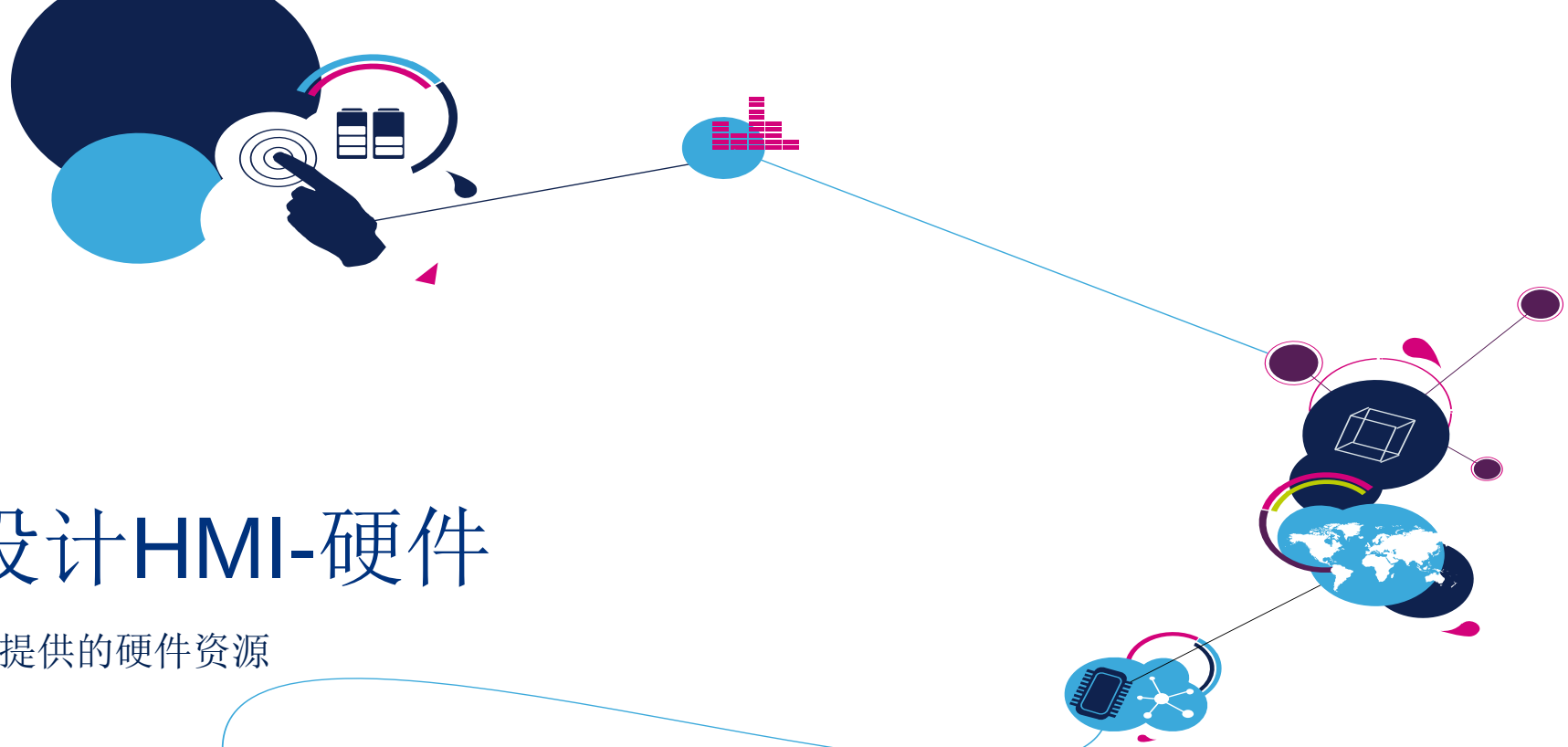


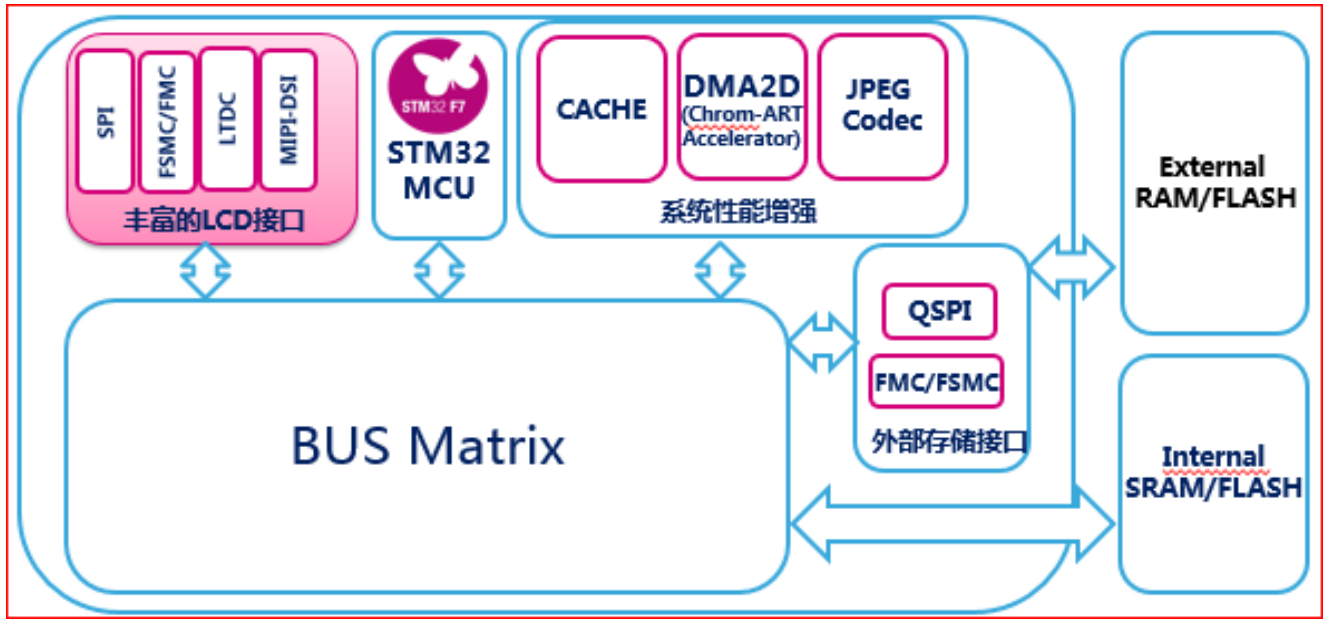
使用STM32 设计HMI-硬件

介绍使用STM32提供的硬件资源

life.augmented

ST领先的硬件

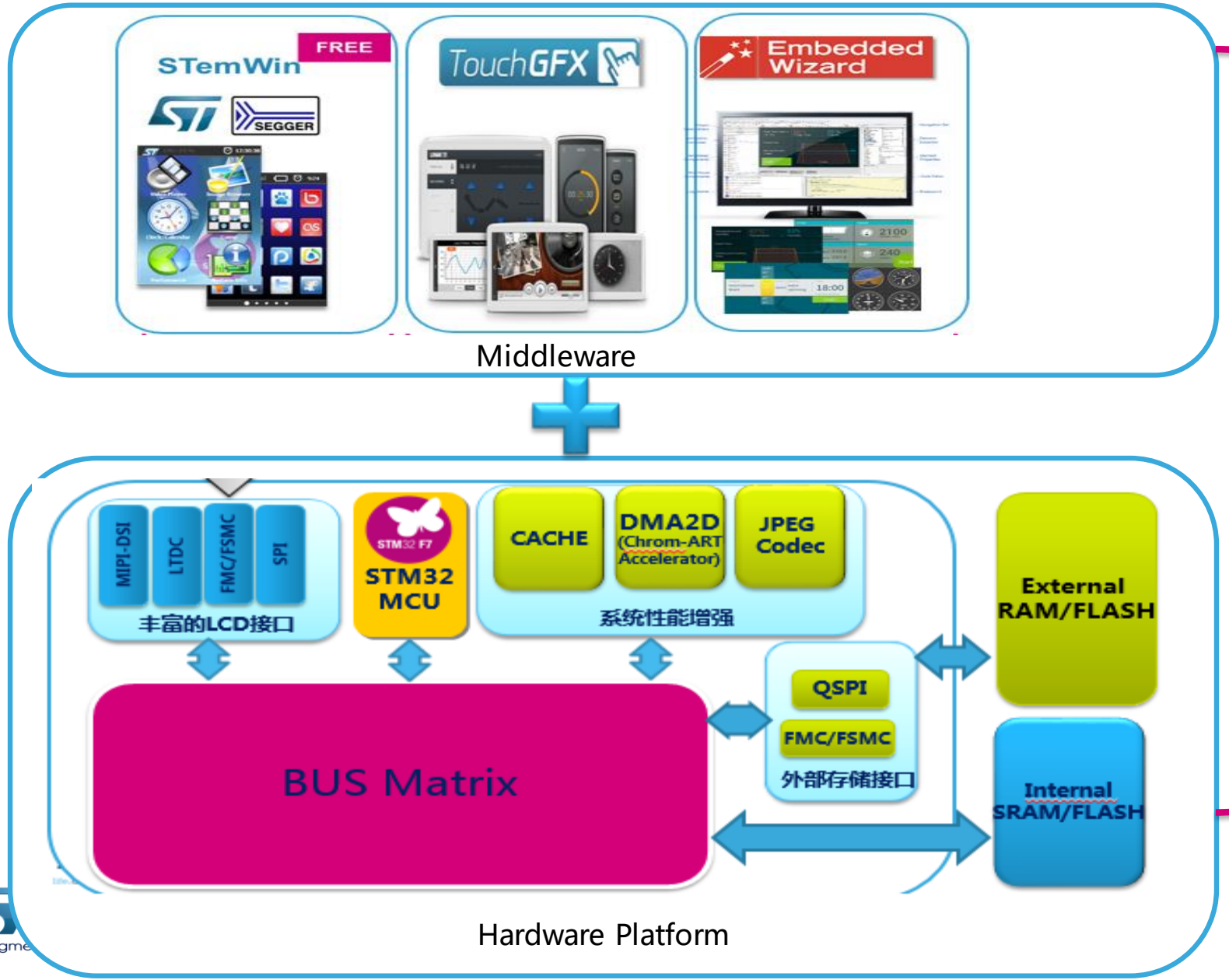
硬件平台



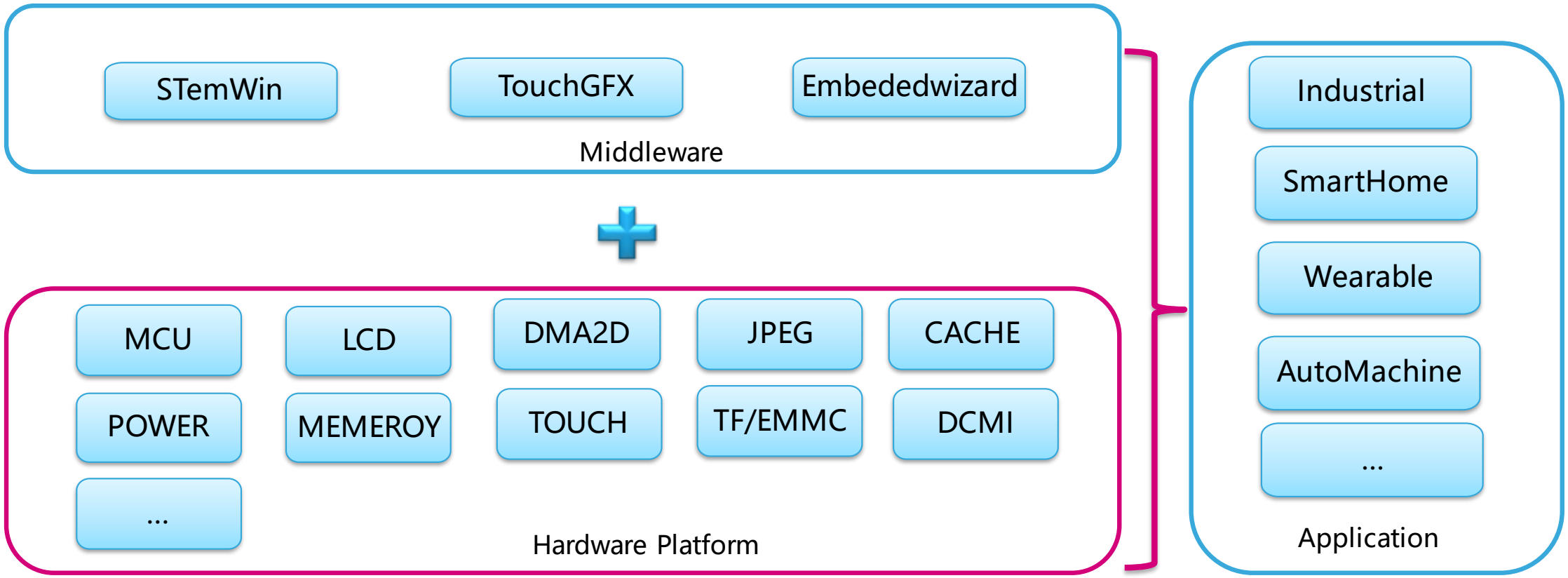
L496	L4R9	F429	F469	F746	F769	H7
M4 @ 80 MHz Chrom-ART	M4@ 120 MHz Chrom-ART Chrom-GRC	M4 @ 180MHz Chrom-ART	M4 @ 180MHz Chrom-ART	M7@ 216 MHz Chrom-ART	M7@ 216 MHz Chrom-ART JPEG Codec	M7@ 400 MHz Chrom-ART JPEG Codec
Display Parallel IF 320 KB RAM FSMC QSPI	DSI Display parallel IF 640 KB RAM FSMC QSPI	TFT ctrl Display parallel IF 256 KB RAM FMC	DSI & TFT ctrl Display parallel IF 384 KB RAM FMC QSPI	TFT Ctrl Display parallel IF 320 KB RAM FMC QSPI	DSI & TFT ctrl Display parallel IF 512 KB RAM FMC QSPI	TFT ctrl Display parallel IF 1M RAM FMC QSPI

life.augmented

STM32 GUI Solution



Zoom In
STM32 GUI Solution



STM32 GUI HW Solution

STM32 GUI 平台介绍

- STM32图像显示过程
- LCD接口(DBI/DPI/DSI)
- 图形加速(ChromART & ChromGRC)

STM32图像显示过程

LCD接口(DBI/DPI/DSI)

图形加速(ChromART & ChromGRC)

智能家居应用

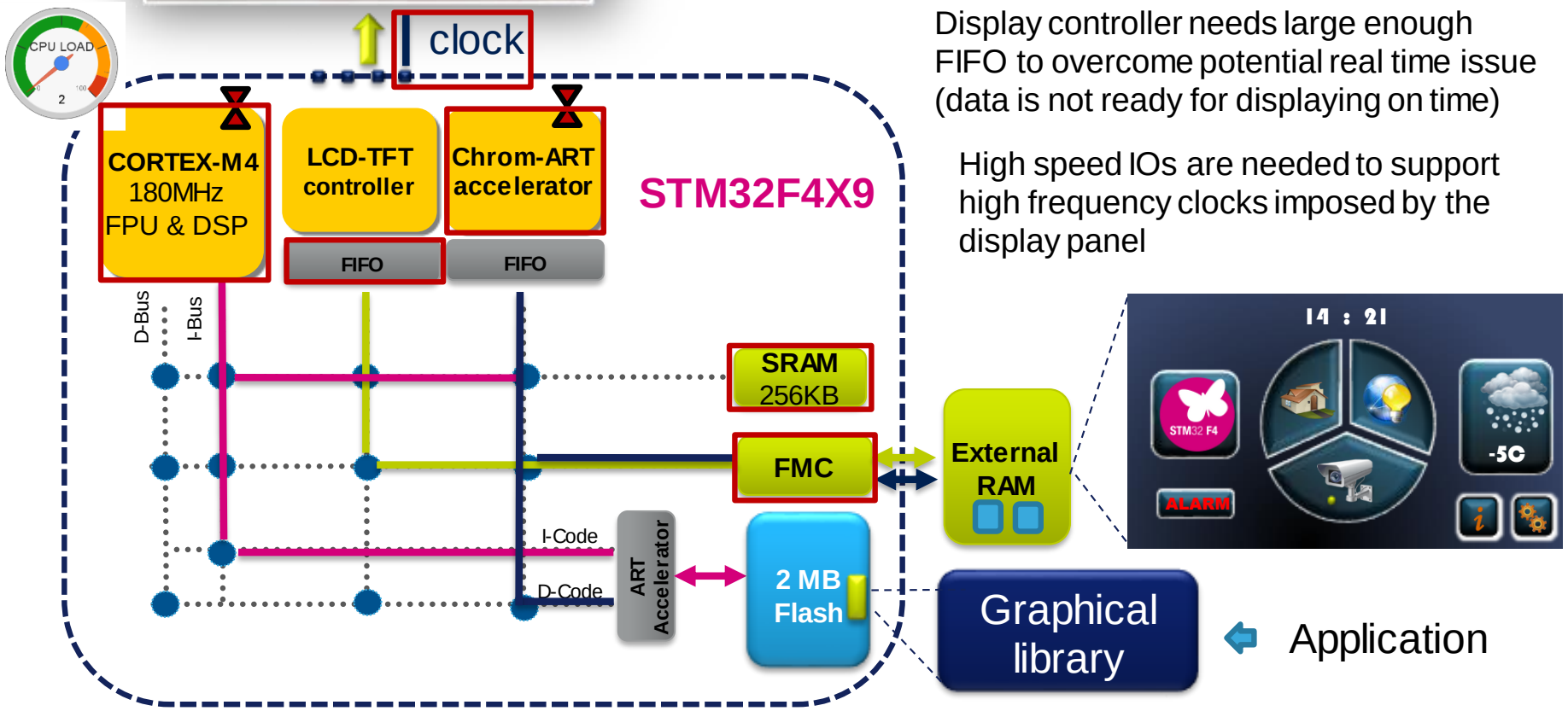
System architecture challenges

Graphical acceleration offloads the CPU enabling quality GUI **AND** high performance applications

For enhanced GUIs, Memory bandwidth is key. For low-mid resolutions SRAM can be used and so internal buses must support high bandwidth. For High resolution, a high performance external memory interface is a must

Display controller needs large enough FIFO to overcome potential real time issue (data is not ready for displaying on time)

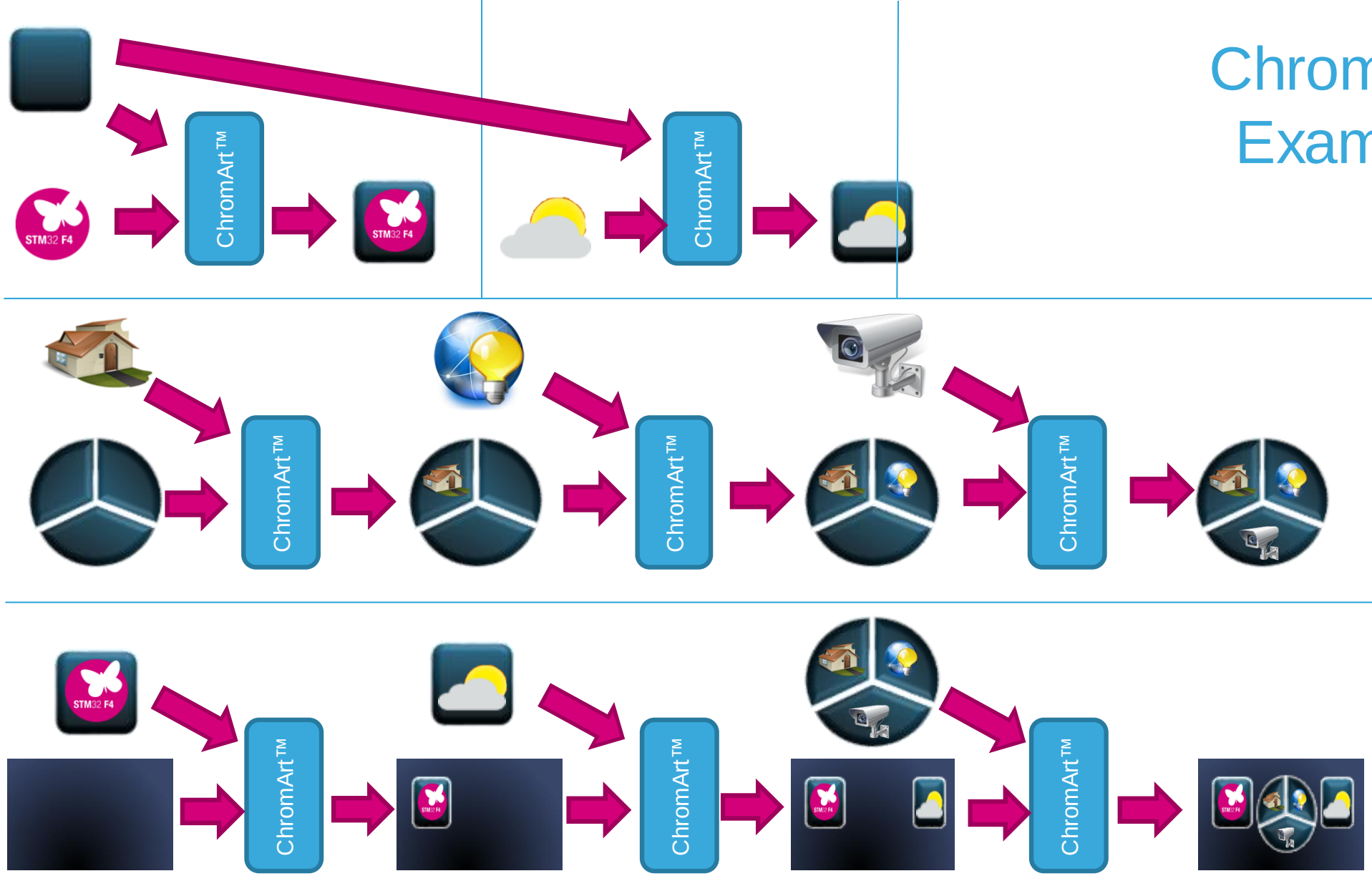
High speed IOs are needed to support high frequency clocks imposed by the display panel



14 : 21
STM32 F4
ALARM
-5C

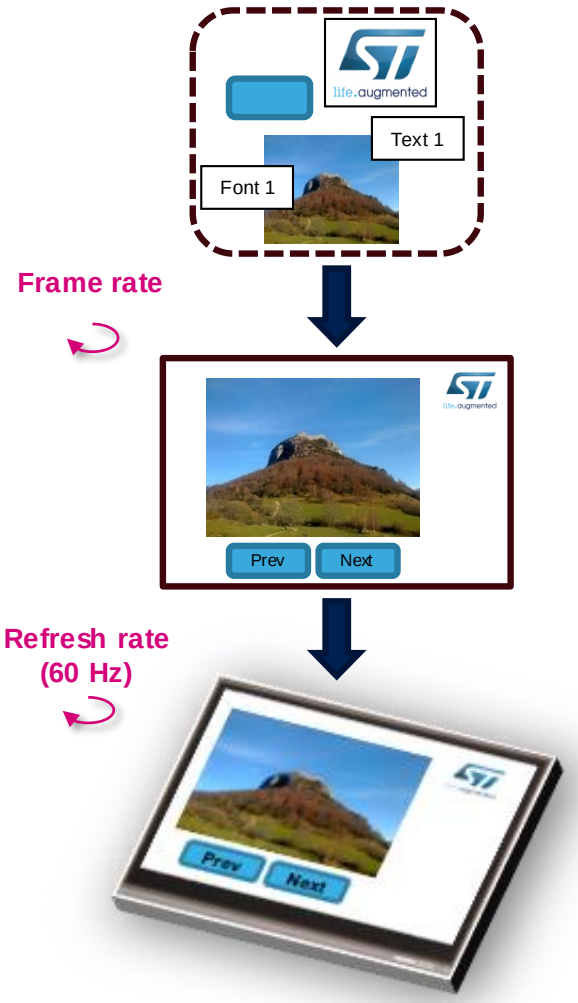
life.augmented

ChromArt Example



ChromART™ efficiently offload CPU from graphic tasks

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

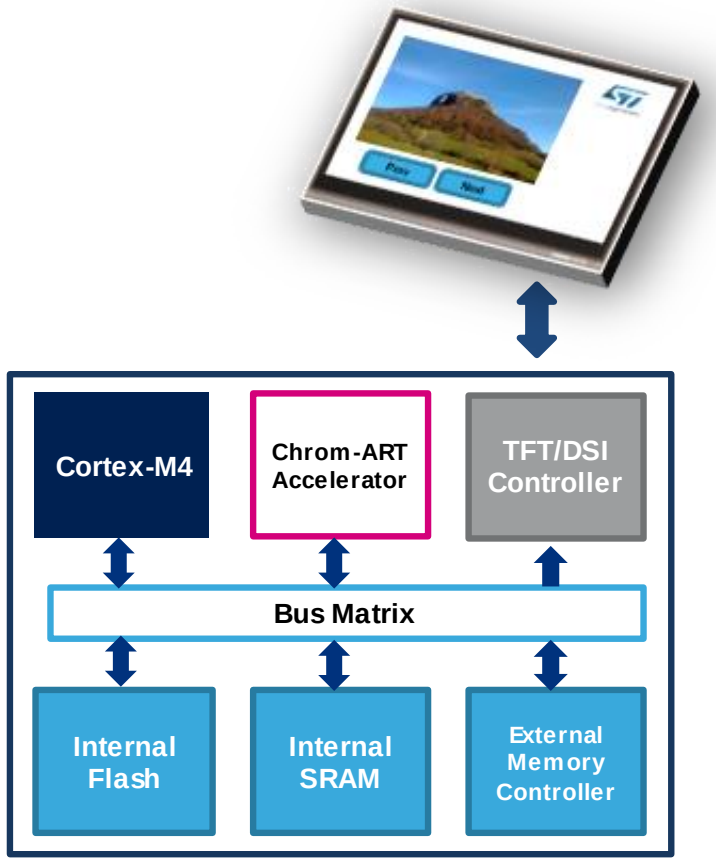
- The frame buffer is transferred to the display through a dedicated hardware interface
- Graphical oriented microcontroller are offering a **TFT controller** to drive directly the display
- The frame buffer must be sent at 60fps to have a perfect image color and stability (independently from the animation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

Single Chip MCU

- Internal Flash up to 2MB
- Internal SRAM up to 512KB
  - Frame buffer in internal SRAM
    - 16-bit QVGA (400 x 400 ~320 KB)
- Package **LQFP 100pin**

Cost saving + Graphic Acceleration

16-bit QVGA, 8-bit WQVGA



STM32F4x9

MCU with External Memory

- Internal Flash up to 2MB
- Internal SRAM up to 512KB
- External Memory for frame buffer
  - 16-bit or 32-bit SDRAM / SRAM
- Package: **LQFP 144pin,** up to 208.

Unique Graphical Capability and Flexible architecture

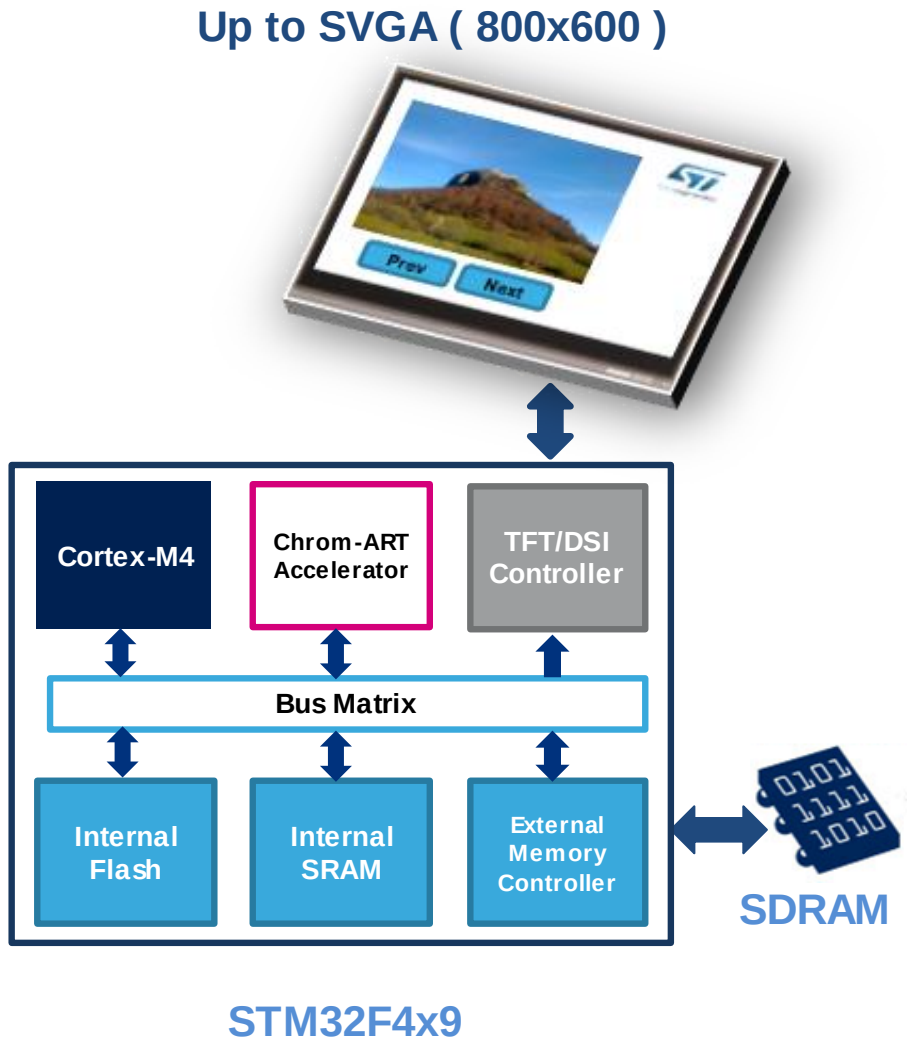


Image Building process

Using STM32F4x7/4X9 internal RAM (192K) for small resolutions

BUFFER SIZES (Kbytes) ↘	CGA (320x200)	QVGA (320x240)	WQVGA (480x272)	VGA (640x480)	WVGA (800x480)	SVGA (800x600)	XGA (1024x768)
1 (2 colors)	7,8	9,4	15,8	37,5	46,9	58,6	96,0
2 (4 colors)	15,6	18,8	31,6	75,0	93,8	117,2	192,0
4(16 colors)	31,3	37,5	63,3	150,0	187,5	234,4	384,0
8 (256 colors)	62,5	75,0	126,6	300,0	375,0	468,8	768,0
16 (high color)	125,0	150,0	253,1	600,0	750,0	937,5	1536,0
24 (true color)	187,5	225,0	379,7	900,0	1125,0	1406,3	2304,0
32 (deep color)	250,0	300,0	506,3	1200,0	1500,0	1875,0	3072,0

bpp ↓

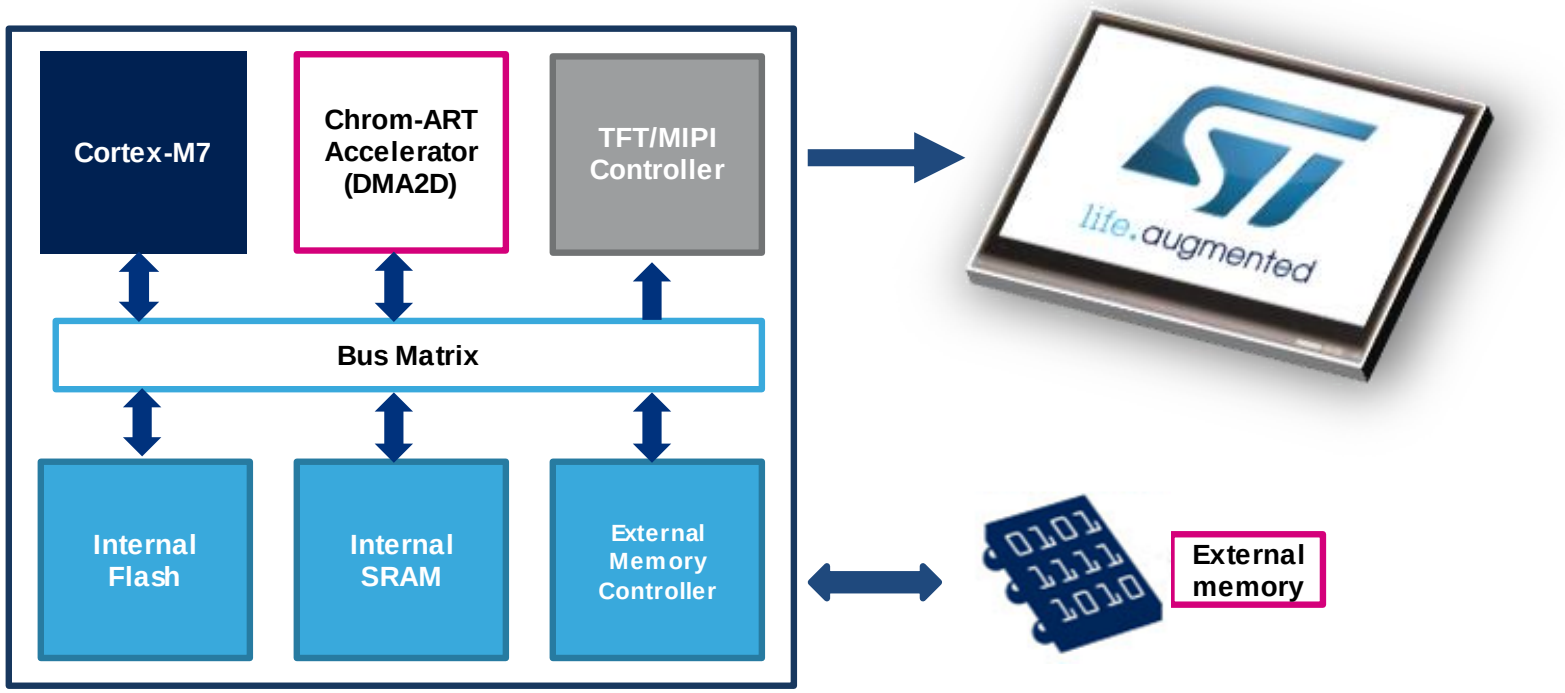
Double buffer: New image and Frame buffers can be in internal RAM

Single buffer : 2 options:

- Image can be handled by regions: one region to be displayed while the other is been built and so only one buffer will be needed
- new image buffer in internal RAM and frame buffer in external RAM or in display internal memory.

External memory needed for both frame buffer and new image buffer

STM32F7x9 举例

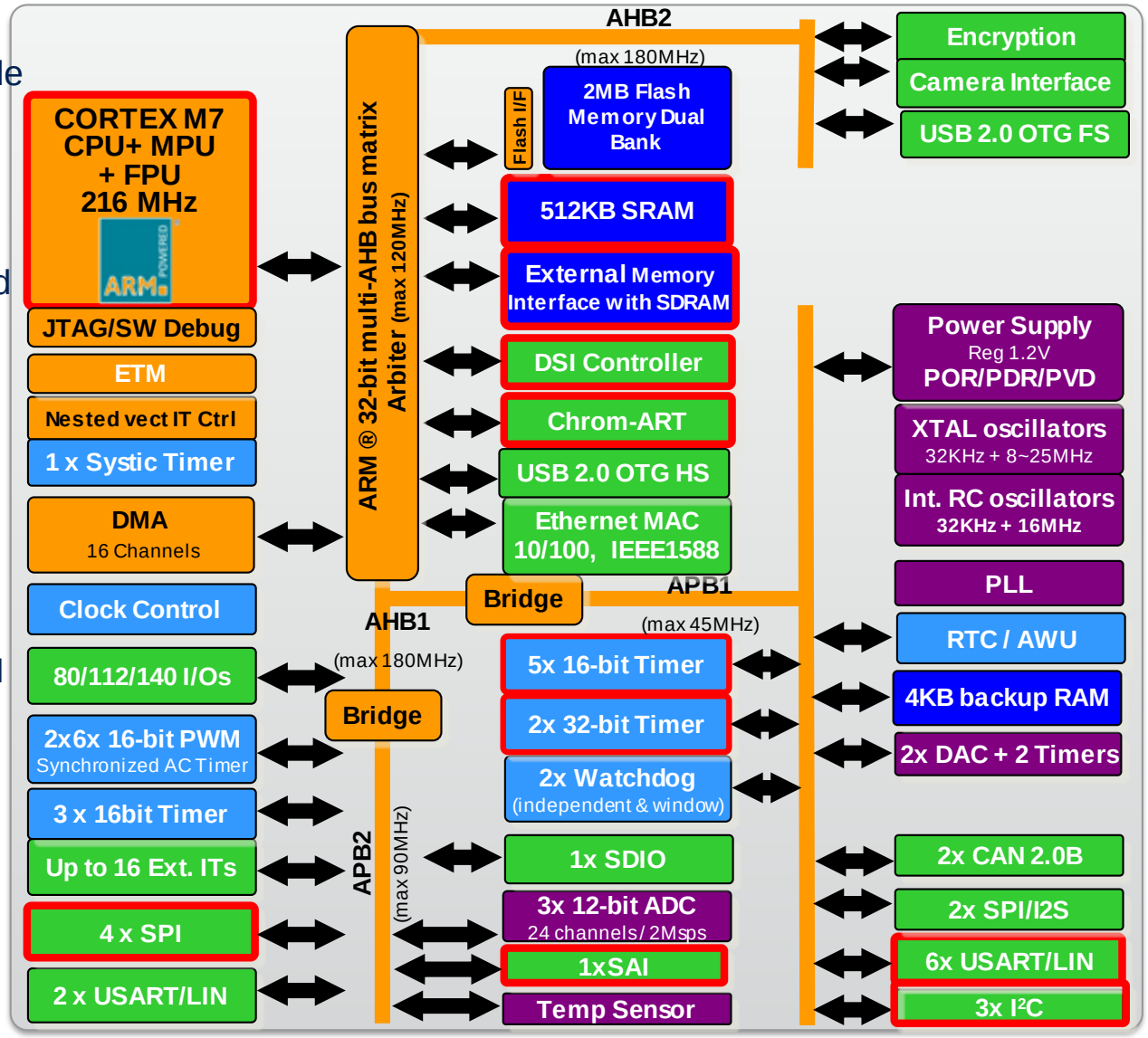


- **TFT controller** allows the interfacing
- **Chrome-ART** (DMA2D) provides a **true HW graphical acceleration**
- **DMA2D offloads the CPU** for operations like rectangle filling, rectangle copy (with or without pixel format conversion), and image blending
- **DMA2D goes faster than the CPU** for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

STM32F7x9 Block Diagram

- Fully compatible with F4x
- Up to 216MHz with over-drive mode
- Dual Bank 2 x 1MB Flash
- 512KB SRAM
- FMC with SDRAM + Support and 32-bit data
- Audio PLL + Serial Audio I/F
- LCD-TFT Controller
- LCD-DSI 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

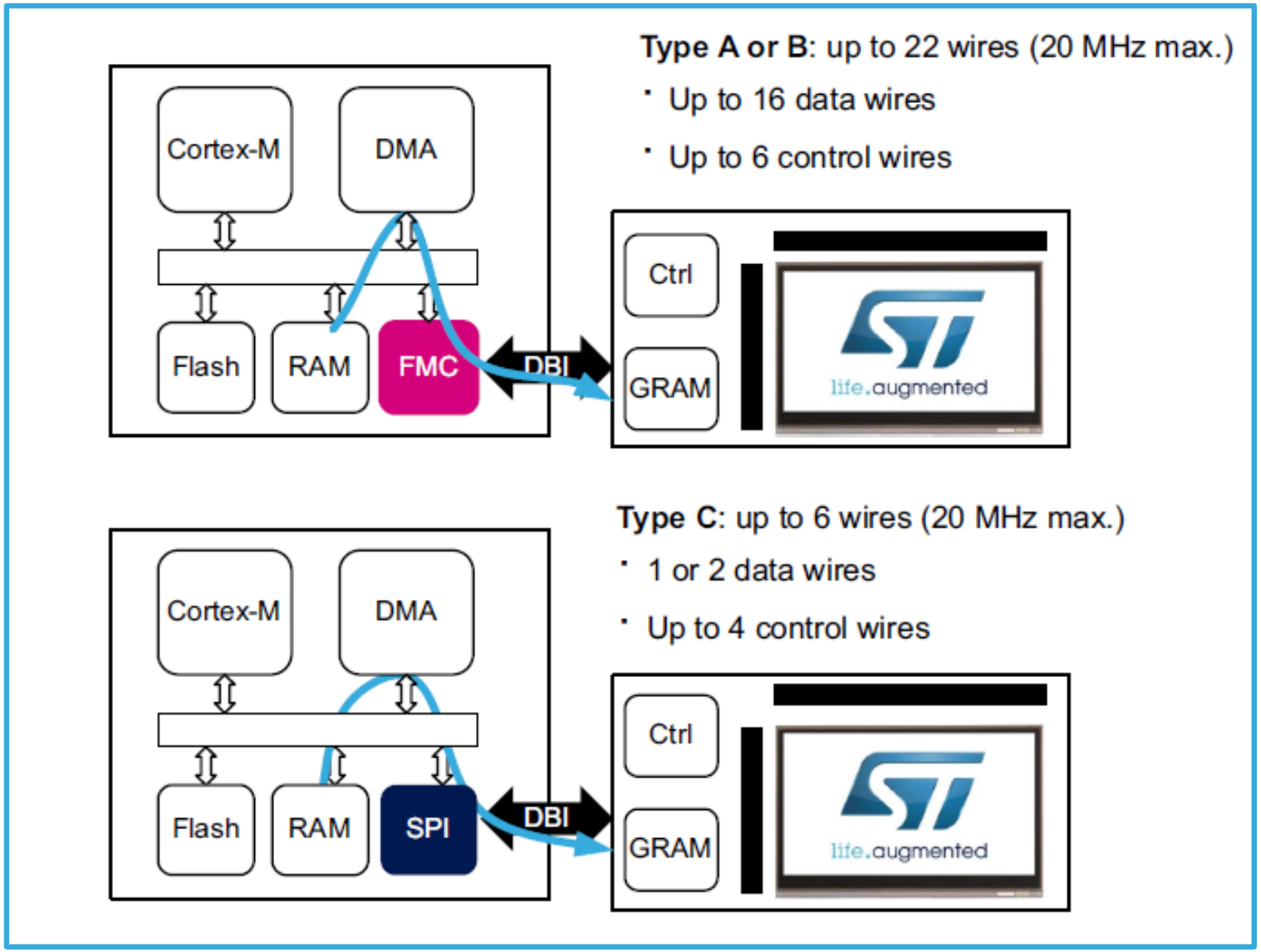


life.augmented

STM32 GUI 平台介绍

- STM32图像显示过程
- LCD接口(DBI/DPI/DSI)
- 图形加速(ChromART & ChromGRC)
- STM32CubeMX GUI Features

DBI

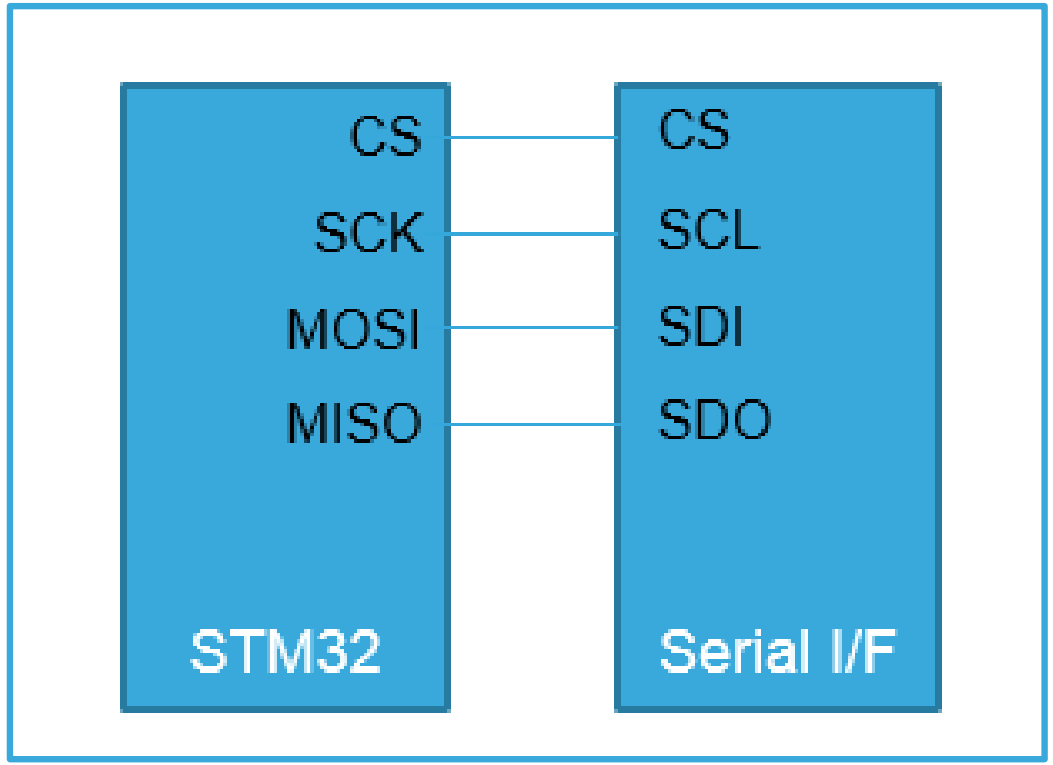


DBI-SPI

DBI-FMC/FSMC

DBI-SPI接口

DBI接口

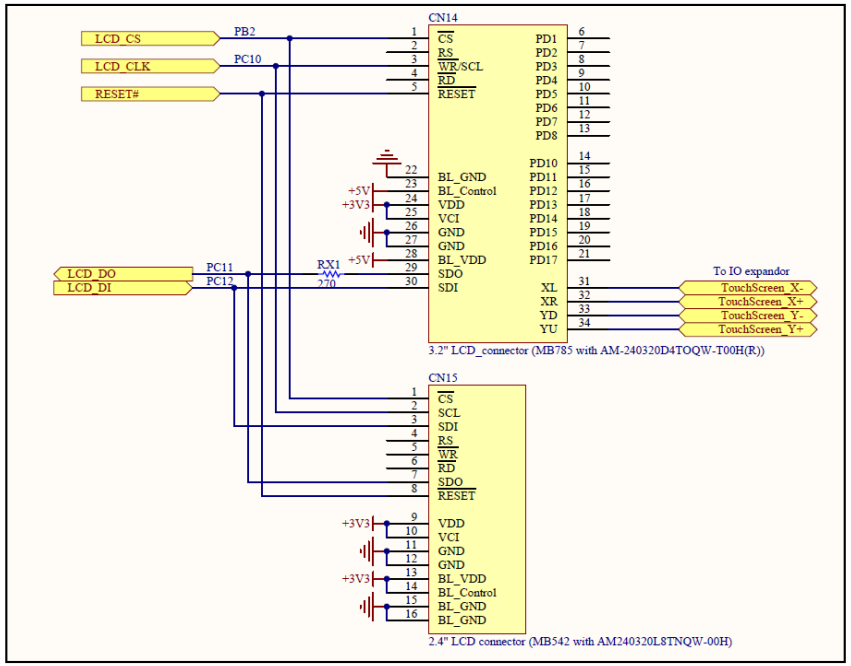


SPI

STM32 SPI Interfacing Example

SPI can be used for serial interfacing

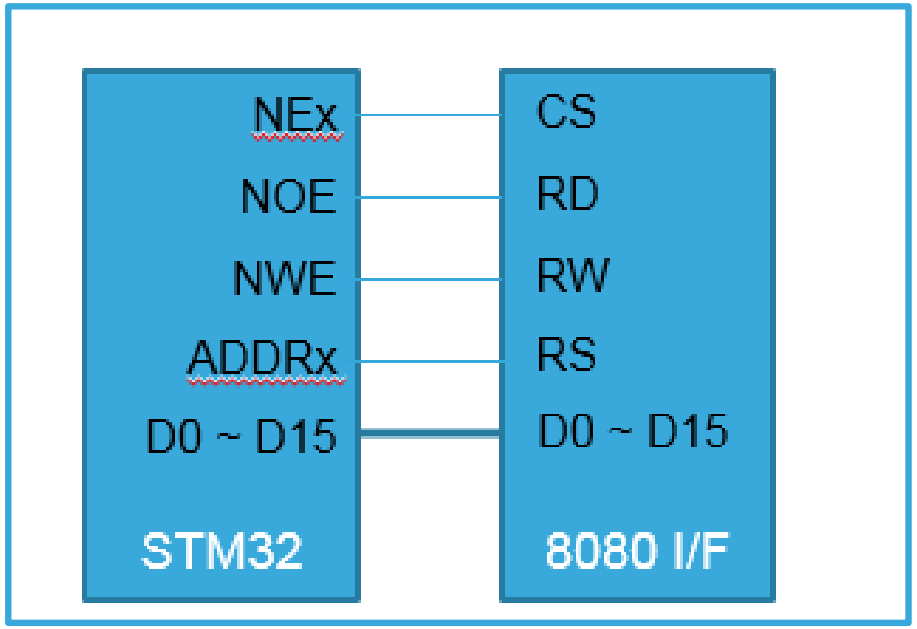
- **SPI is the serial way of being interfaced with a display having a controller and a GRAM.**
  - Data organization (8/9/16/18-bit) may differ from a display to another.



From STM3210C Evaluation Board – MB784

DBI-FMC/FSMC接口

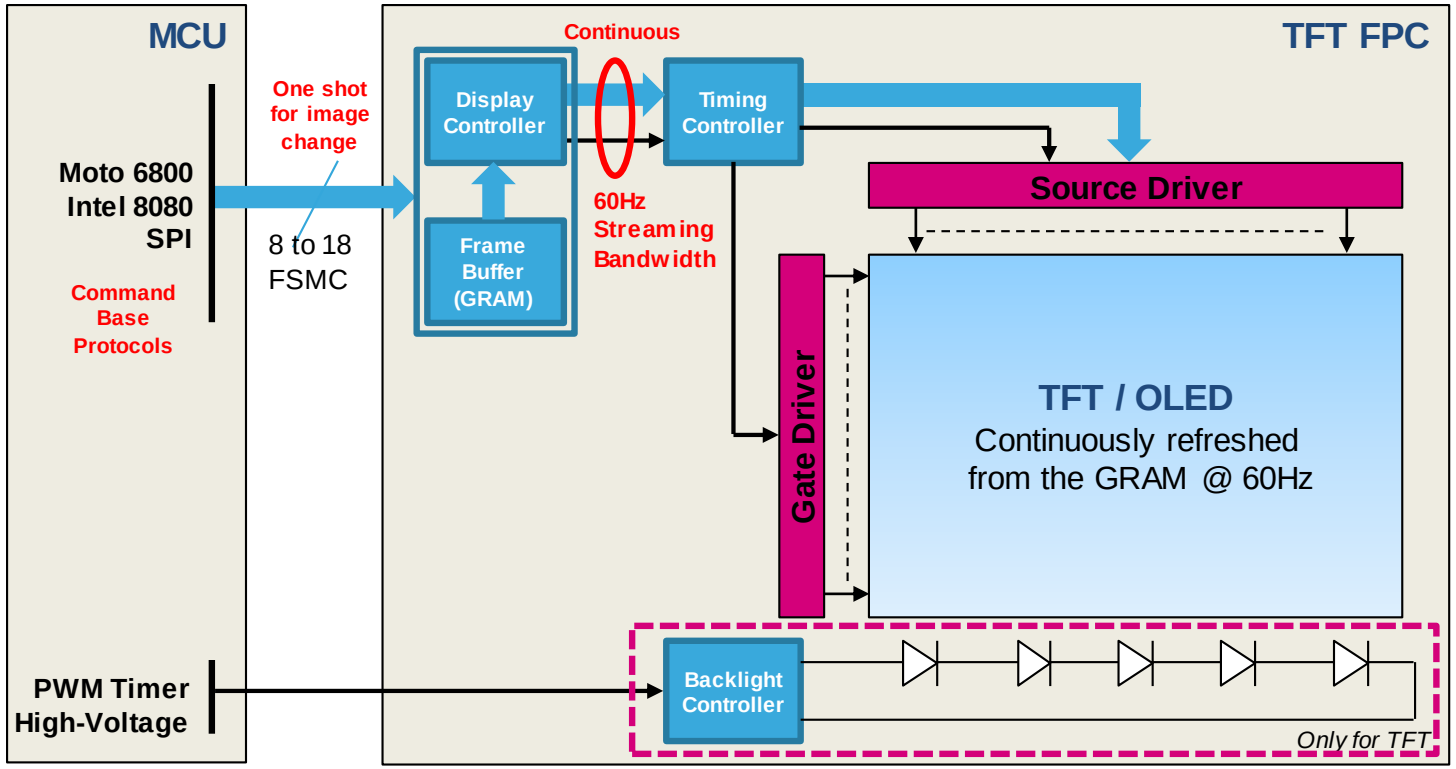
DBI接口



FSMC/FMC

Display with Controller and GRAM

One shot command to write the GRAM are sent by the MCU

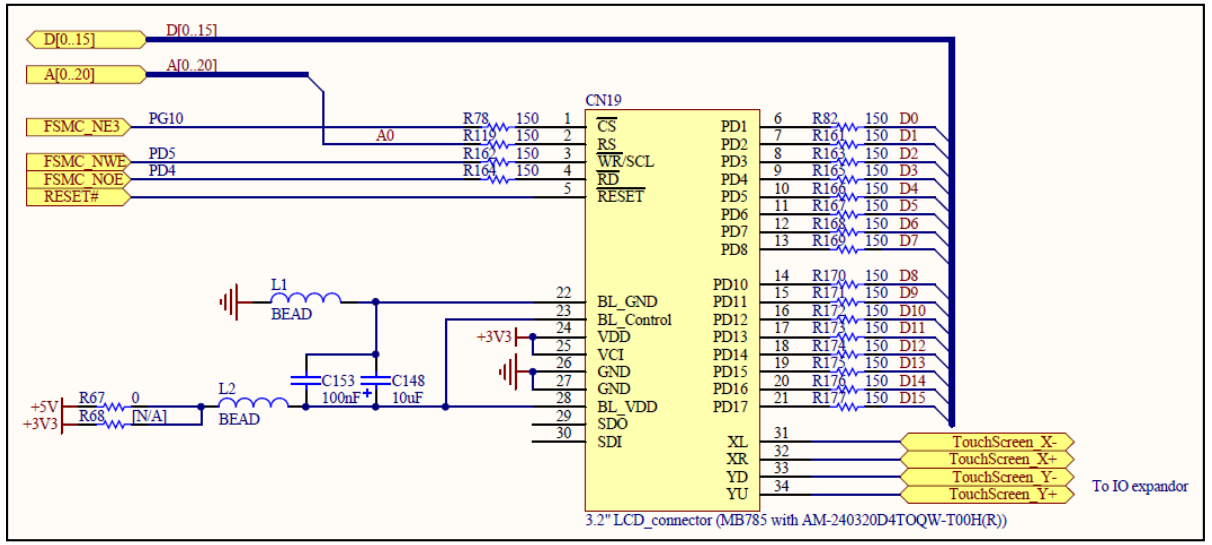


STM32 F(S)MC support Motorola 6800 and Intel 8080 : Reference: Application Note AN2790

STM32 FSMC I/F Connection

FSMC can be used for parallel interfacing (Intel or Motorola)

- **Interfacing using intel 8080 mode**
  - D/CX is done thanks to an address line
  - \WR and \RD done by NWE and NOE respectively



From STM32 20-21-45-46 G Evaluation Board – MB786

Sequence Example

- Sequence example
  - Write & Read sequence for Type B
  - Timing define

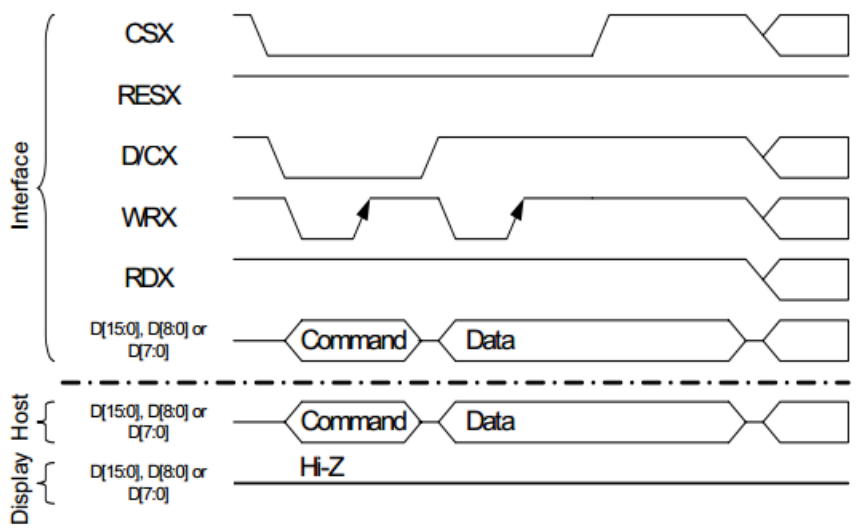


Figure 19 Type B Interface Write Sequence

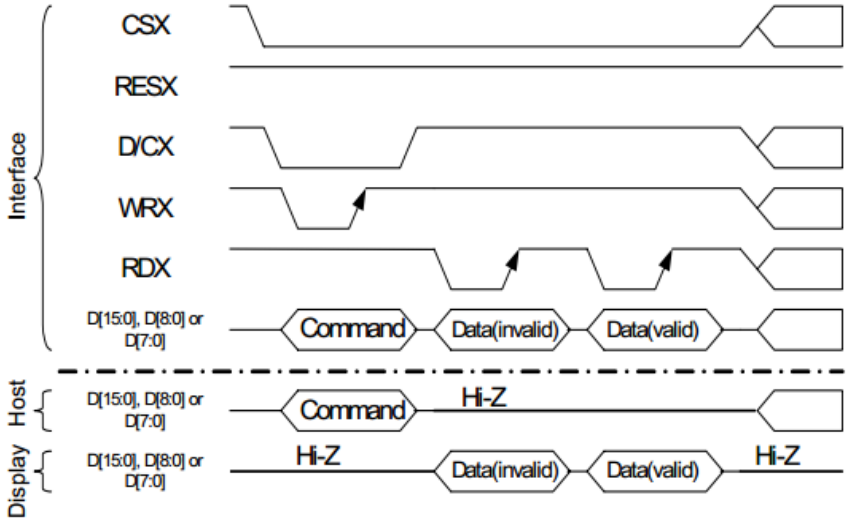
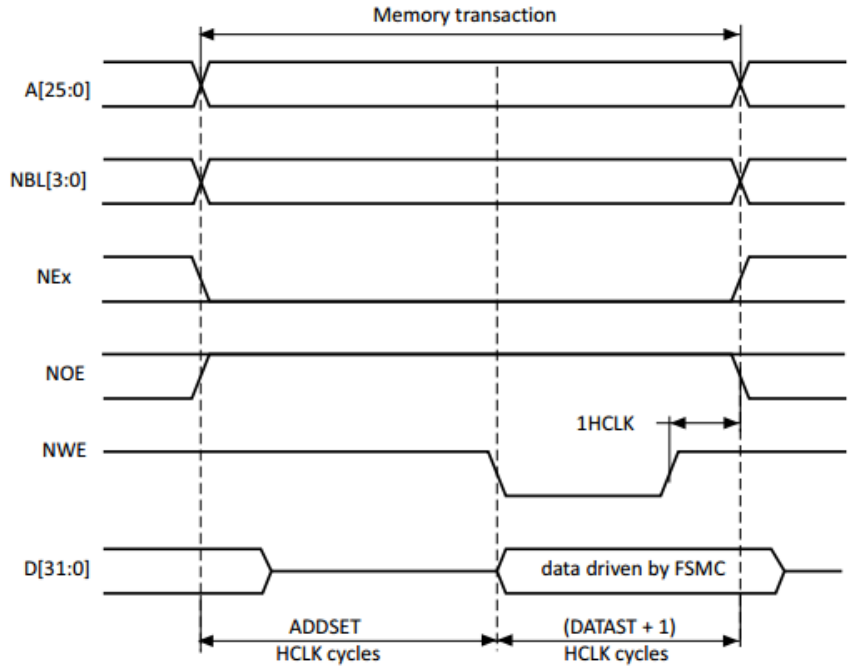
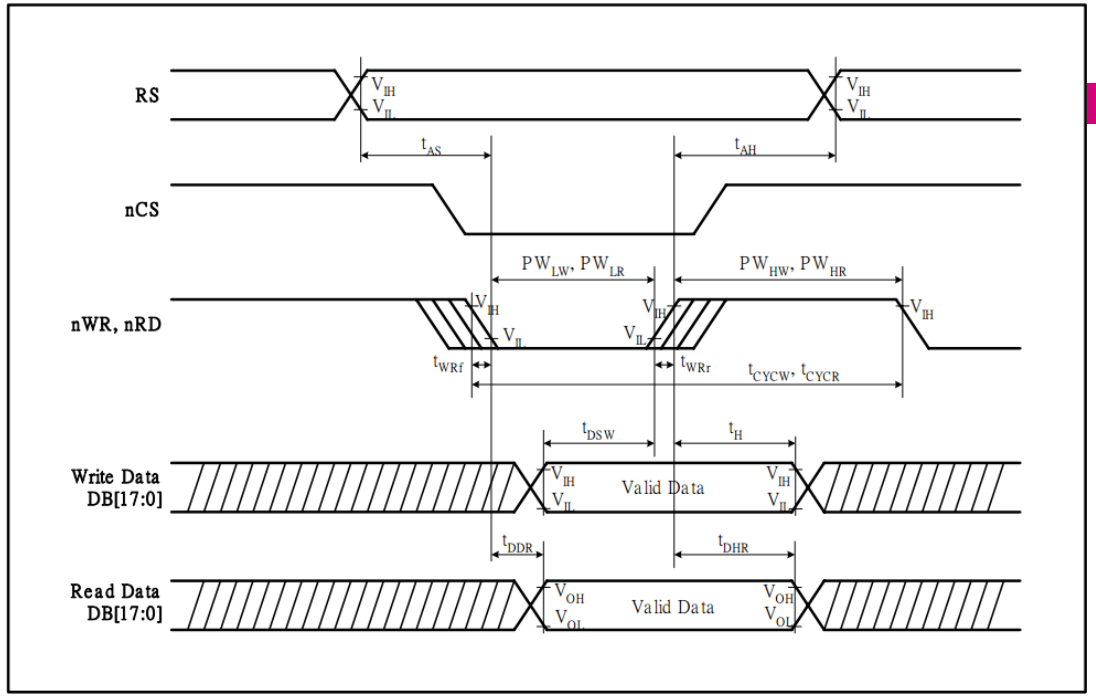


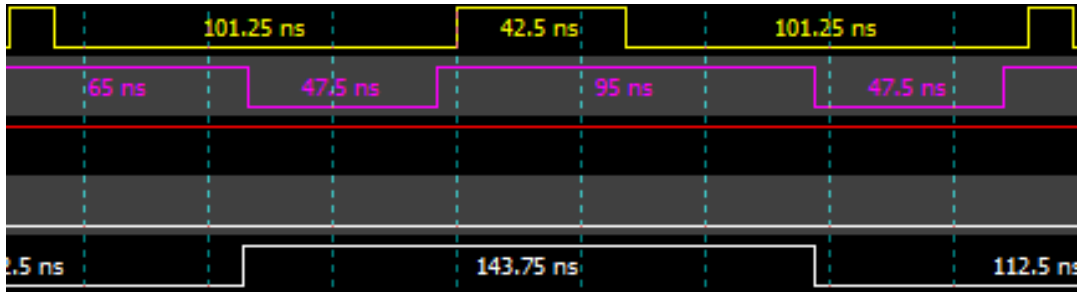
Figure 20 Type B Interface Read Sequence

Figure 459. ModeA write access waveforms





- With CPU running at 168MHz
  - (DATAST + 1)*HCLK > 50ns
    - DATAST = 8 (7.4)
  - (ADDSET + DATAST + 1)*HCLK > 100ns
    - ADDSET = 8 (7.8)
  - Verify NWE high period more than tH (10ns)

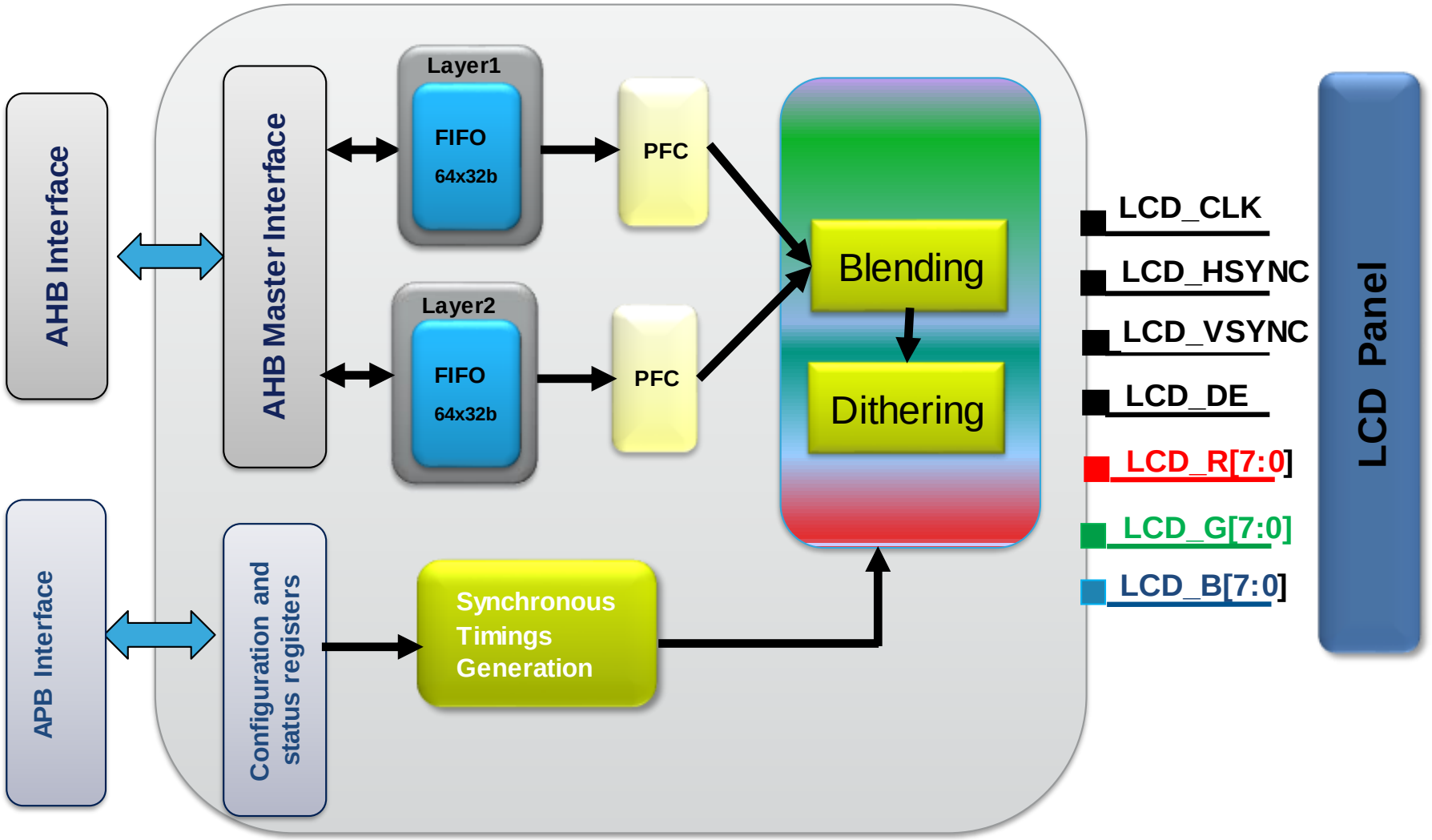


Item		Symbol	Unit	Min.	Typ.	Max.
Bus cycle time	Write	t_{CYCW}	ns	100	-	-
Bus cycle time	Read	t_{CYCR}	ns	300	-	-
	Write low-level pulse width	PW_{LW}	ns	50	-	-
	Write high-level pulse width	PW_{HW}	ns	50	-	-
	Read low-level pulse width	PW_{LR}	ns	150	-	-
	Read high-level pulse width	PW_{HR}	ns	150	-	-
	Write / Read rise / fall time	t_{WRr}/t_{WRf}	ns	-	-	25
Setup time	Write (RS to nCS, E/nWR)	tAS	ns	10	-	-
	Read (RS to nCS, RW/nRD)			5	-	-
	Address hold time	t_{AH}	ns	5	-	-
	Write data set up time	t_{DSW}	ns	10	-	-
	Write data hold time	t_H	ns	15	-	-
	Read data delay time	t_{DDR}	ns	-	-	100
	Read data hold time	t_{DHR}	ns	5	-	-

life.augmented

DPI-LTDC接口

LCD-TFT architecture



life.augmented

LTDC引脚接口

Figure 12. LTDC signal interface

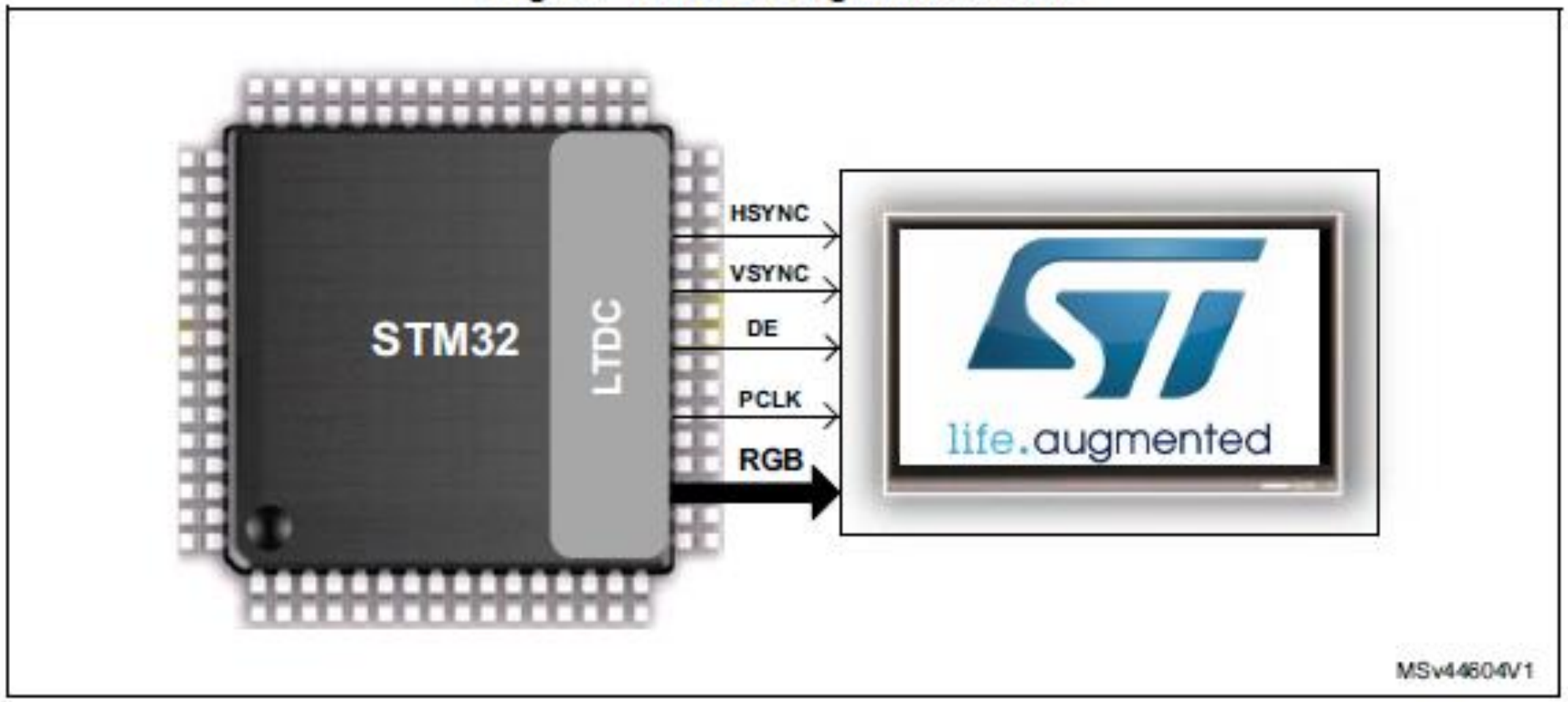


Figure 34. LCD-TFT connection in the STM32F746G-DISCO board

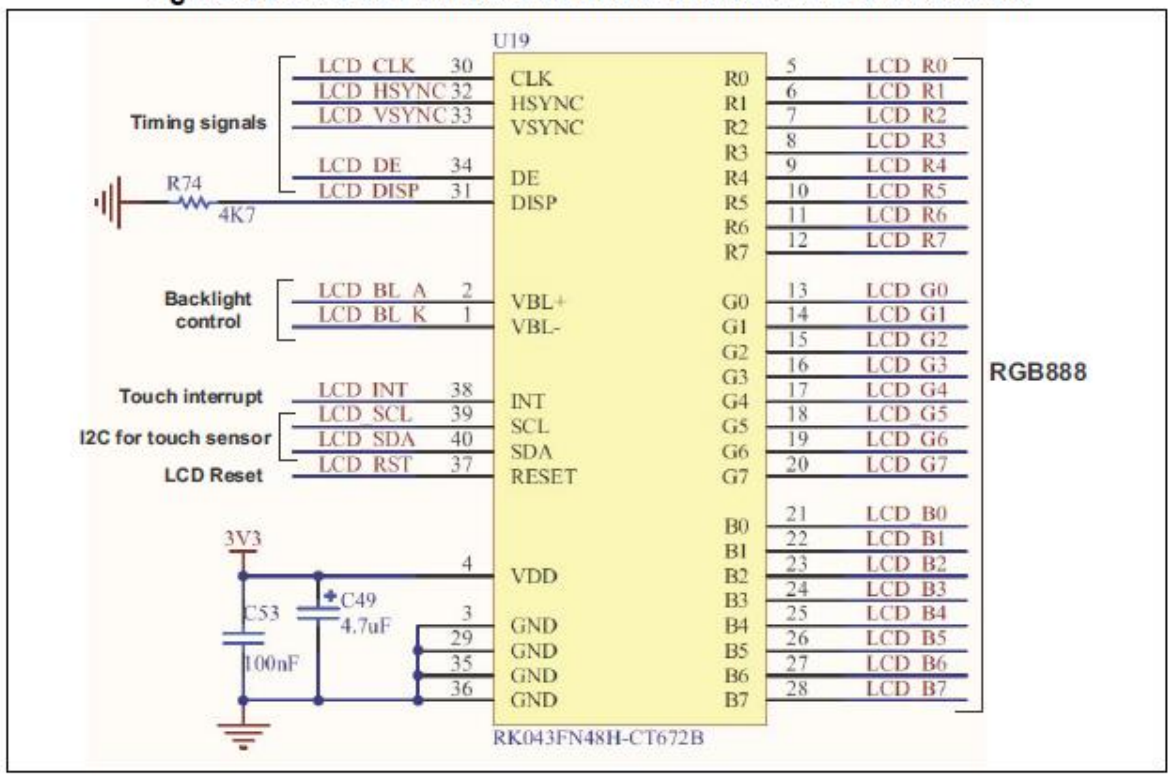
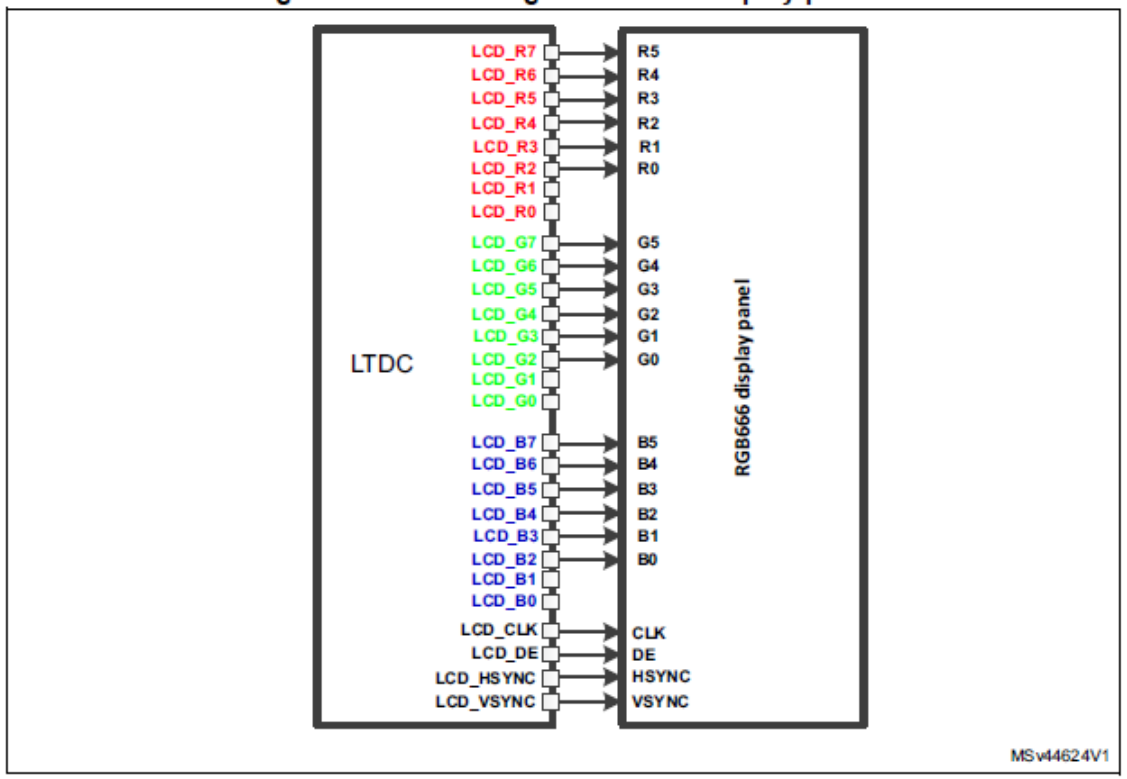


Figure 30. Connecting an RGB666 display panel



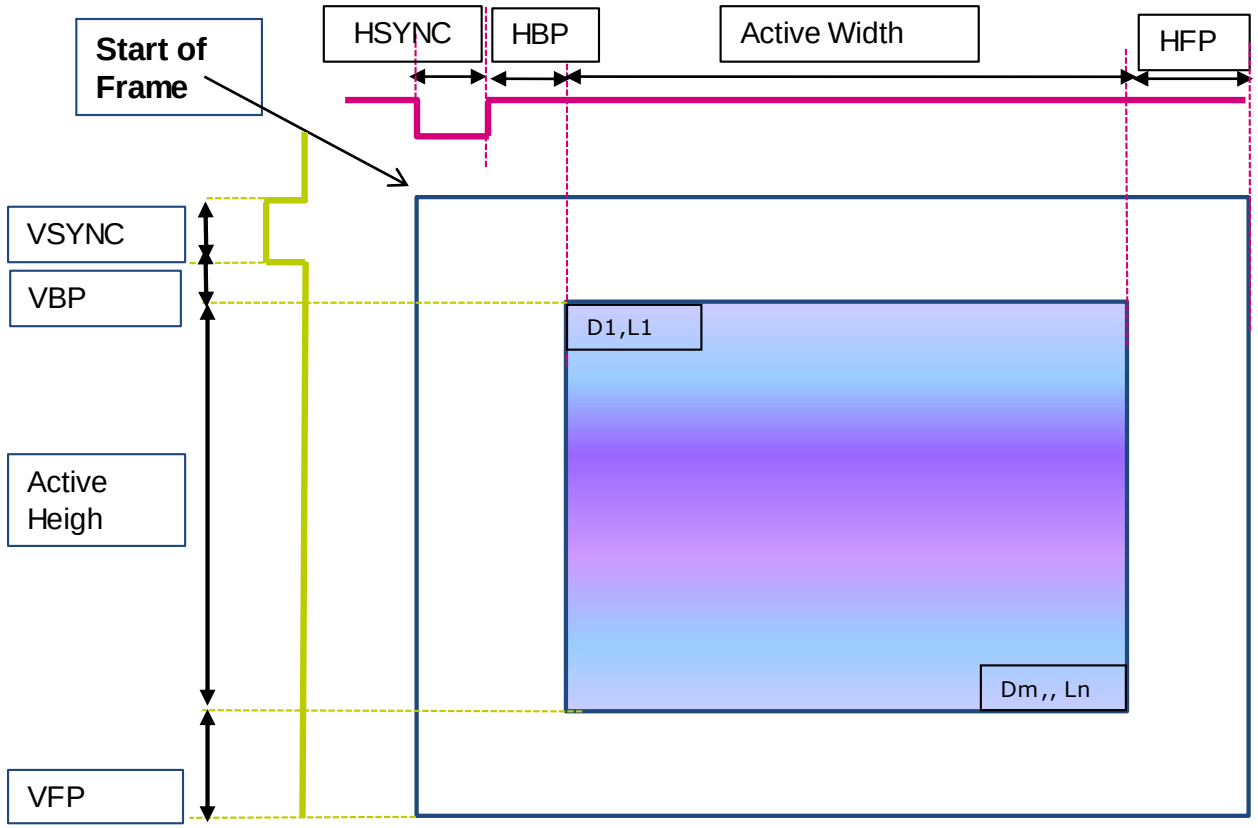
LCD-TFT Signals

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

Table 5. LTDC interface output signals .

LCD-TFT signal	Description
LCD_CLK	The LCD_CLK acts as the data valid signal for the LCD-TFT. The data is considered by the display only on the LCD_CLK rising or falling edge.
LCD_HSYNC	The line synchronization signal (LCD_HSYNC) manages horizontal line scanning and acts as line display strobe.
LCD_VSYNC	The frame synchronization signal (LCD_VSYNC) manages vertical scanning and acts as a frame update strobe.
LCD_DE	The DE signal, indicates to the LCD-TFT that the data in the RGB bus is valid and must be latched to be drawn.
Pixel RGB data	The LTDC interface can be configured to output more than one color depth. It can use up to 24 data lines (RGB888) as display interface bus.

LTDC Timings



VBP: Vertical Back porch

VFP: Vertical Front porch

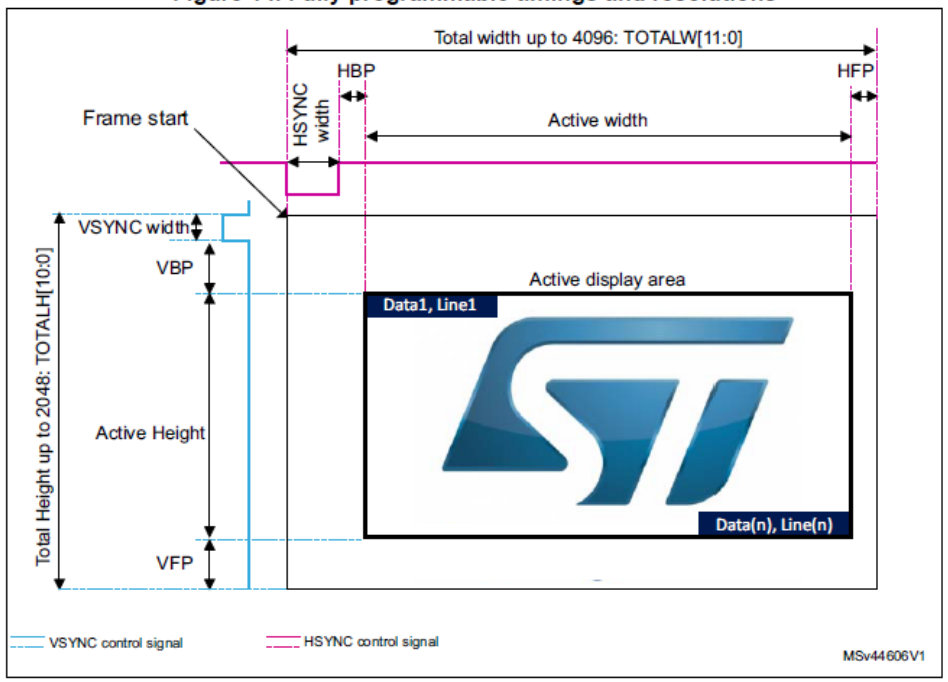
HBP: Horizontal Back porch

HFP: Horizontal Front porch

These timings come from the datasheet of the display. E.g. TFT on discovery kit.

Parameters	Symbols	Condition	Min.	Typ.	Max.	Units
Horizontal Synchronization	Hsync		2	10	16	DOTCLK
Horizontal Back Porch	HBP		2	20	24	DOTCLK
Horizontal Address	HAdr		-	240	-	DOTCLK
Horizontal Front Porch	HFP		2	10	16	DOTCLK
Vertical Synchronization	Vsync		1	2	4	Line
Vertical Back Porch	VBP		1	2	-	Line
Vertical Address	VAdr		-	320	-	Line
Vertical Front Porch	VFP		3	4	-	Line

Figure 14. Fully programmable timings and resolutions

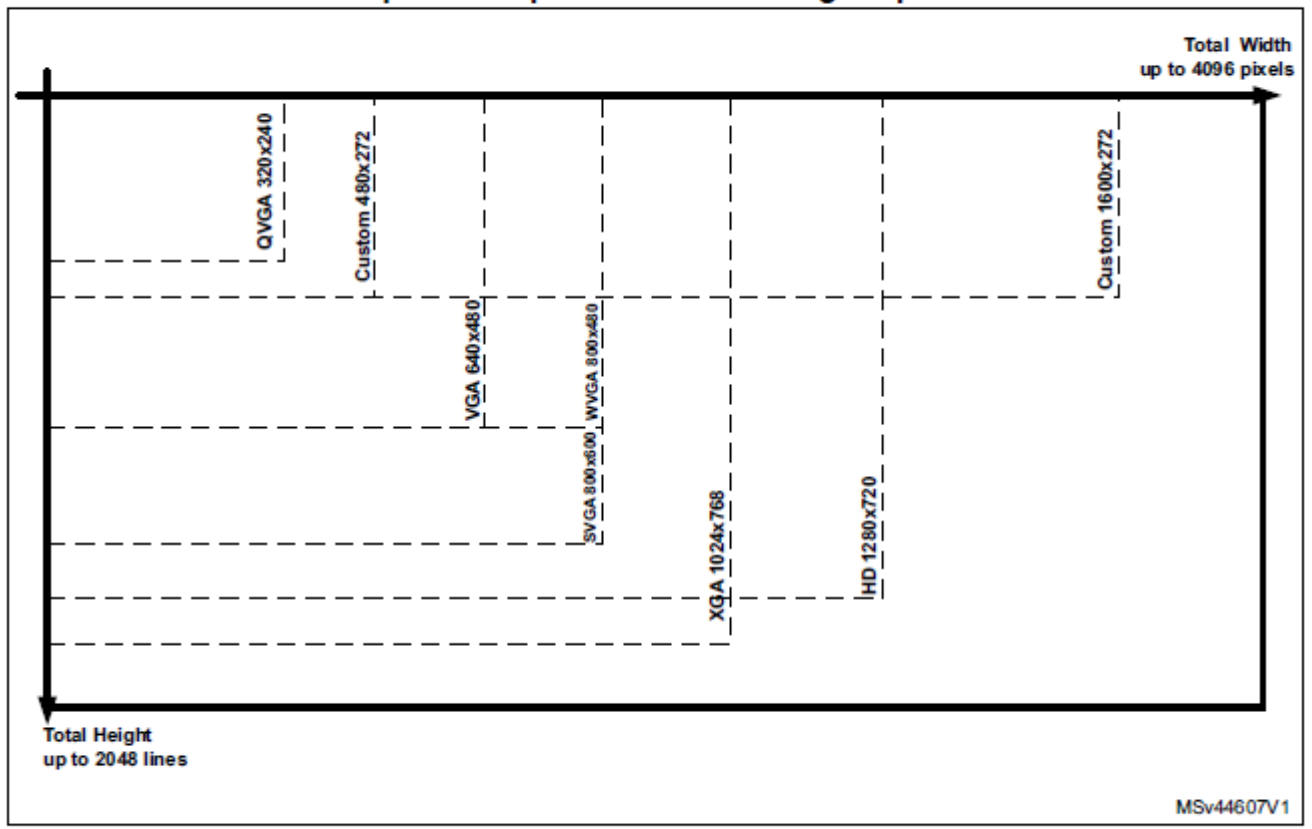


Any display resolution belonging to the maximal total area in 4096 x 2048 as described in *Figure 15* is supported by the LTDC only if the following conditions are met:

- The display panel pixel clock must not exceed the maximal LTDC pixel clock in *Table 2*
- The display panel pixel clock must not exceed the maximal STM32 pixel clock respecting the framebuffer bandwidth (see *Section 4.2: Checking the display size and color depth compatibility with the hardware configuration*).

Figure 15 shows some custom and standard resolutions belonging to the maximal 4096 x 2048 supported by the LTDC.

Figure 15. LTDC fully programmable display resolution with total width up to 4096 pixels and total height up to 2048 lines



1. Only the active display area is shown in this figure.

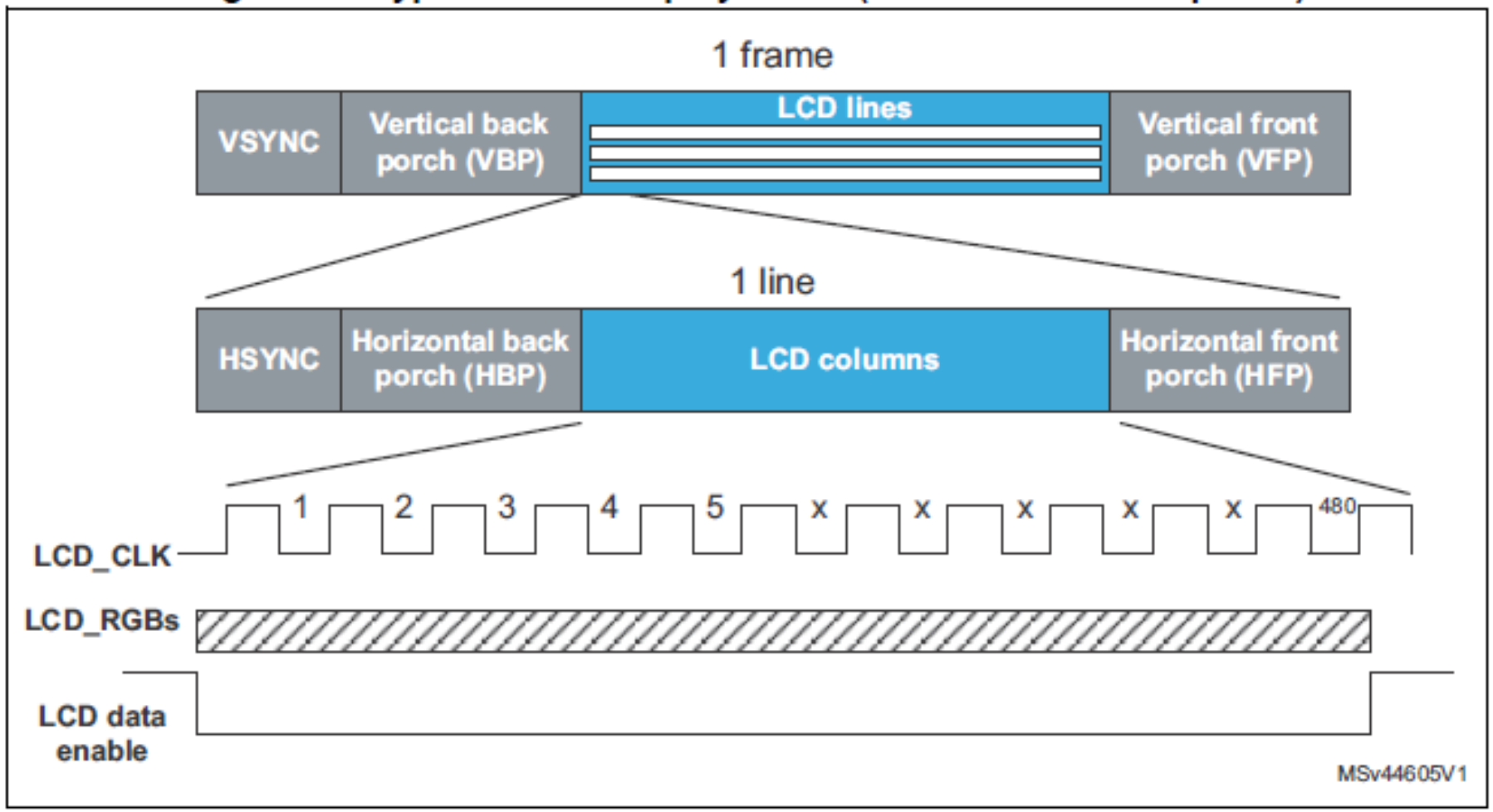
STM32 LTDC参数

Table 6. LTDC timing registers

Register		Timing parameter	Value to be programmed
LTDC_SSCR(1)	HSW[11:0]	HSYNC width - 1	From 1 to 4096 pixels
	VSH[11:0]	VSYNC height - 1	From 1 to 2048 lines
LTDC_BPCR	AHBP[11:0]	HSYNC width + HBP - 1	From 1 to 4096 pixels
	AVBP[10:0]	VSYNC height + VBP - 1	From 1 to 2048 lines
LTDC_AWCR	AAW[11:0]	HSYNC width + HBP + active width - 1	From 1 to 4096 pixels
	AAH[10:0]	VSYNC height + BVBP + active height - 1	From 1 to 2048 lines
LTDC_TWCR	TOTALW[11:0]	HSYNC width + HBP + active width + HFP - 1	From 1 to 4096 pixels
	TOTALH[10:0]	VSYNC height + BVBP + active height + VFP - 1	From 1 to 2048 lines

Frame Display

Figure 13. Typical LTDC display frame (active width = 480 pixels)

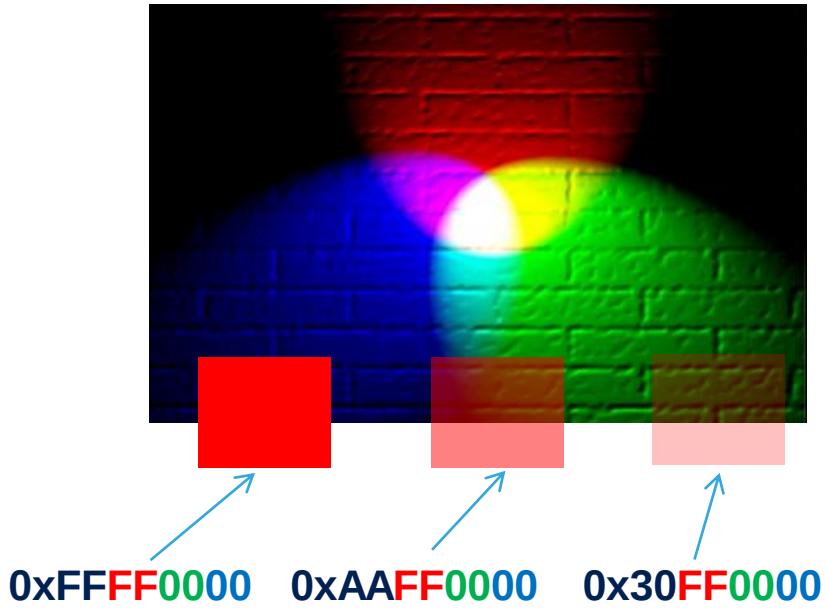


Alpha Channel Usage

- 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**

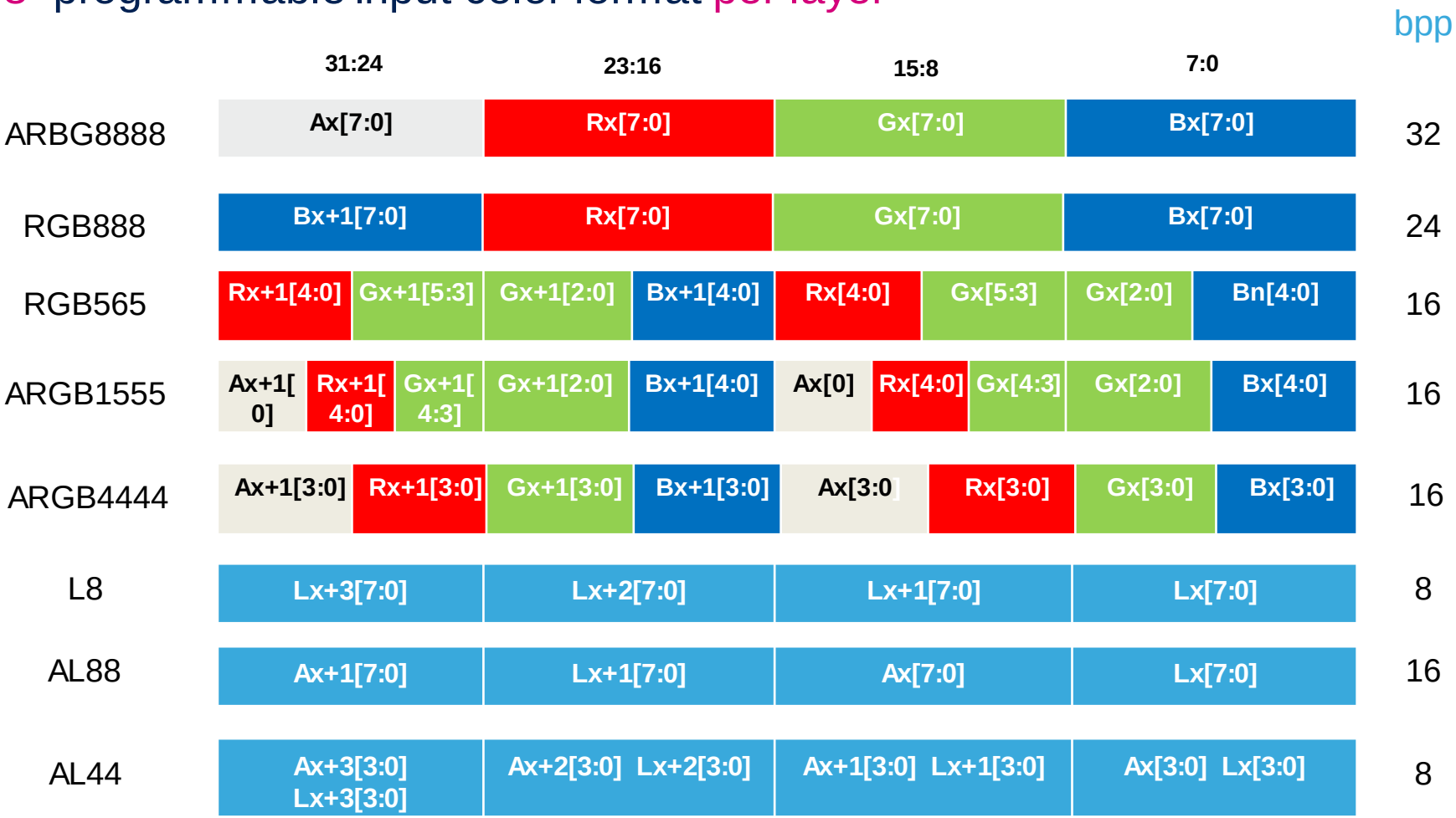
- **0xFF = opaque**
- 0x00 = trasnparent



Box filled with color in ARGB8888 format

Pixel Data Mapping vs Color Format

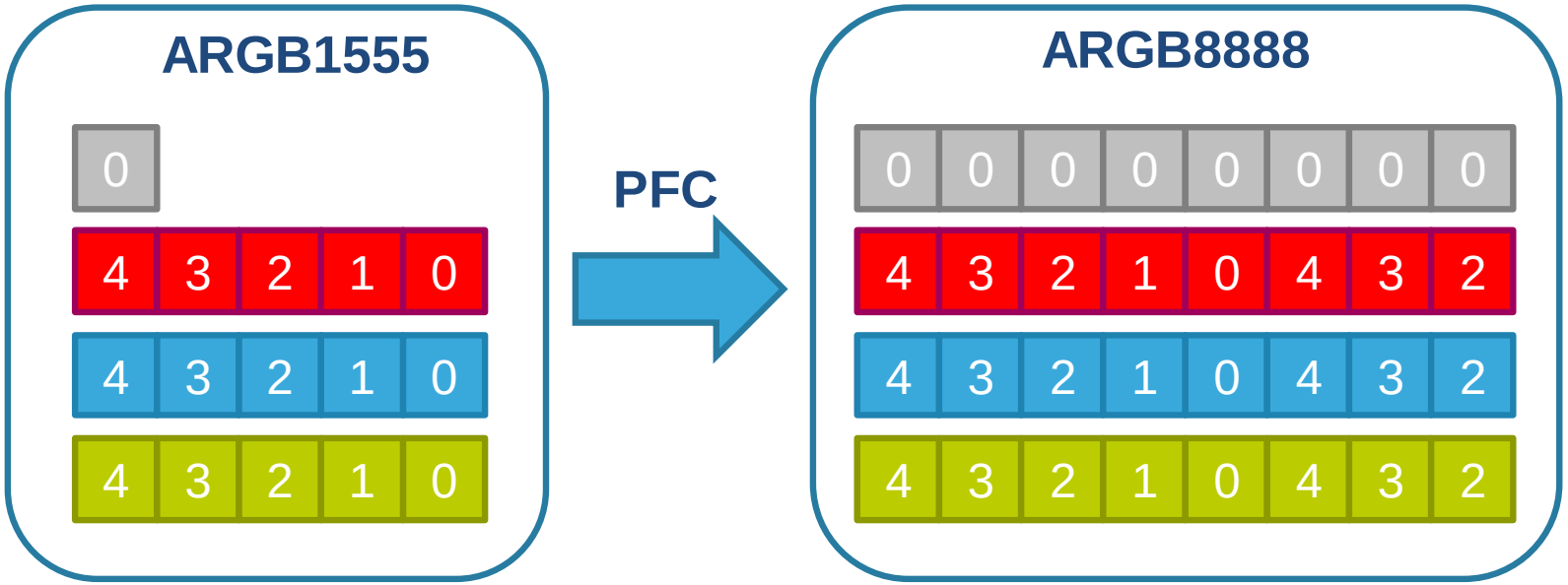
8 programmable Input color format per layer



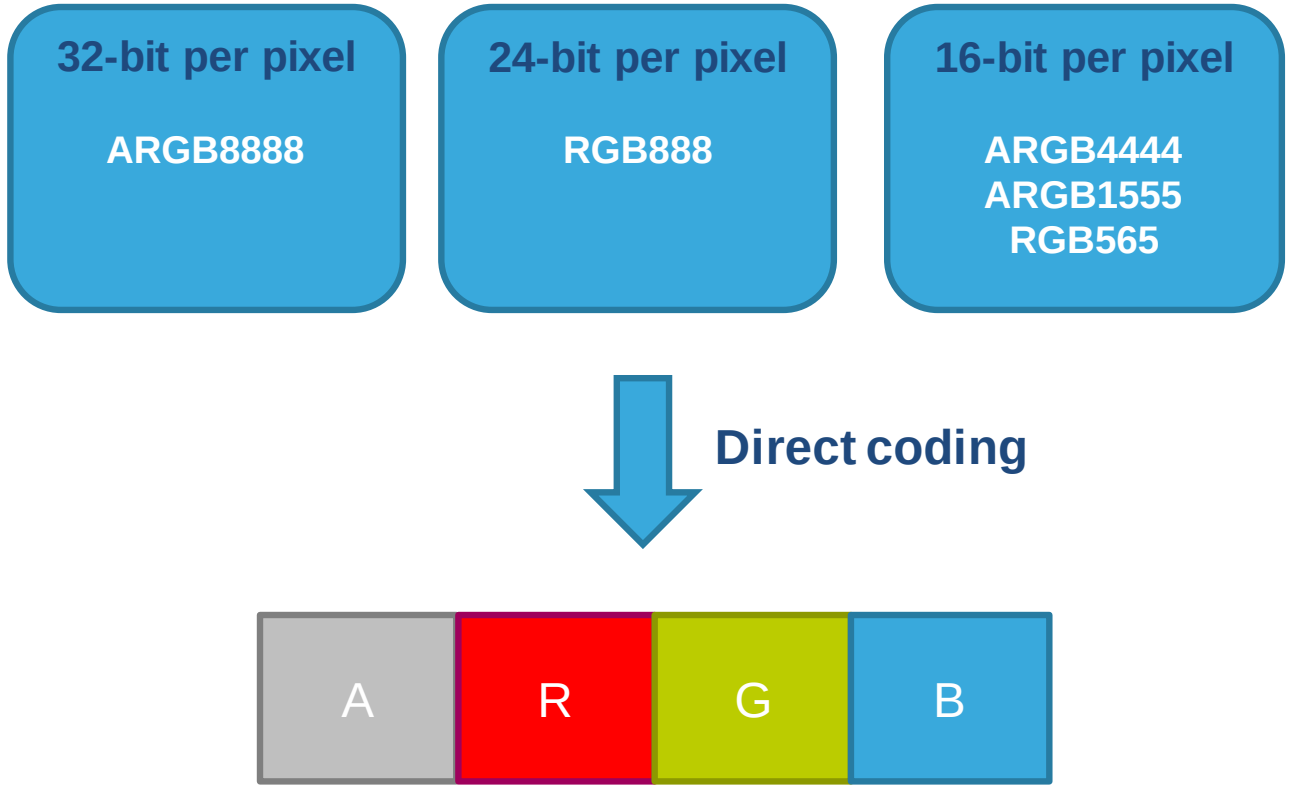
life.augmented

Pixel Format Conversion

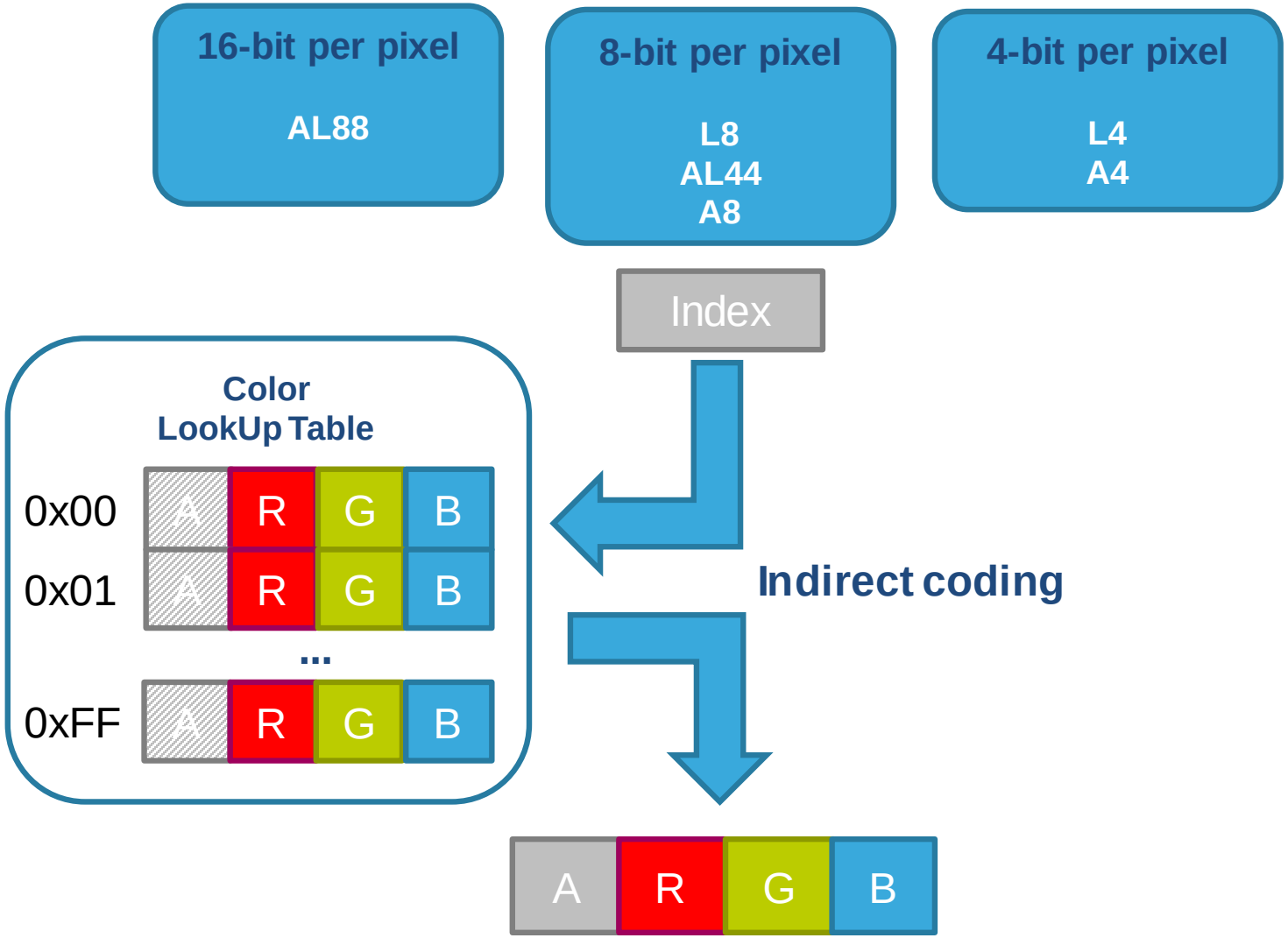
- The color format of a bitmap converted into another one : this operation is called Pixel Format Conversion (PFC)
- Direct Mode to Direct Mode or Indirect Mode to Direct Mode is easy to do, but converting to an indirect color format would mean regenerating a CLUT which is a very complex operation...



PFC - Direct color Mode



PFC - Indirect color Mode



CLUT - Palletized color

- Color Look-Up Table (CLUT) up to 256 entry per layer
- The frame buffer will contain an index value for each pixel
- The CLUT must be pre-loaded with RGB values for each index
  - Each RGB value is stored in a position in the CLUT.
  - The CLUT must be reloaded only when LTDC is disabled or during vertical blanking period
- The CLUT can be enabled on the fly for every layer through the **LTDC_LxCR** register

CLUT- Palletized

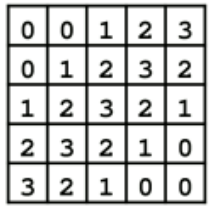
- The R, G and B values and their own respective address are programmed through the **LTDC_LxCLUTWR** register.
  - L8 and AL88 input pixel format, the CLUT has to be loaded by 256 colors
  - AL44 input pixel format, the CLUT has to be loaded by 16 colors. The address of each color must be filled by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

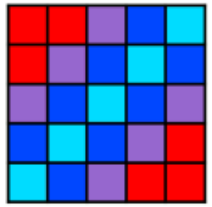
Indexed 16 – L4

Indexed 256 - L8

RGB888

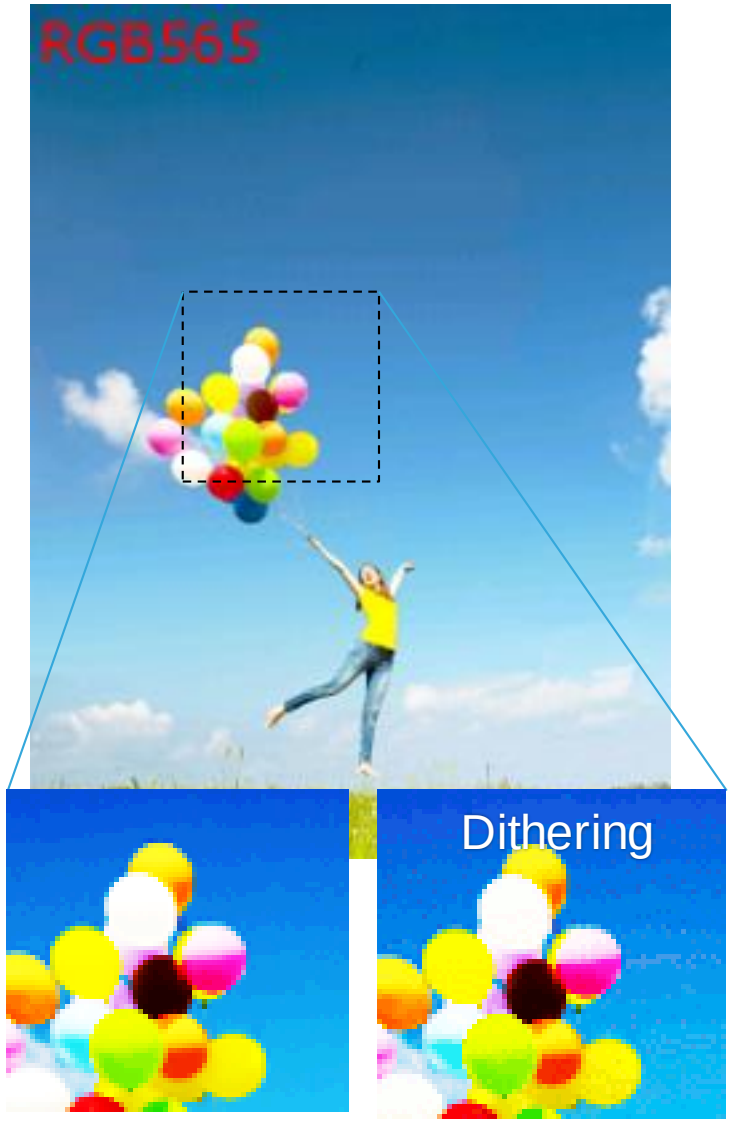
Source - wikipedia.org





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



Demo - Moving layer

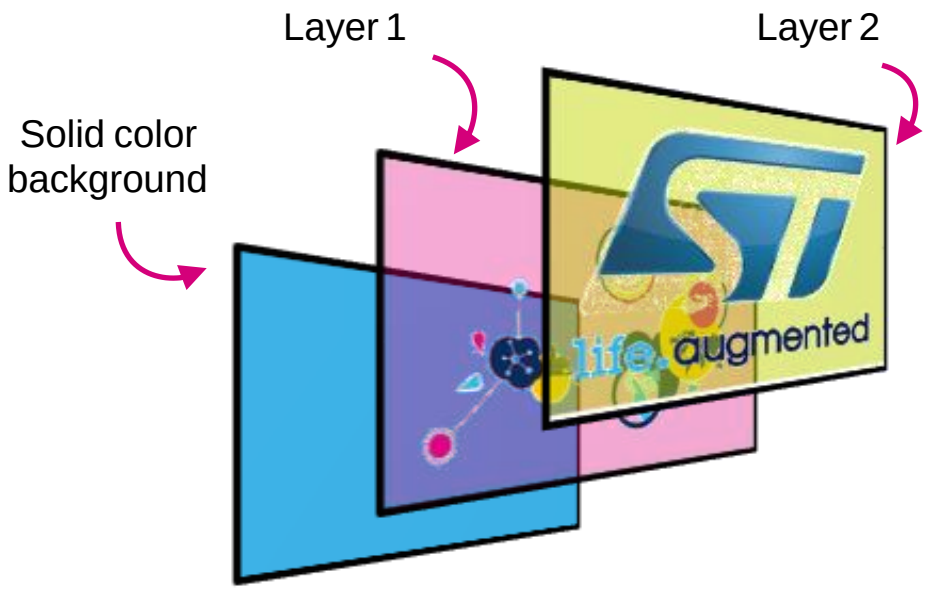

```
LTDC_Layer_InitStruct.LTDC_CFBStartAdress =
(uint32_t)Image + 3 * (offsetH + (offsetV * Image_width));
```

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

- 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 %

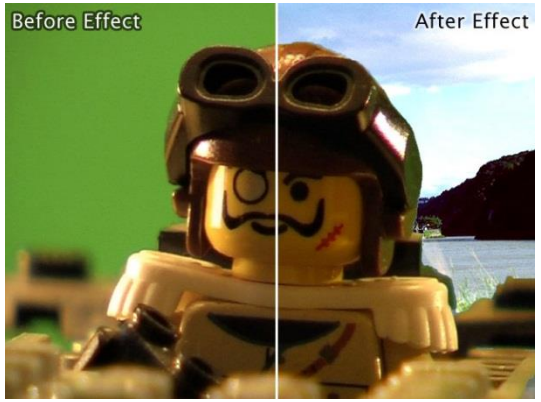
50 %

25 %

0 %

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



MIPI-DSI接口

mipi alliance standards

- **MIPI : Mobile Industry Processor Interface**
  - Organization specifying interconnections between ICs within phones
  - Founded in 2003 by ARM, Intel, Nokia, Samsung, **STMicroelectronics** & TI

- **MIPI is organized in working groups covering all the topic related to ICs interconnect**
  - Analog control interface
  - Battery interface
  - Camera
  - Debug
  - **Display**
  - **PHY**
  - …

Display Working Group

Standardize existing interface and develop next generation

- **The display working group is specifying the interconnection between an IC and a display**
- **It standardized already existing protocol addressing displays**
  - SPI, parallel Intel 8080 and Motorola 6800 ➔ DBI
  - Display specific commands ➔ DCS
  - Parallel RGB ➔ DPI
- **As the mobile industry is driving the demand for display, the MIPI standards are required for the next gen of graphic MCUs**
  - DSI

MIPI & STM32

STM32 is supporting the mipi alliance protocols

- **STM32 with LTDC are also supporting DPI**
- **All STM32 are supporting DBI Type C (SPI)**
- **STM32 with F(S)MC are also supporting DBI Type A&B**
- **All STM32 are supporting DCS (pure software)**
- **New generation of STM32 is supporting DSI with embedded D-PHY**
  - **STM32F469, STM32F769, STM32L4R9 & STM32L4S9**

DSI Introduction

- Target to decrease the number of wires between MCU and displays
  - Builds on DPI-2, DBI-2 & DCS
- Able to have bidirectional communication
  - Send pixels or commands to the peripheral
  - Read back status or pixel information from the peripheral
  - Traditional pixel data, commands, events, control signals serializes by DSI
- DSI uses MIPI D-PHY for the physical link
  - Provide DSI and CSI (camera serial interface) in the physical definition.
  - High speed(HS) & LP(Low Power) Communication.
    - 80Mbps to 1500Mbps/lane in HS (High Speed) mode, fastest PHY to 2.0GHz.
    - STM32 support from 31.25MHz to 500MHz
    - Low Power speed < 20MHz same as STM32

DSI Introduction

- Serialization and deserialization operations are transparent

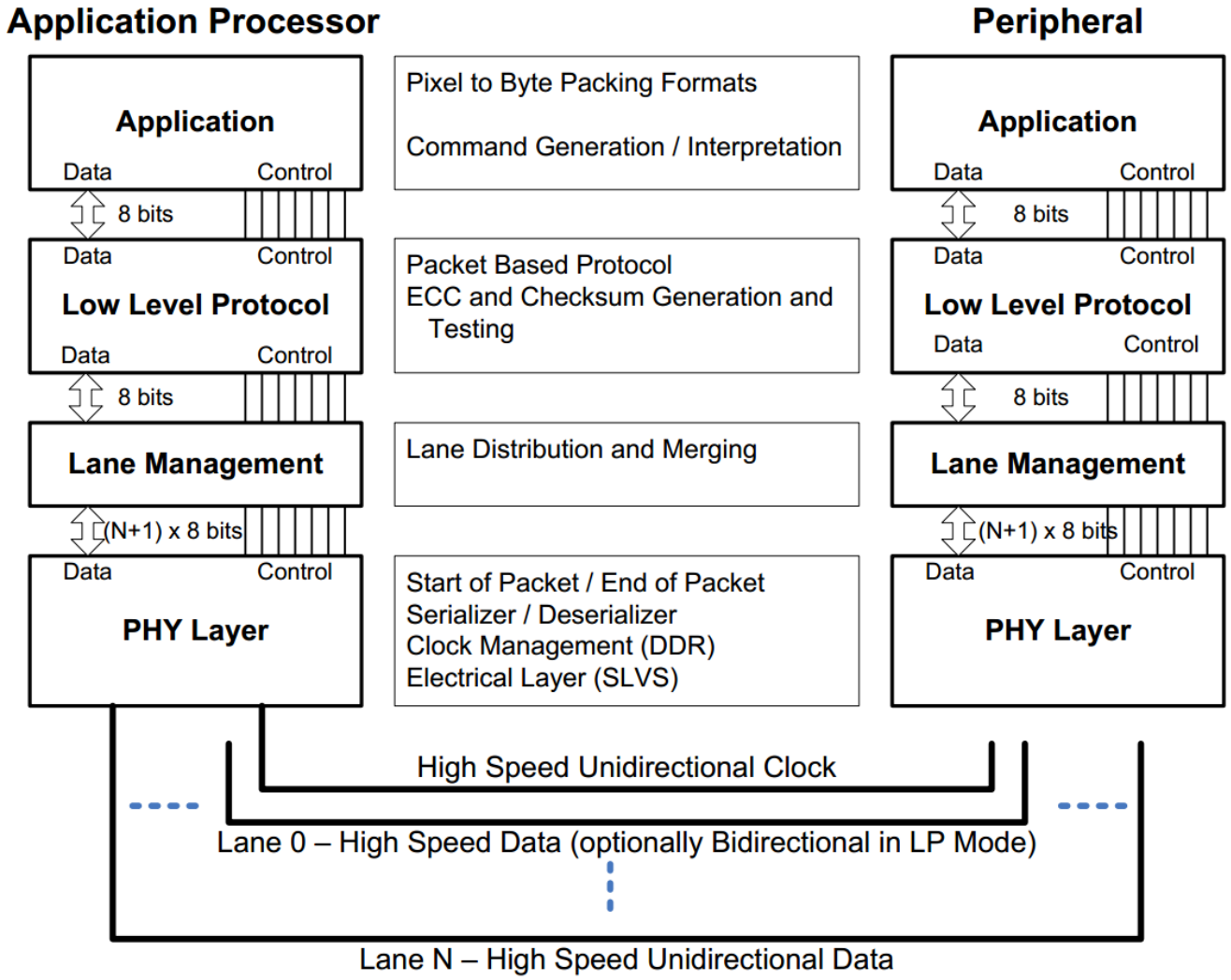
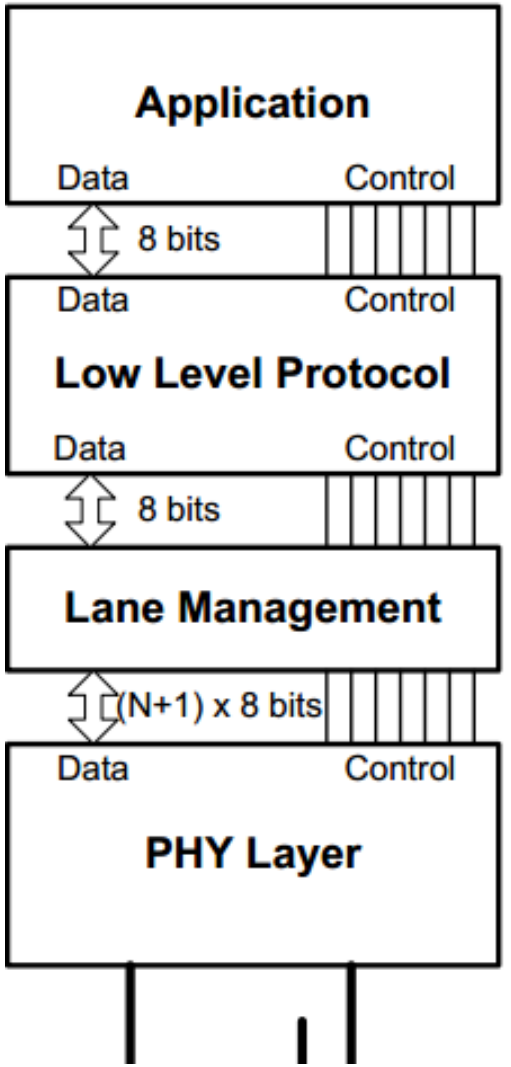


Figure 2 DSI Layers

DSI Layer

- *PHY Layer* specifies transmission medium
  - SoT, EoT
  - Capability to have bidirectional communication
- Lane Management exchange data between lanes
  - Maximum 4 lanes defined by DSI
- Protocol Layer specifies sequence and value
  - Packing Application data into package in bytes
  - Append & stripe package header
    - PH(Package Header) & PF(Package Footer) describes package attributes like package start, end, length, ECC, etc
- Application Layer
  - Describes higher-level encoding & interpretation of data contained in the data stream for command or pixel format
  - DSI defined Mapping of pixel values & command & Parameters



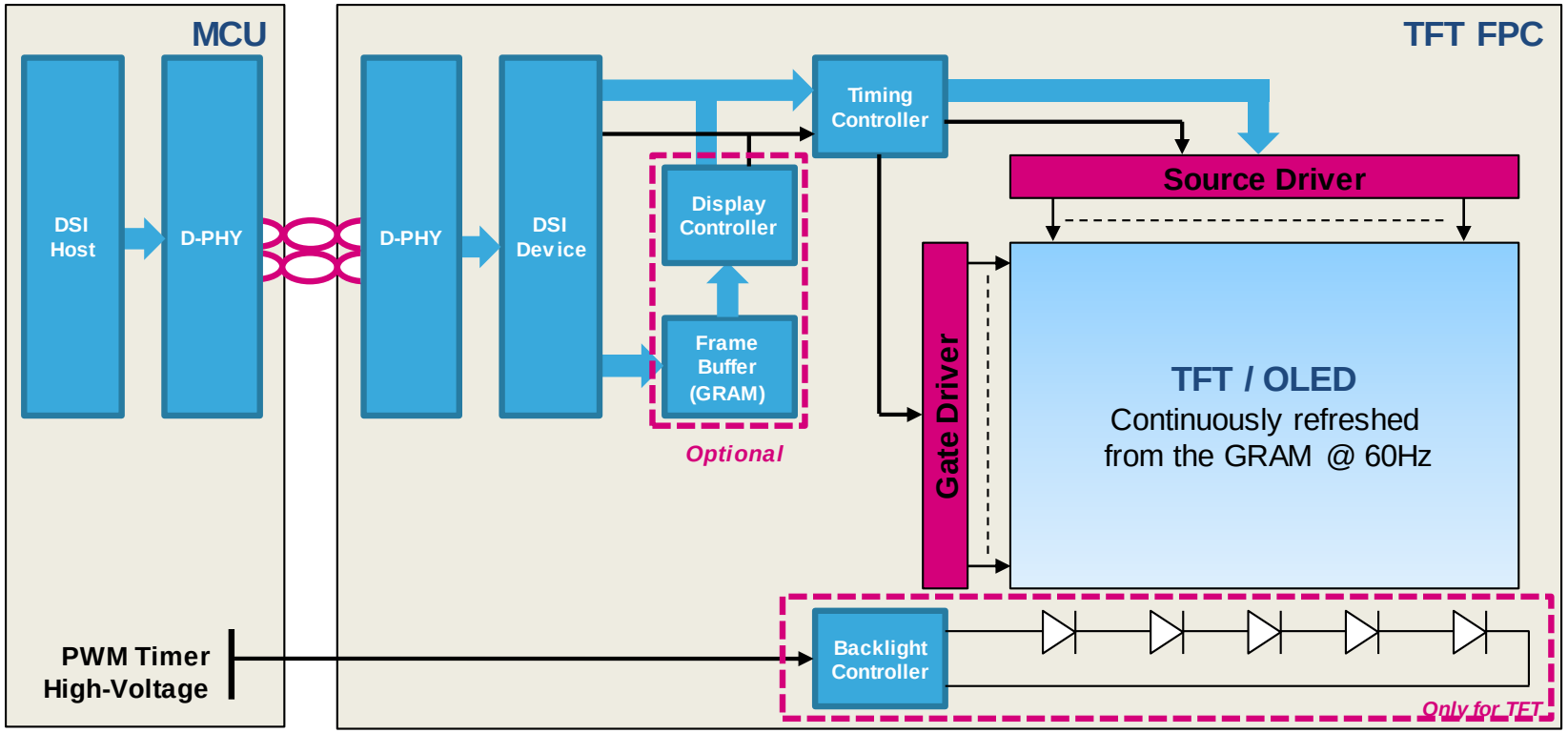
DSI Protocol

High Speed and Low Power communication

- **Two communication speed**
  - **High-Speed** (HS)
    - Need the Clock lane and 1 to 4 data lanes (2 max. on STM32) in **differential mode**
    - Maximum speed for fast communication (video mode or GRAM filling in command mode)
  - **Low-Power** (LP)
    - 20MHz maximum over data lane 0 only (called **escape mode**) in **single ended mode**
    - Mainly used for display initialization and basic communication not requiring high-speed (data or triggers)
    - Ultra-Low-Power-Mode
- **Data lane can be**
  - Unidirectional
  - Bi-directional (i.e. Support turnaround & reverse communication)
    - HS reverse data communication
    - LP reverse escape mode

Display with DSI Interface

Supports all possible architecture with on single interface



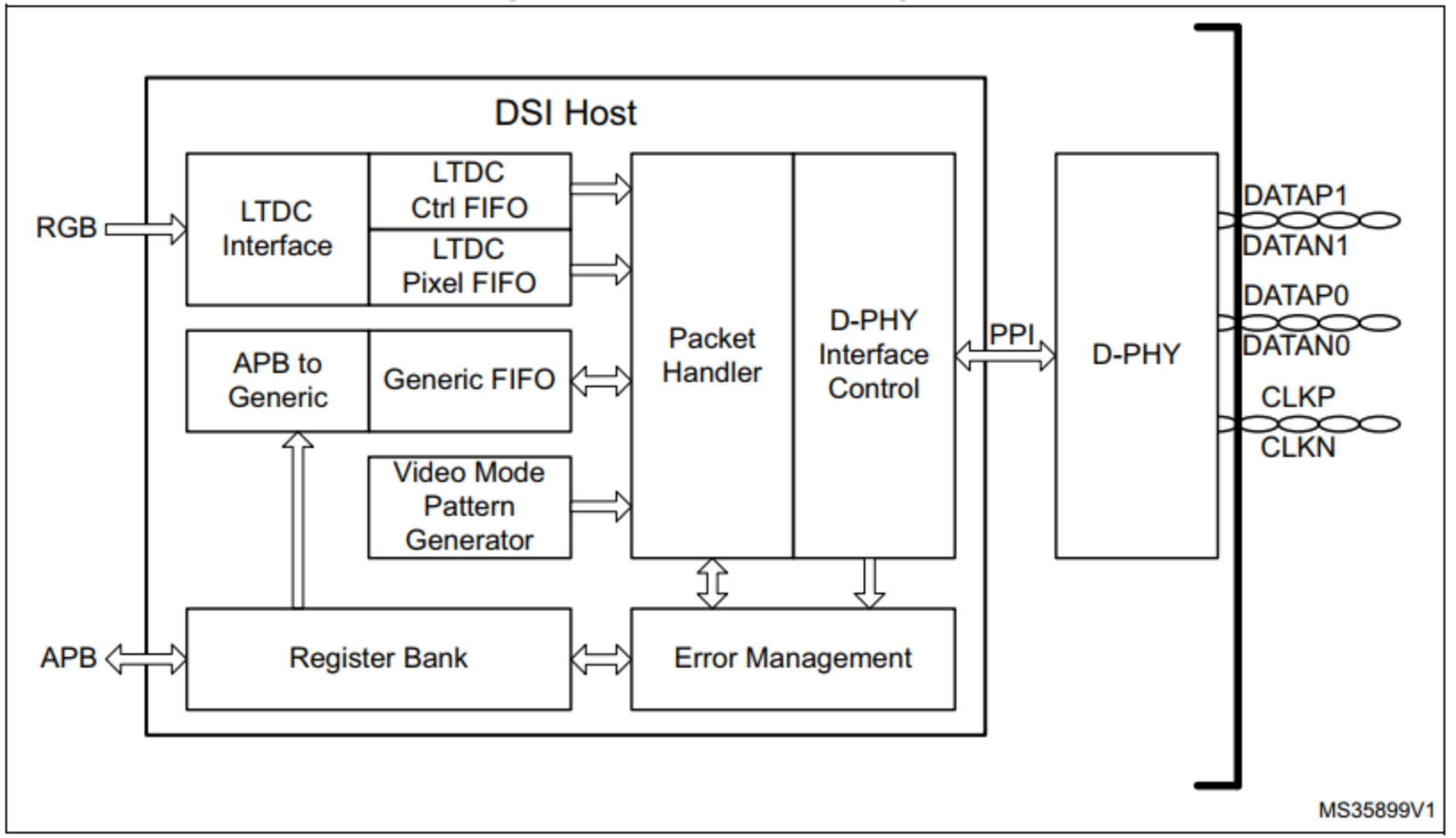
DSI in STM32

life.augmented

STM32 DSI Key features

Flexible operating modes to efficiently support all display types

- Three operating modes
  - Video mode
  - APB Command mode
  - Adapted Command mode
- Very fast transfers up to 1Gbit/s (500 Mbits/s per lane) on STM32F MCUs
- User selects the number of data lanes (up to 2)
  - Number of pins fits exactly with the bandwidth requirement
- Deeply integrated with LTDC
  - LTDC remains the “streamer” and feeds the DSI Host


DSI Host
RGB
LTDC Interface
LTDC Ctrl FIFO
LTDC Pixel FIFO
APB to Generic
Generic FIFO
Video Mode Pattern Generator
Packet Handler
D-PHY Interface Control
PPI
D-PHY
DATAP1
DATAN1
DATAP0
DATAN0
CLKP
CLKN
APB
Register Bank
Error Management
MS35899V1

Operating modes

Flexible operating modes to support efficiently all display types

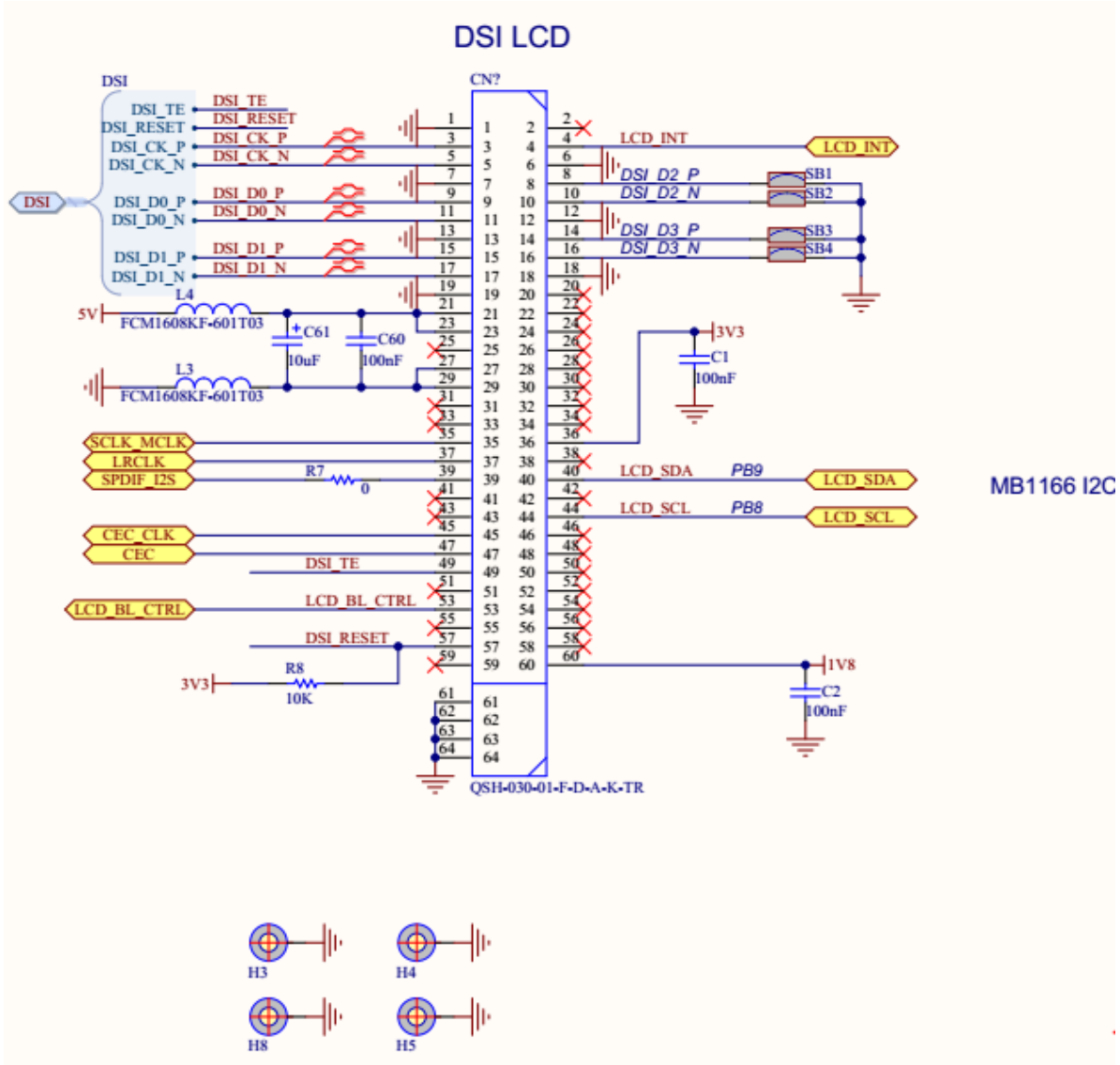
- Video mode
  - DSI Host sends LTDC output, including HSYNC and VSYNC signals over DSI (streaming)
- APB Command mode
  - DSI Host sends DCS or custom commands over DSI (similar to serial LCDs with SPI or FMC interface)
  - Commands are launched using DSI Host APB interface
- Adapted Command mode
  - DSI Host captures one full LTDC frame and transforms it automatically to a series of DCS commands to update the display's Graphics RAM (as previously done through SPI or FMC)
  - Most efficient way to manage Graphics RAM updates

Performance

Equivalent Pixel Clock

- Relationship between DSI bandwidth and LTDC Pixel clock
  - Depends on the color coding of the targeted display (bpp)
  - Equivalent LTDC clock: 1 Gbit/s / 16 bpp = 62.5 MHz in 16 bpp
  - Equivalent LTDC clock: 1 Gbit/s / 24 bpp = 41.5 MHz in 24 bpp
- Application example
  - 200 MHz DSI with 1 data lane ~200 Mbit/s bandwidth
    - For a 16 bpp display, PCLK = 200/16 ~ 10 MHz (~400x400 / 60 Hz)
  - 500 MHz DSI with 2 data lanes ~ 1 Gbit/s bandwidth
    - For a 24 bpp display, PCLK = 1000/24 ~ 40 MHz (~800x600 / 60 Hz)

STM32F769 MIPI-DSI schematic

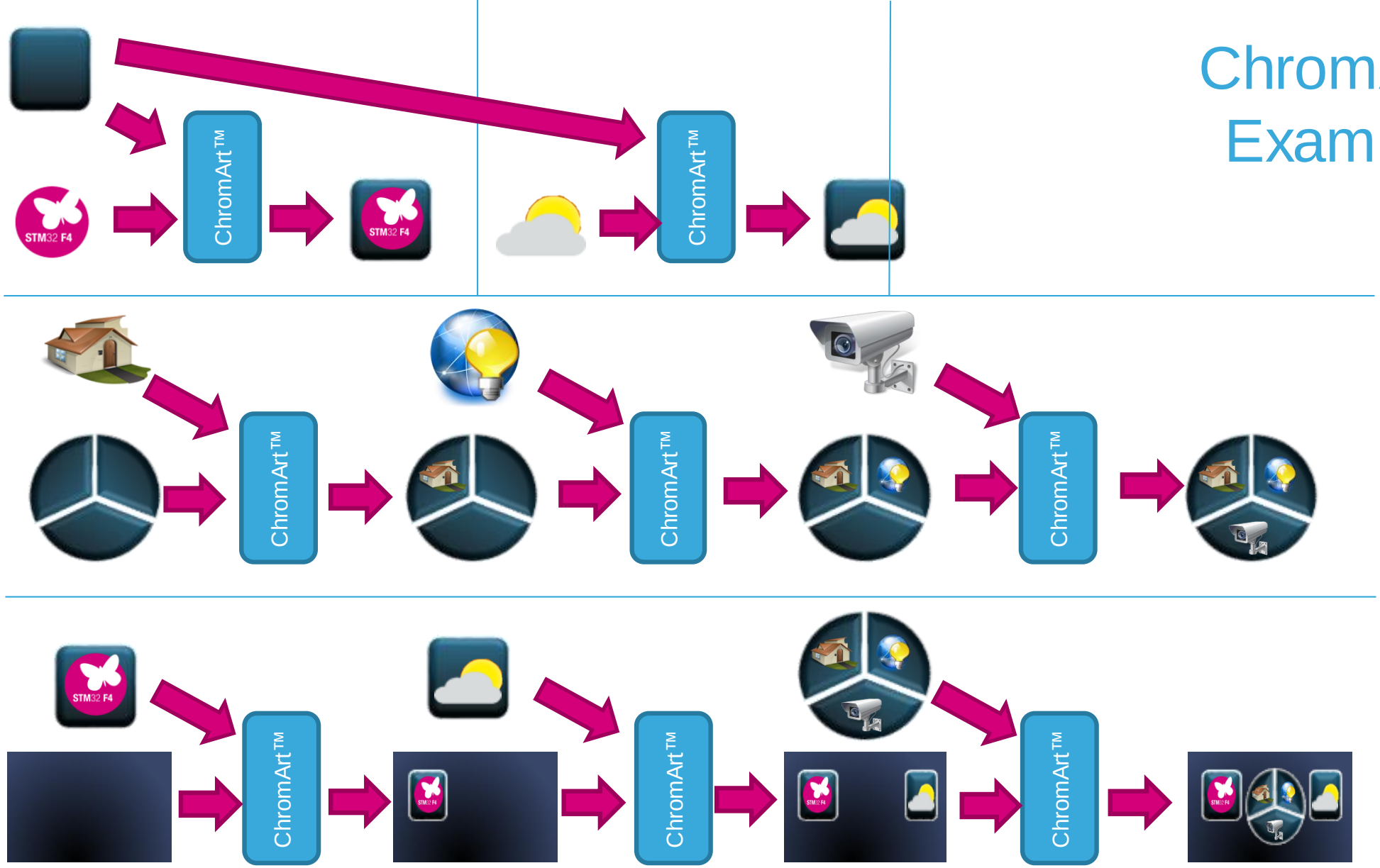


STM32 GUI 平台介绍

- STM32图像显示过程
- LCD接口(DBI/DPI/DSI)
- 图形加速(ChromART & ChromGRC)

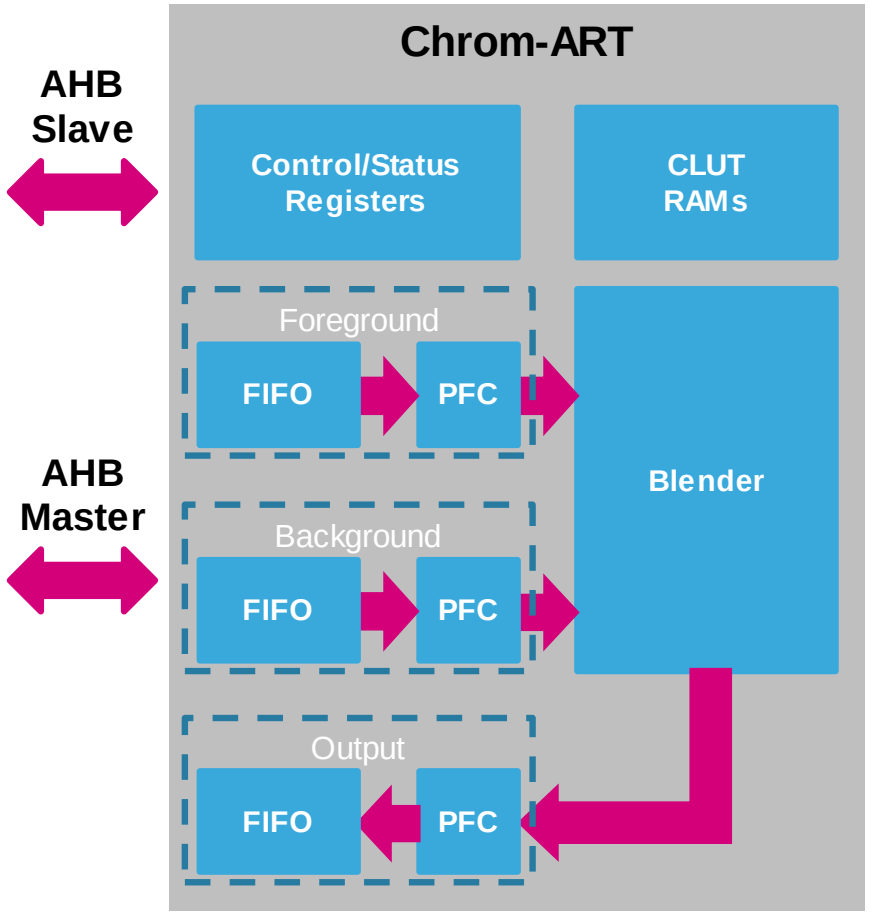
Chrom-ART Accelerator (DMA2D)

ChromArt Example



ChromART™ efficiently offload CPU from graphic tasks

Overview



For category 1 devices only

- Provides hardware acceleration for graphical operations
  - Graphics-oriented 2D DMA
  - Planes blending & pixel format conversion
  - Specific modes for anti-aliased fonts

Application benefits

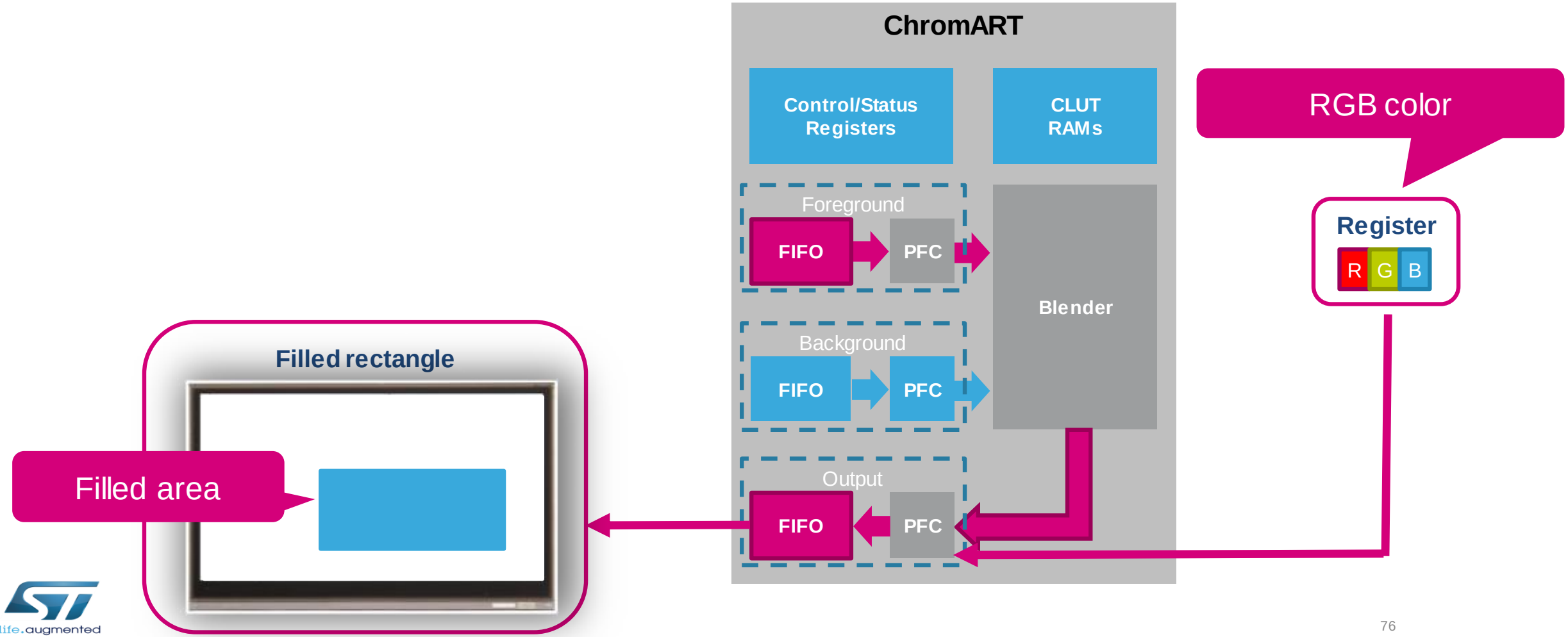
- Offloads CPU for graphical operations
- One pixel per cycle calculation
- Integrated pixel format converter & blender
- Simple integration through graphical stack

Key features

- 2D DMA with graphical-oriented features with 4 operating modes
  - Register-to-memory
  - Memory-to-memory
  - Memory-to-memory with pixel format conversion
  - Memory-to-memory with pixel format conversion and blending
- User programmable parameters
  - Sources and destination addresses on the whole memory
  - Sources and destination size and offset
  - Source and destination color format
    - Up to 11 color formats supported from 4 up to 32 bits per pixel with indirect or direct color coding
    - Internal memories for CLUT storage in indirect color mode with automatic loading
    - Alpha value can be modified (source value, fixed value or modulated value)

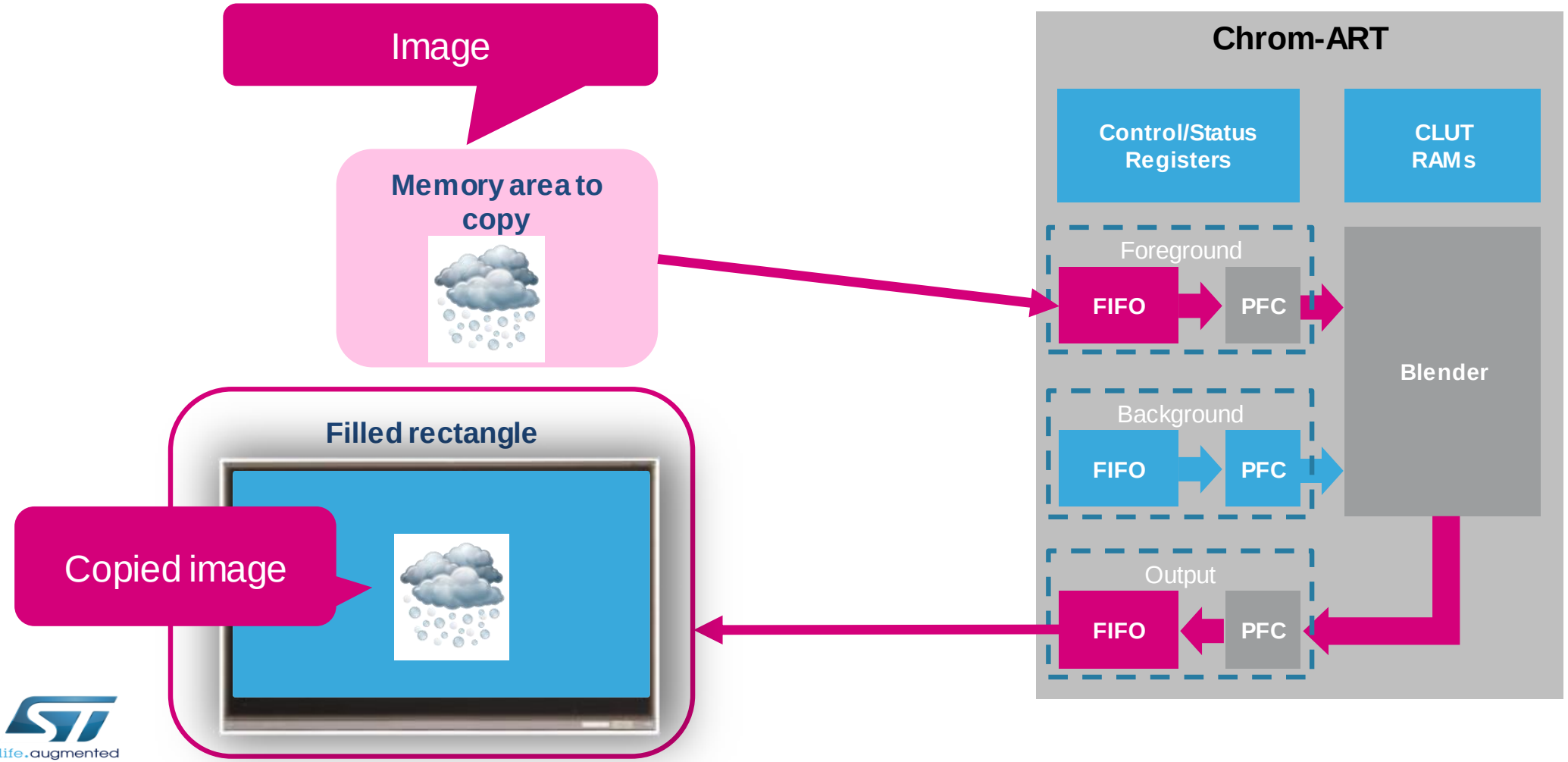
Register to memory

Filling a part or the whole of a destination image with a specific color



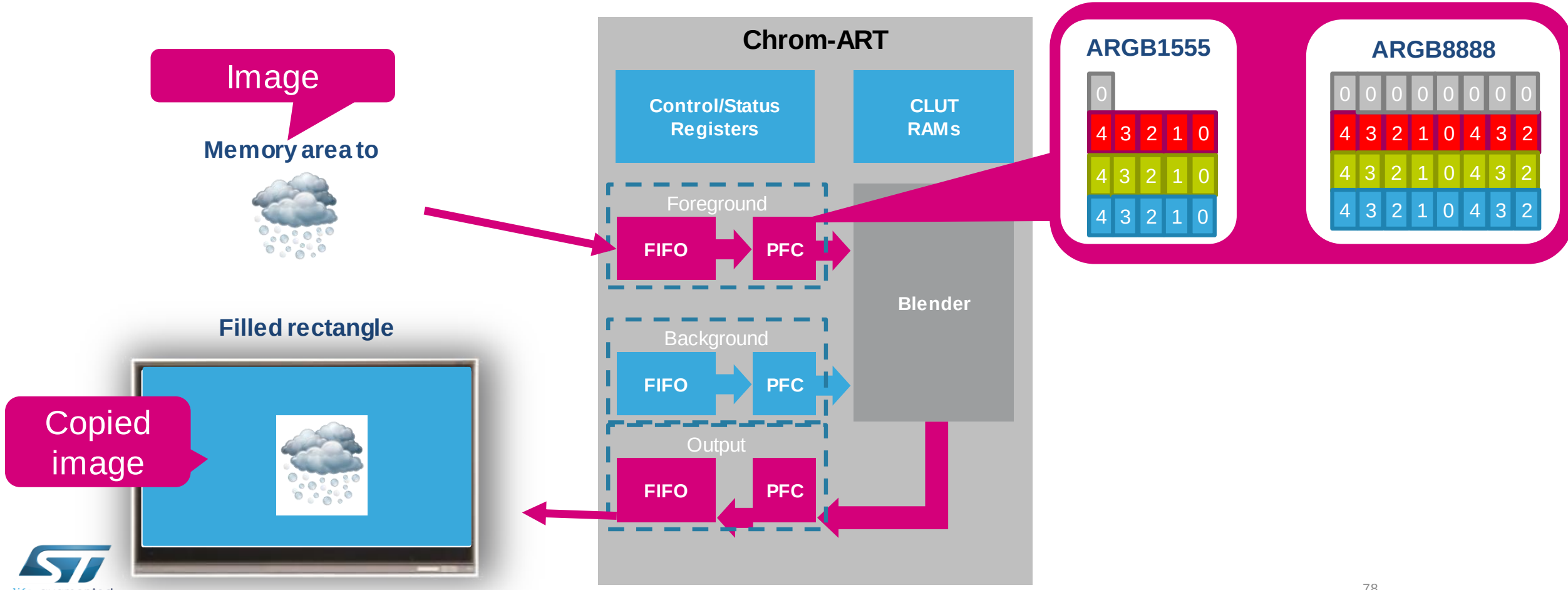
Memory-to-memory

Copying a part or whole source image into a part or whole destination image



Memory-to-memory with pixel format conversion

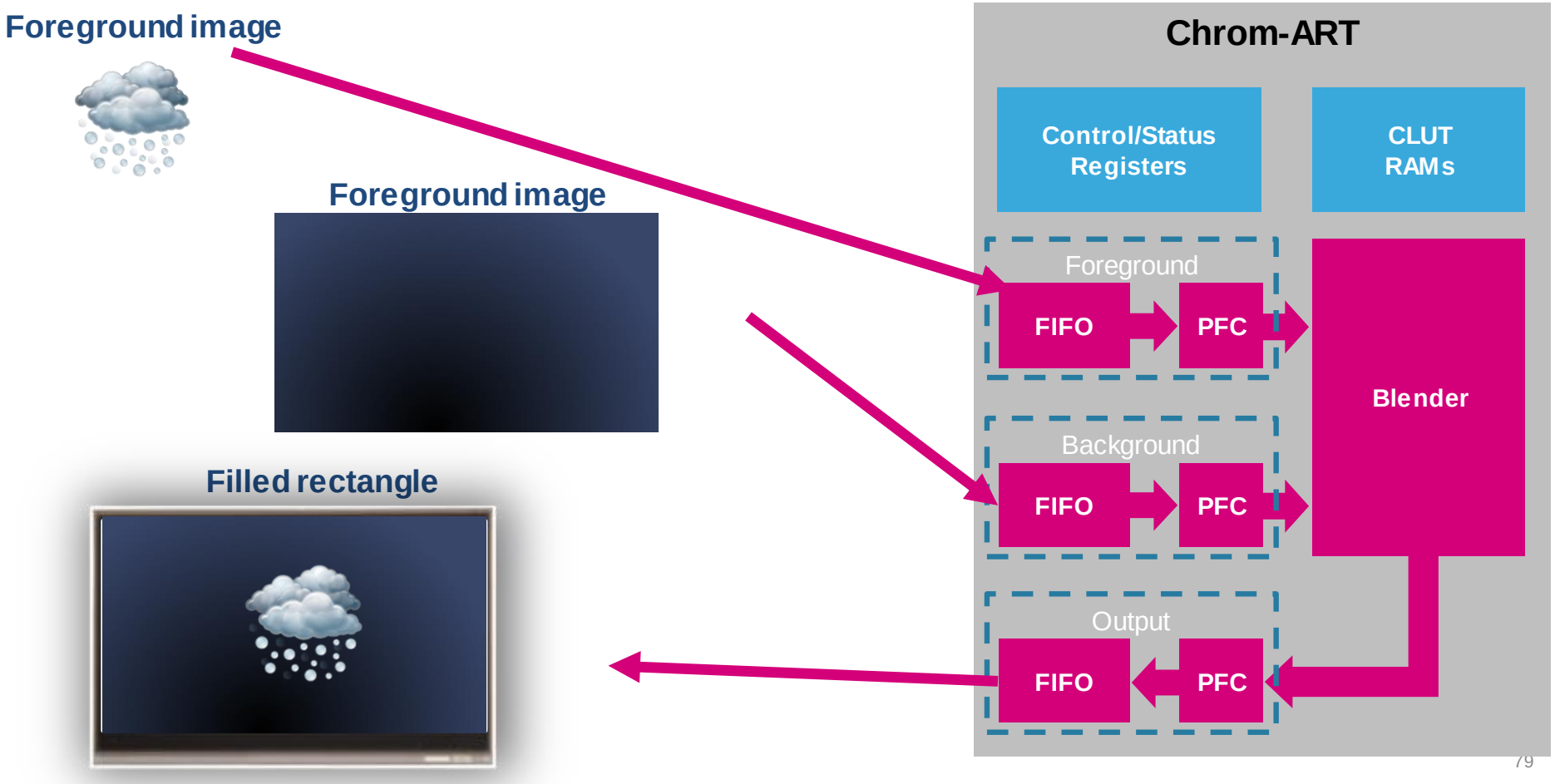
Copying a part or whole source image into a part or whole destination image with a pixel format conversion



life.augmented

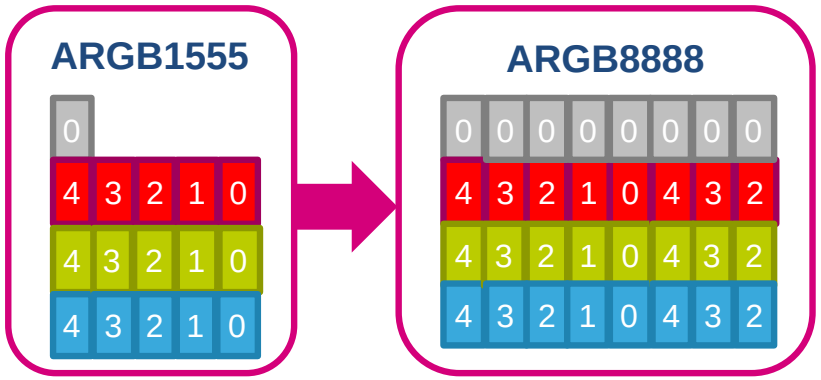
Memory-to-memory with pixel format conversion and blending

Copying a part or whole source image into a part or whole destination image with a pixel format conversion and blending

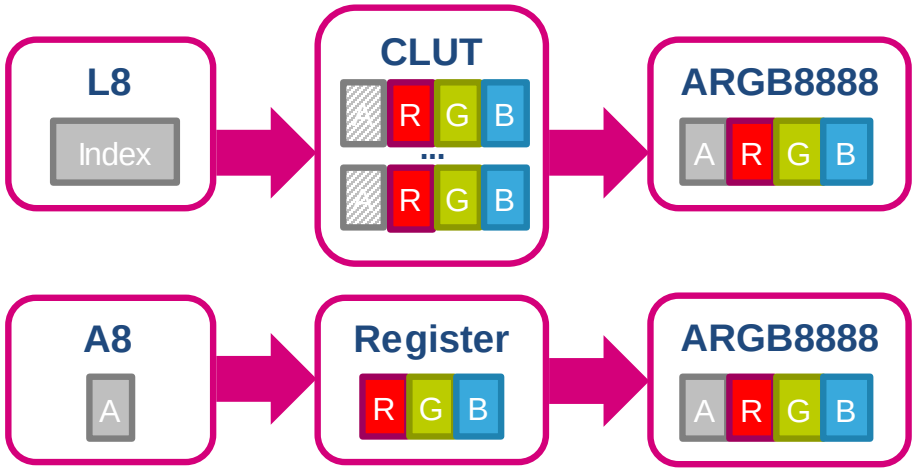


Input pixel format conversion

Independently supports different input color depths



- Direct mode
  - ARGB8888, ARGB444444 or ARGB1555
  - RGB888 or RGB565 (with alpha in register)
  - Alpha can be replaced or modulated



- **Indirect mode**
  - L8 or L4 and a Color Look-up Table (**CLUT**)
  - A8 or A4 and a color register (for fonts)
  - Mixed AL88 or AL44

- **Internal operations are done in ARGB8888**

life.augmented

Font special modes

Efficient support for anti aliased bitmap fonts

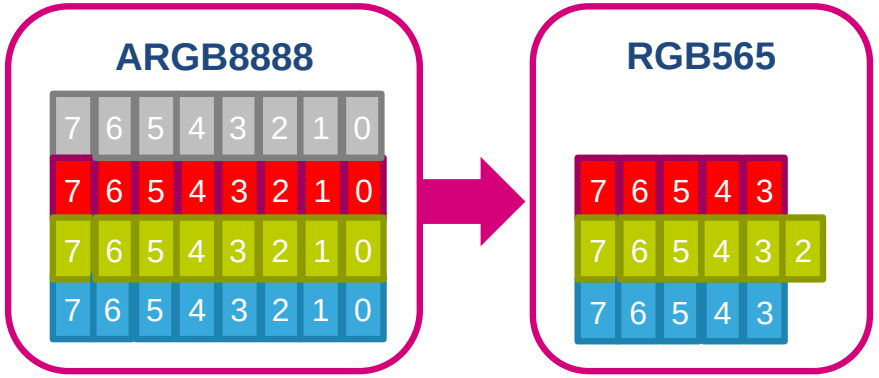
- Using A8 or A4 coding
  - Only the Alpha channel is stored in the memory
  - The Chrom-ART accelerator adds a programmed color

Aliased **Anti-aliased**

Output pixel format conversion

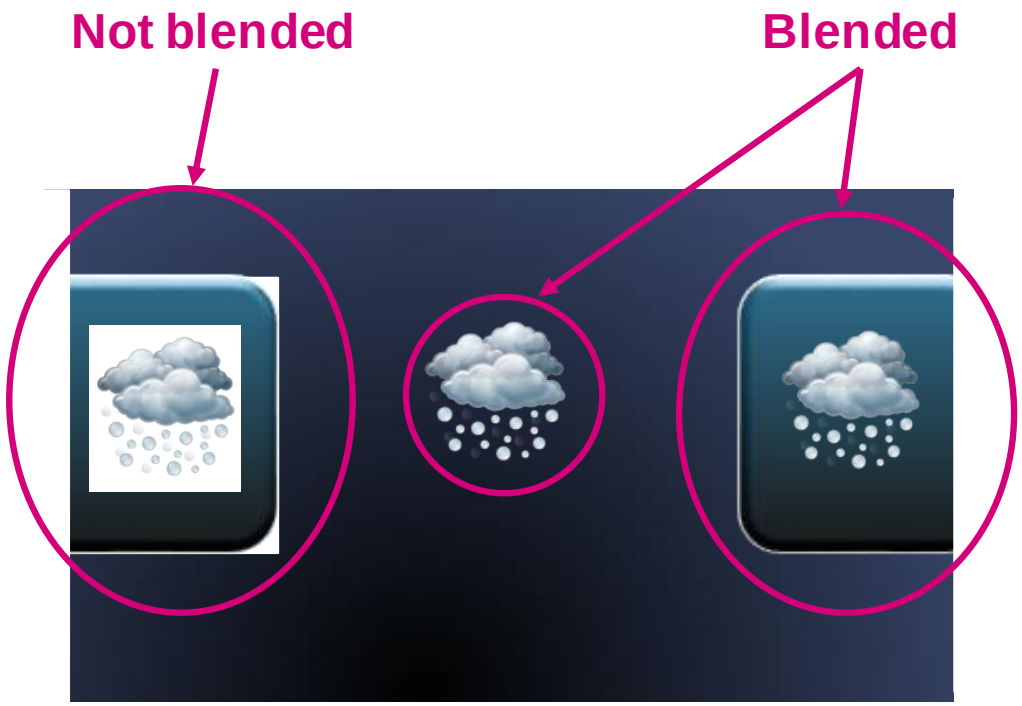
Support several output color depth

- Direct mode
  - ARGB8888, ARGB444444 or ARGB1555
  - RGB888 or RGB565
- Memory-to-memory without Pixel Format Conversion (PFC) is agnostic of the pixel format



Blending

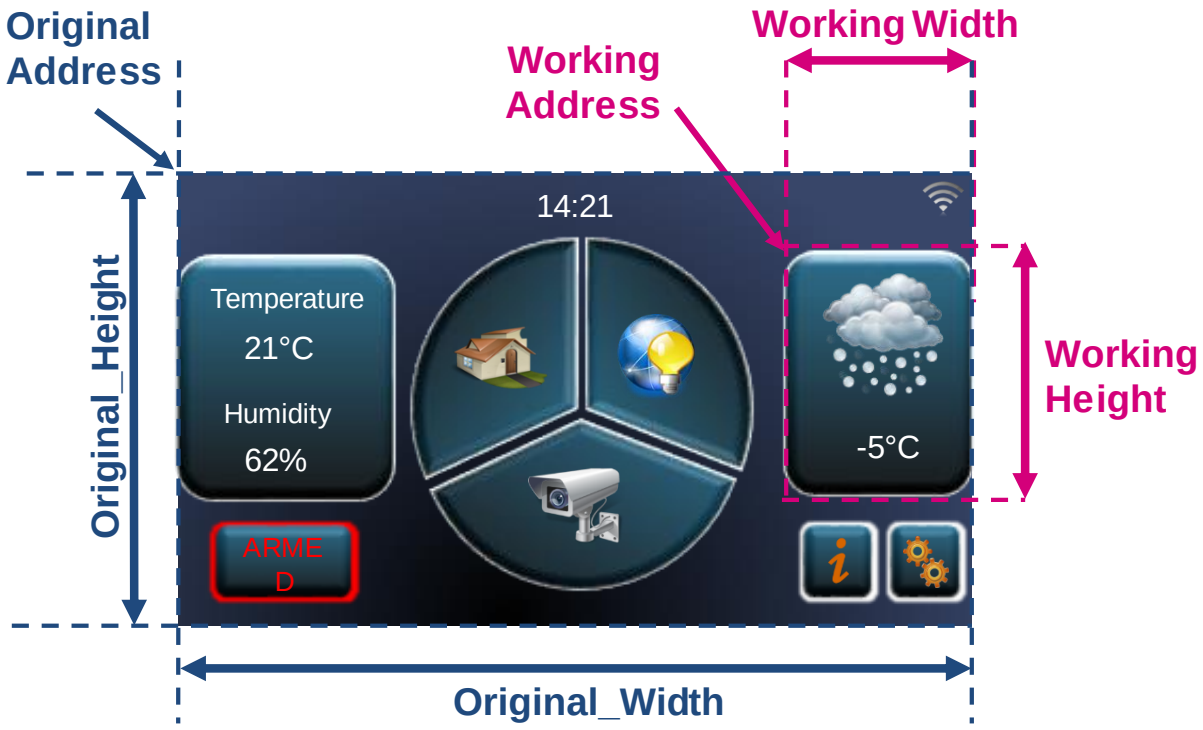
Fully hardware blending process



- Hardware Blending
  - Blend Foreground & Background pixels
  - 1 pixel generated per cycle
  - Native ARGB8888 operation
  - Input data are converted into ARGB8888 by their respective PFC
  - Output data has also its own PFC

Output configuration

Output configuration defines the area of work & output color coding



- Area of work
  - Width
  - Height
- Output configuration
  - Address
  - LineOffset
  - Color Format (BPP)

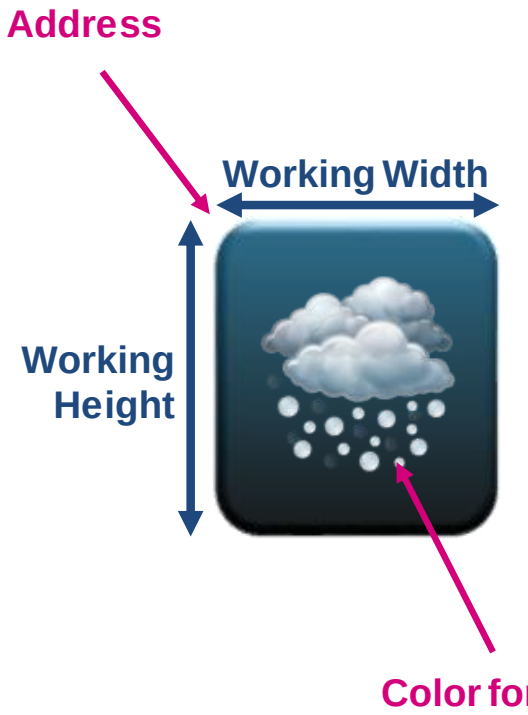
Working_Address = Original_Address + (X + Original_Width * Y) * BPP

LineOffset = Original_Width – Working_Width

FG & BG configuration

FG & BG configuration define address and color coding

- Foreground parameters
  - Address
  - LineOffset (as per output)
  - Color format



- Background parameters
  - Address (as per output)
  - LineOffset (as per output)
  - Color format

Most of the time, background and output have the same parameters

Interrupts

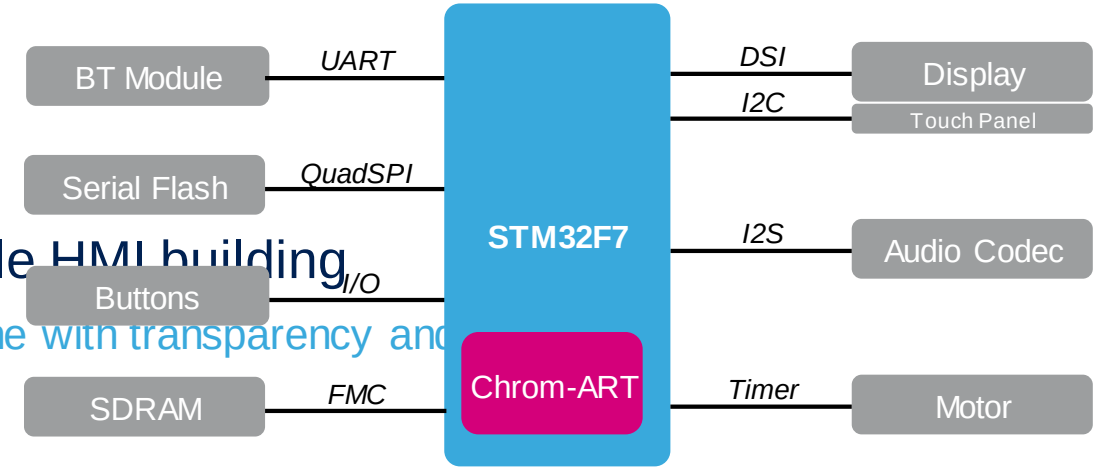
Interrupt event	Description
Configuration error	Error of configuration detected when starting the Chrom-ART (not allowed or wrong configuration)
CLUT transfer complete	CLUT copy from a system memory area to the internal Chrom-ART memory is complete
CLUT access error	CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal Chrom-ART memory
Transfer watermark	Indicates the last pixel of the watermarked line has been transferred
Transfer complete	Chrom-ART transfer operation is complete
Transfer error	An error occurred during Chrom-ART transfer

Low-power modes

Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Frozen. Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.

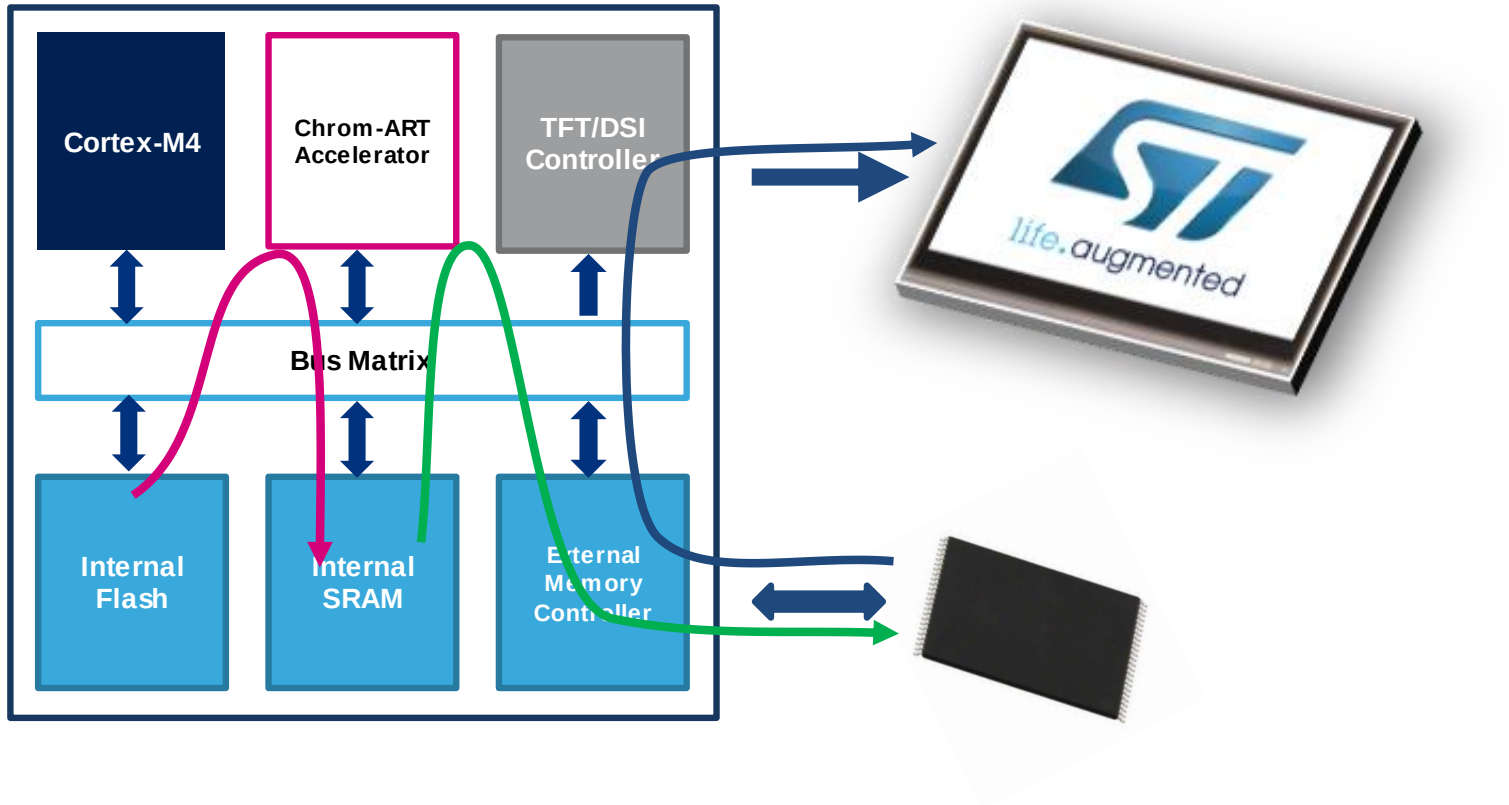
Application examples

- Home appliances including connectivity and HMI:



- Chrom-ART can handle HMI building
  - Composition of the scene with transparency and
  - Text management

Typical data flow

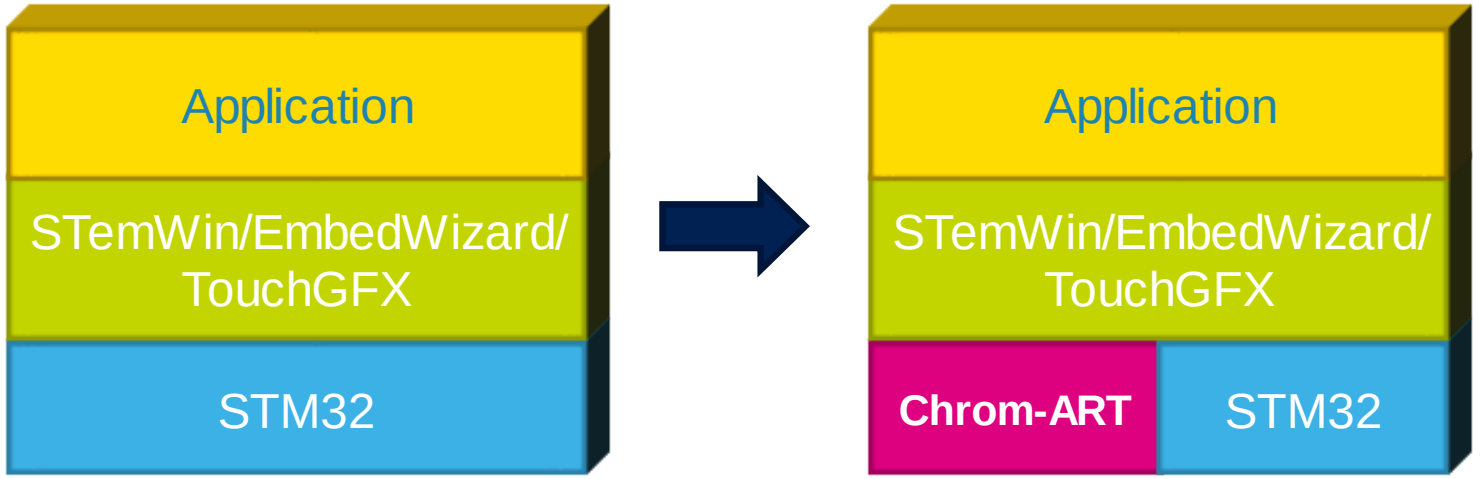


Creation of an object in a memory device by the Chrom-ART

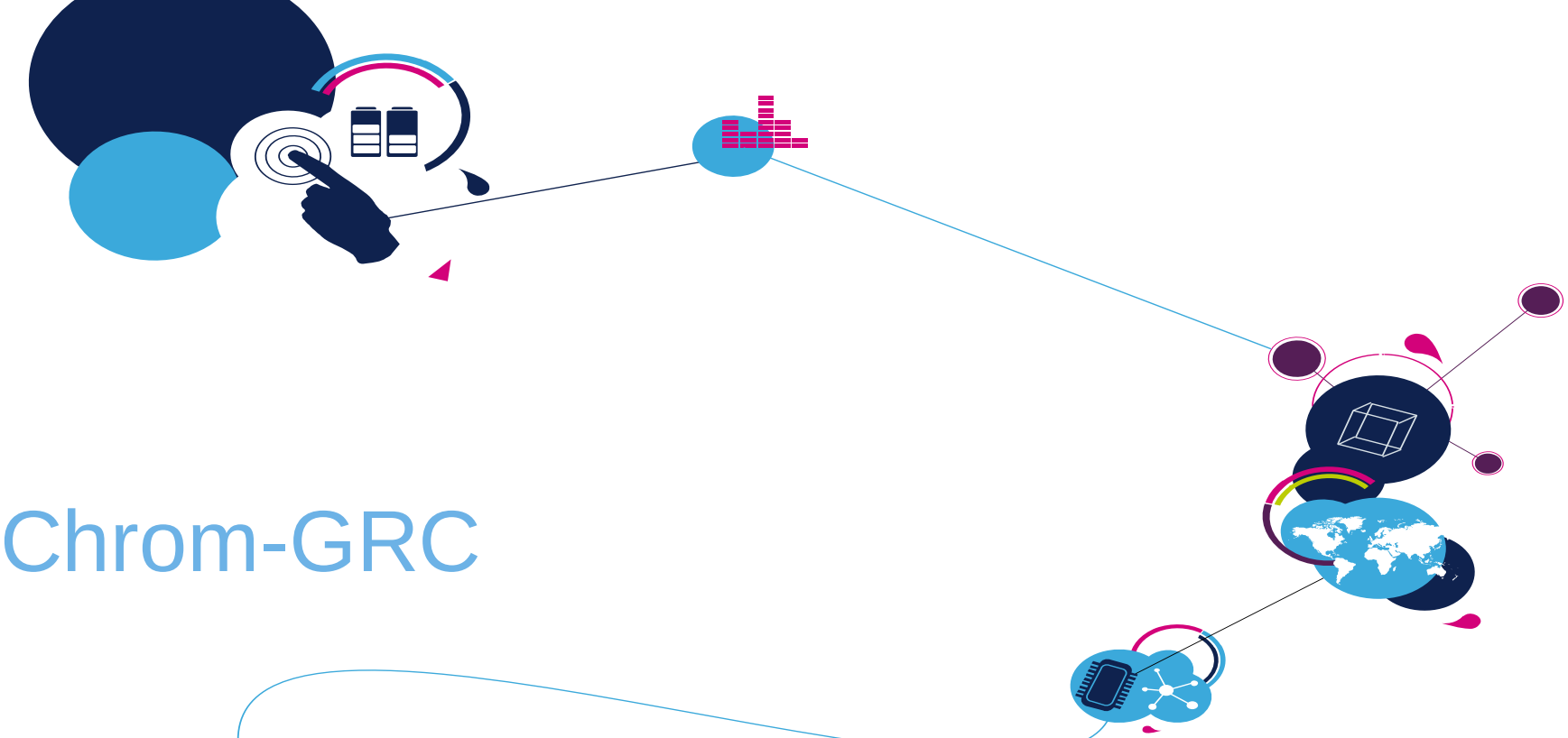
Update/Creation of the frame buffer in the external RAM by the Chrom-ART

TFT controller data flow

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**



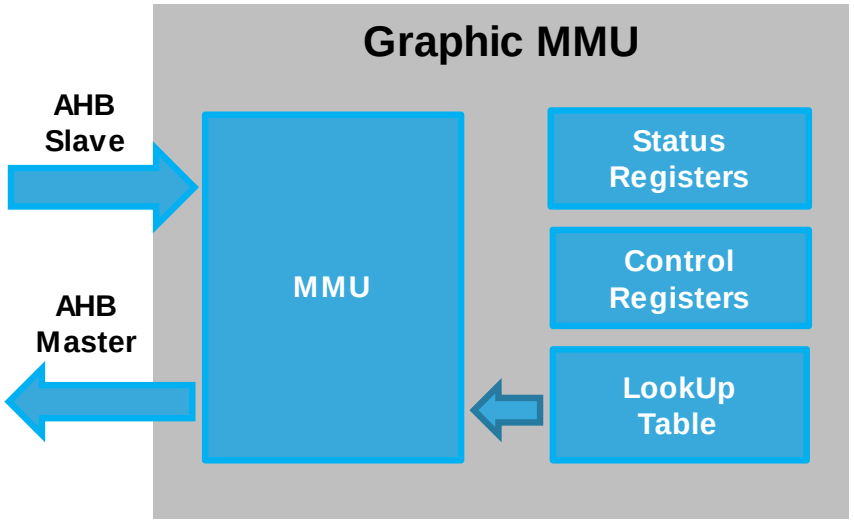
STM32L4+ - Chrom-GRC

Graphic MMU

Revision 1.0

life.augmented

Overview



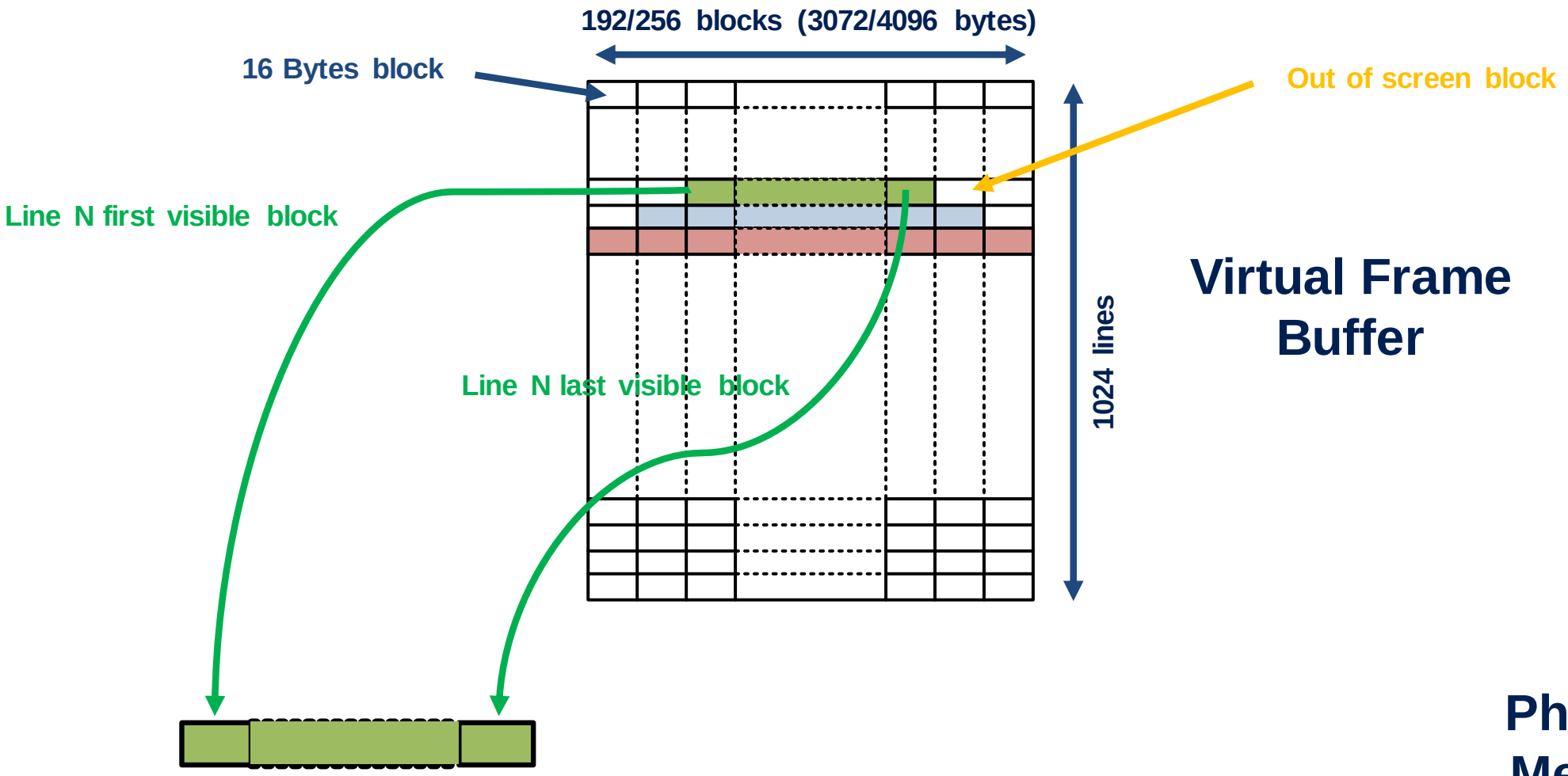
- The **Graphic MMU** is a graphical oriented Memory Management Unit aimed to optimize memory usage according to the display shape (ex. round display).

Application benefits

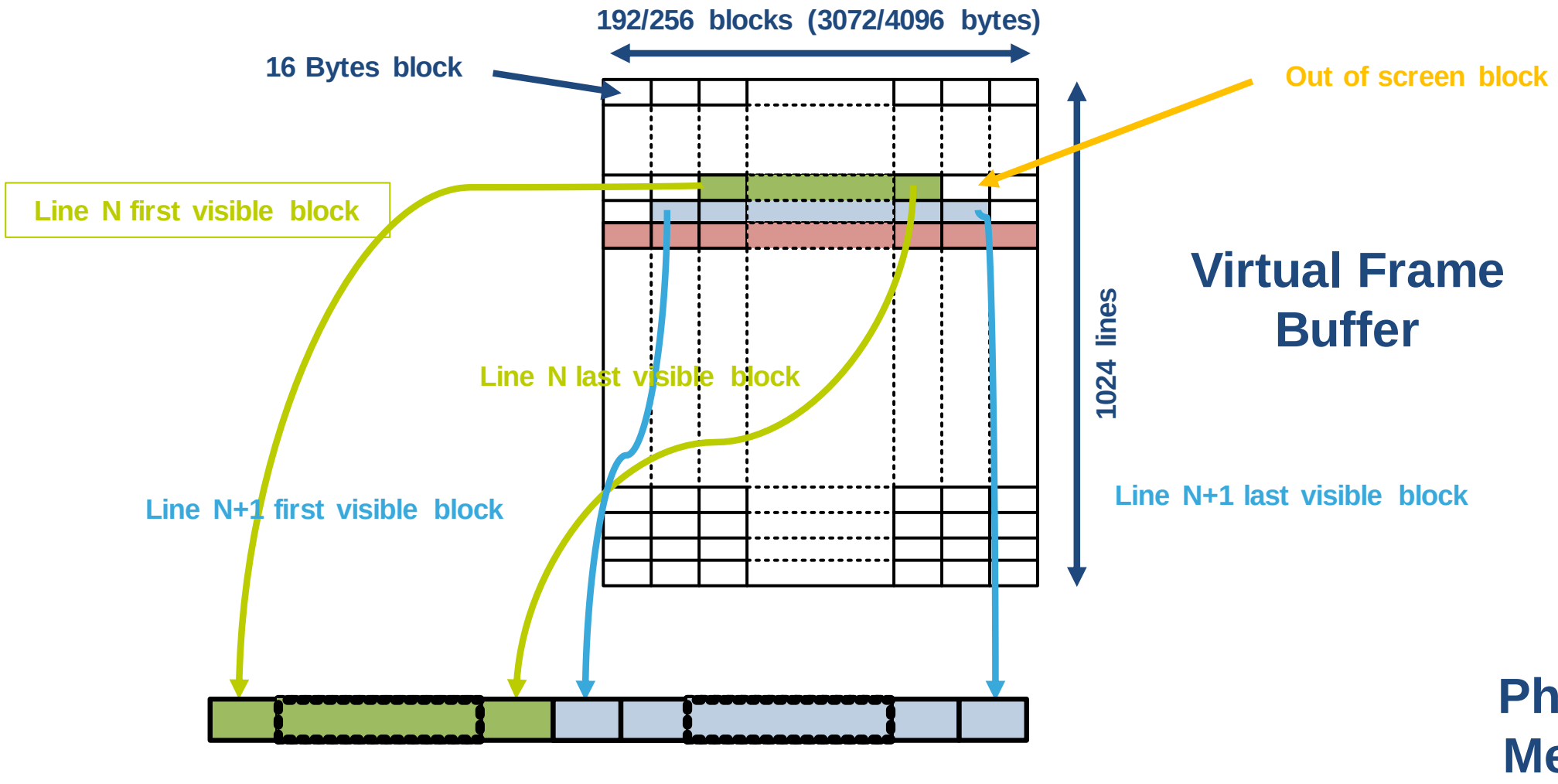
- Lower memory usage according to display shape
- Fully configurable display shape
- Transparent integration
- Works with any memory of the system

life.augmented

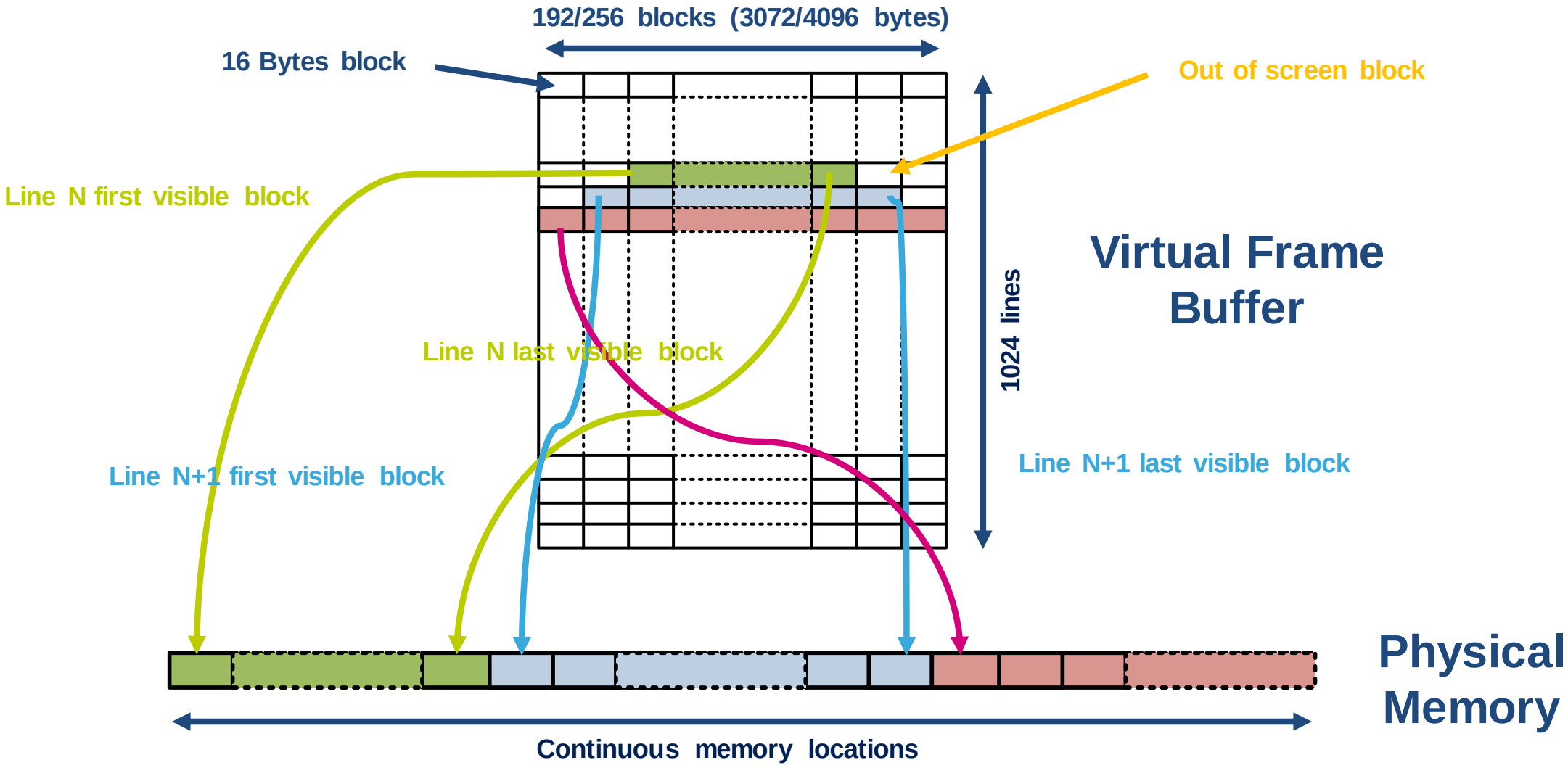
Functional overview



Functional overview

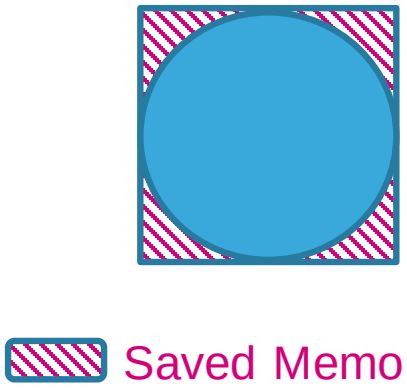


Functional overview



Chrom-GRC (GFXMMU)

- Graphic Ressources Cutter for non square displays
- No modification nor special management at SW level
  - ➔Saving up to 20% of RAM needs



Saved Memory

- **For 360x360 round display**
  - @16bpp **~205kBytes** (vs.253kBytes)
  - @24bpp **~307kBytes** (vs.380kBytes)
- **For 400x400 round display**
  - @16bpp: **250kBytes** (vs.312kBytes)
  - @24bpp: **372kBytes** (vs.469kBytes)

Display shape description

- The shape of a display is described in a Look-Up Table
- This description is common for the 4 virtual frame buffers
- For each line, the LUT entry content is
  - Enable
  - First visible block
  - Last visible block
  - Line offset in the physical memory
- Each virtual buffer has its own base address in the physical memory

Operating modes

- Two operating modes
  - 192 Block Mode: each frame buffer line is composed of 192 blocks or 16 bytes
  - 256 Block Mode: each frame buffer line is composed of 256 blocks or 16 bytes
- Line size in pixel of the framebuffers

Mode	16bpp	24bpp	32bpp
192 BM	1536 pixels/line	1024 pixels/line	768 pixels/line
256 BM	2048 pixels/line	1365,3 pixels/line	1024 pixels/line

- 192 Block Mode allows to have an integer number of pixels in 24bpp mode

1536 = 192 block * 16byte / 2 byte
1024 = 192 block * 16byte / 3 byte

Gain

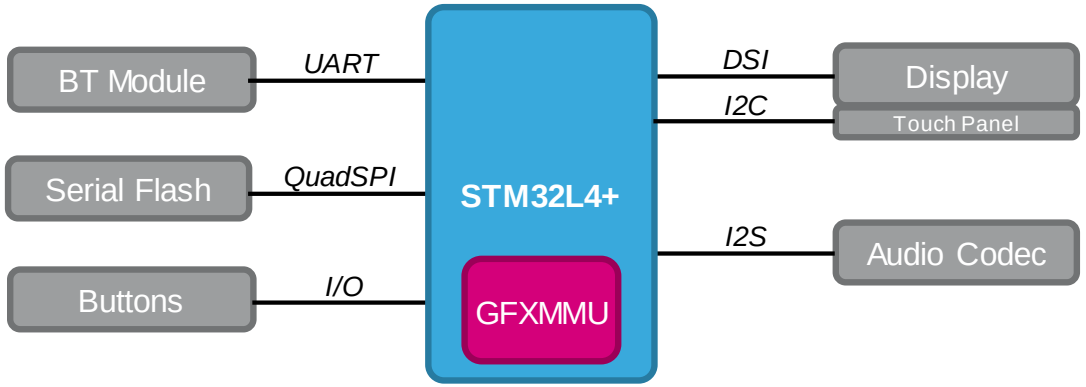
- The gain in size on the graphical frame buffer on round display is ~20% depending on the number of bit per pixels
  - Example for the 390x390 round display of the STM32L4+ kit

Mode	Square size	Chrom-GRC optimized size	Size reduction
16bpp	297,1 KBytes	238,3 KBytes	**-19,8 %**
24bpp	445,6 KBytes	355,5 KBytes	**-20,2 %**
32bpp	594,1 KBytes	471,0 KBytes	**-20,7 %**

Note: SRAM3 384KByte

Application examples

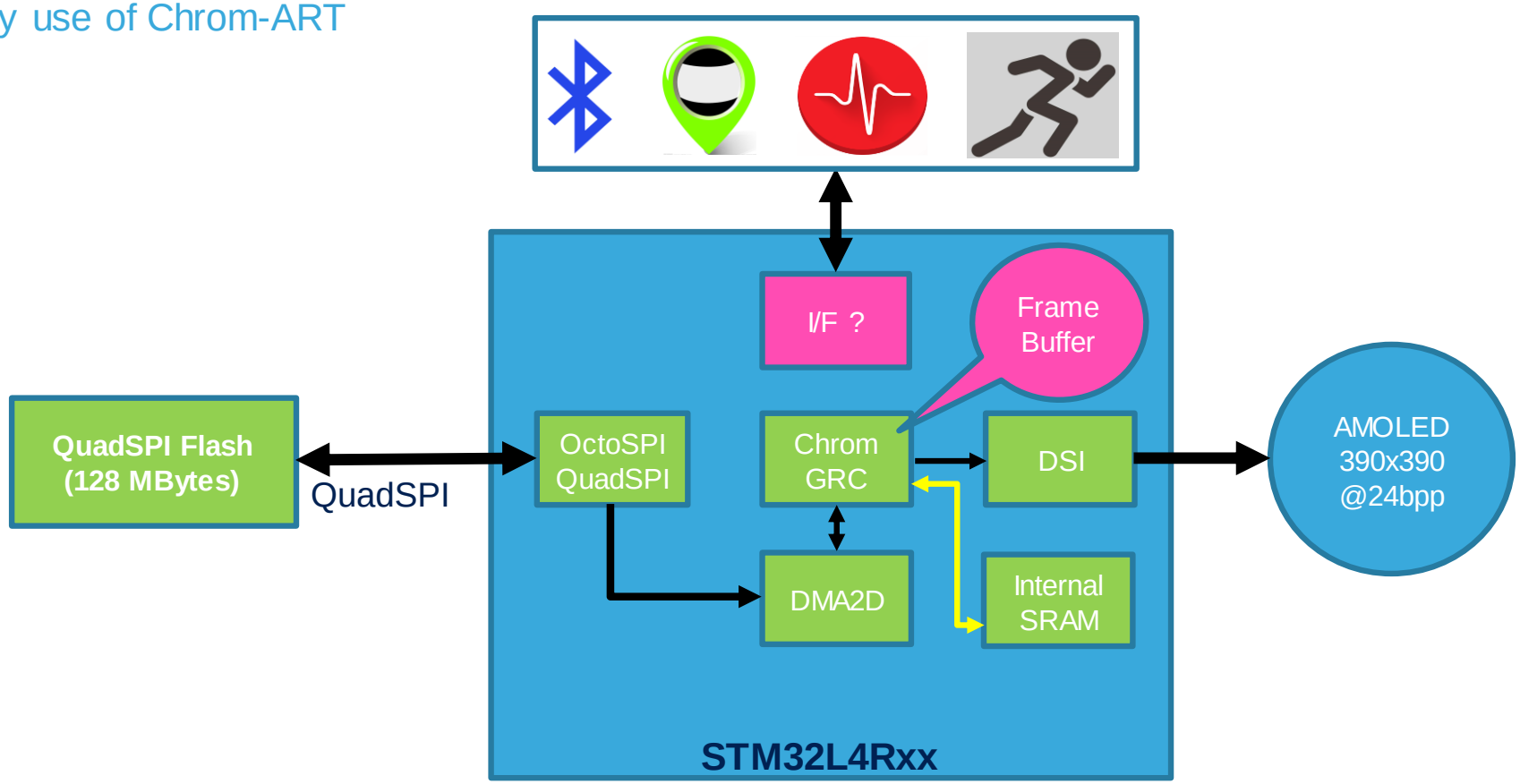
- Embedded applications with connectivity and graphics:



- The Graphic MMU optimizes internal RAM usage for graphic applications
  - No need for external SRAM/SDRAM
  - Faster graphical operations on internal RAM (0-WS)

ST Preferred Topology

- Applications including connectivity and user interface:
  - 390 x 390 / 24 bpp display with embedded GRAM
    - Thanks to Chrom-GRC to use less RAM for round display
  - QSPI to store graphical primitives
  - Heavy use of Chrom-ART



Thanks !

For more information please kindly visit

www.st.com/stm32

www.stmcu.com.cn

www.stmcu.org