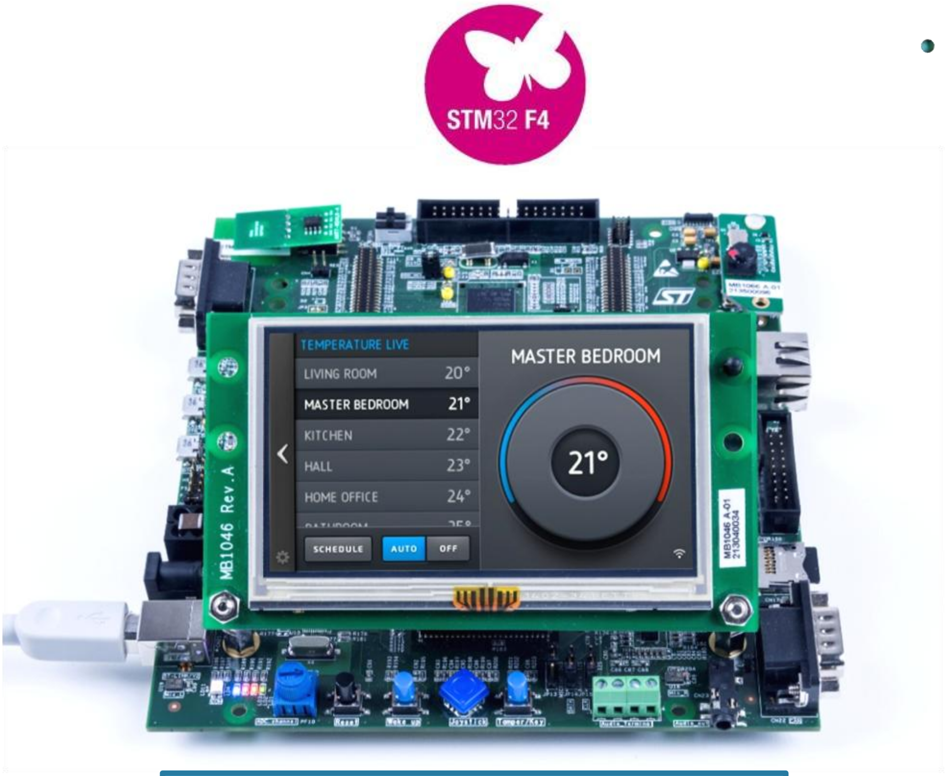
STM32F4 GUI Advantages

- ST LCD-TFT Controller
- ST 2D DMA & Chrom-ART Graphic Acceleration
  - On-the-fly image format conversion and alpha blending
  - Two-dimensional DMA transfers allow very efficient bitmap drawing

Hardware support for alpha-blending and text rendering ensures a very low MCU load, even on the most complex screens.

The TouchGFX 2014 Demo Application

Available on STM32F429I EVAL board



Get the hex file here:
http://ftp.mjolner.dk/TouchGFX/demos/touchgfx_demo2014.zip

- Hardware setup:
  - STM32F429, Cortex M4, 180 MHz
    - MCU load typically 5 - 20 %
  - 4.3" Touch Display
    - 480x272, 25 frames pr. sec
  - Internal RAM: 26 kB
    - 20 kB for Framework/Application
    - 6 kB Stack
  - Internal ROM: 155 kB
    - 15 kB Framework
    - 110 kB Application
    - 30 kB Fonts
  - External RAM: 783 kB (Framebuffers)
  - External Flash: 12 MB (Images)

life.augmented

WWW.EMCU.IT

The TouchGFX 2014 Demo Application

Available on STM32F429I Discovery Kit

- Hardware setup:
  - STM32F429, Cortex M4, 180 MHz
    - MCU load typically 5 - 10 %
  - 2.4” Touch Display
    - 320x240, min. 25 frames pr. sec
  - Internal RAM: 26 kB
    - 20 kB for Framework/Application
    - 6 kB Stack
  - Internal Flash: 75 kB + 1.9 MB
    - 15 kB Framework
    - <30 kB Application
    - <30 kB Fonts
    - 1.9 MB Images (normally placed in external flash)
  - External RAM: 326 kB (Framebuffers)
  - External Flash: Non

Get the hex file here: http://ftp.mjolner.dk/TouchGFX/demos/touchgfx_demo2014_small.zip

Get Started with 7 Simple Steps

1. Request the free, fully functional, TouchGFX Evaluation at www.TouchGFX.com
2. Download the Evaluation and explore the framework and demo examples
3. Create your own TouchGFX demo application and test it on the PC simulator
4. Get a proof of concept by compiling and flashing your demo application to the STM32429I EVAL board or Discovery kit
5. Buy a license matching your business
6. Customize the porting to your target hardware
7. Compile and flash your application to your target hardware

License Models in Detail

License Model	**Production Volume**	**Price in €** *Note 1*	**Deliveries**	**Support & maintenance** *Note 3*
Evaluation *Fully functional, supporting various developer boards incl. STM32F429I EVAL + Discovery kit*	None For evaluation purposes only	Free of charge	• TouchGFX Core - precompiled library • Ports for supported developer boards, precompiled libraries • Demo apps, examples, templates, source code • TouchGFX developing tools • Documentation	• Limited support, 3 months • No maintenance
Product Line Limited *Note 2*	Maximum 3.000 units per year	3.000,- *Optional: Support & maintenance +3.000,-*	*Deliveries as Evaluation License +* • Port for selected developer board, source code.	• Limited support, one year • No maintenance • *Option: Support & maintenance*
Product Line Unlimited *Note 2*	Unlimited	15.000,-	*Deliveries as Evaluation License +* • Port for selected developer board, source code.	• Support & maintenance, one year • *Option: Customized extended support & Maintenance*
Customized	On request	Depending on scope of license	• On request • Full source code available	• On request

Note 1: All license fees are one-off. No royalties and developer licenses.
Note 2: Product Line License: Used on a line of products where all the products have the same function and same user interface. An example would be a set of microwave-ovens where different units (variants) have different features (e.g. level of max. microwave power).
Note 3: Maintenance: Product updates and bugfixes

WWW.EMCU.IT

www.st.com/stm32f4

life.augmented

SILICA
An Avnet Company

WWW.EMCU.IT