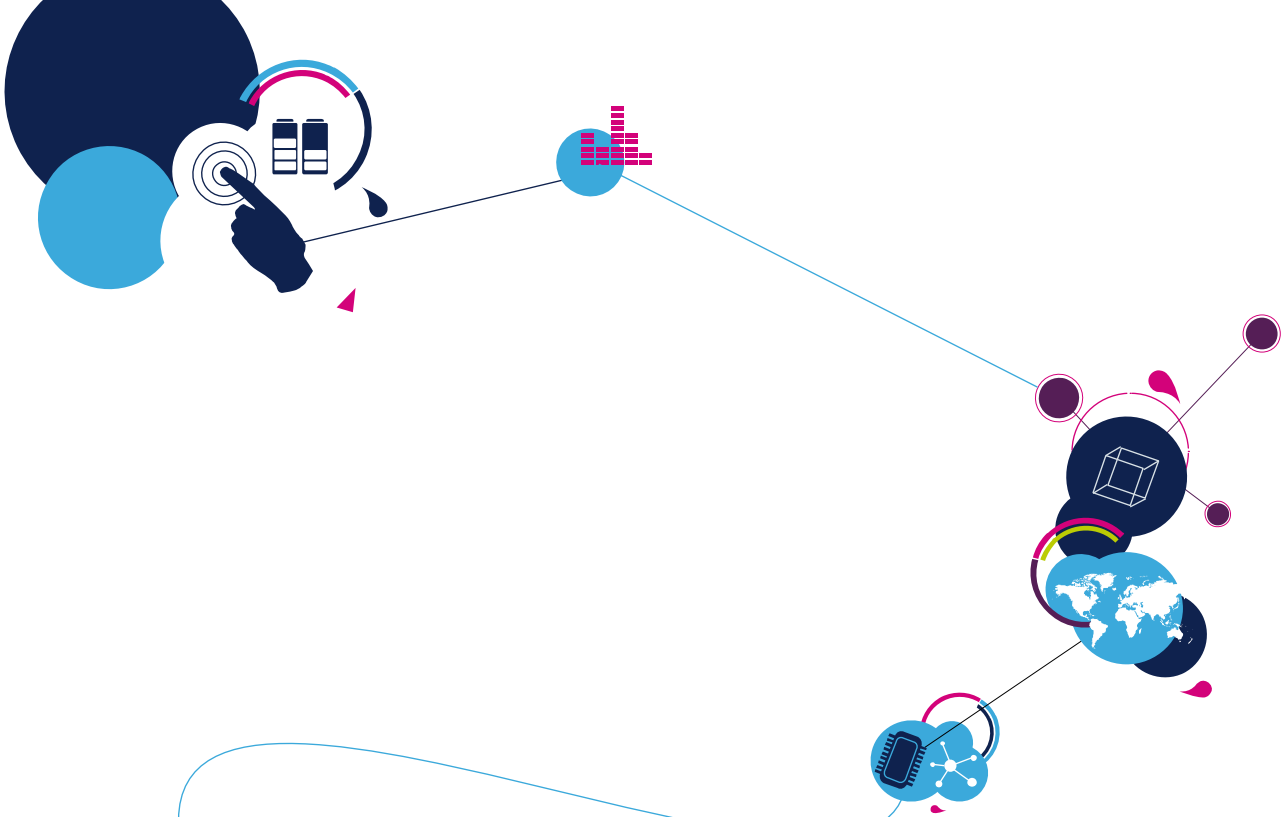


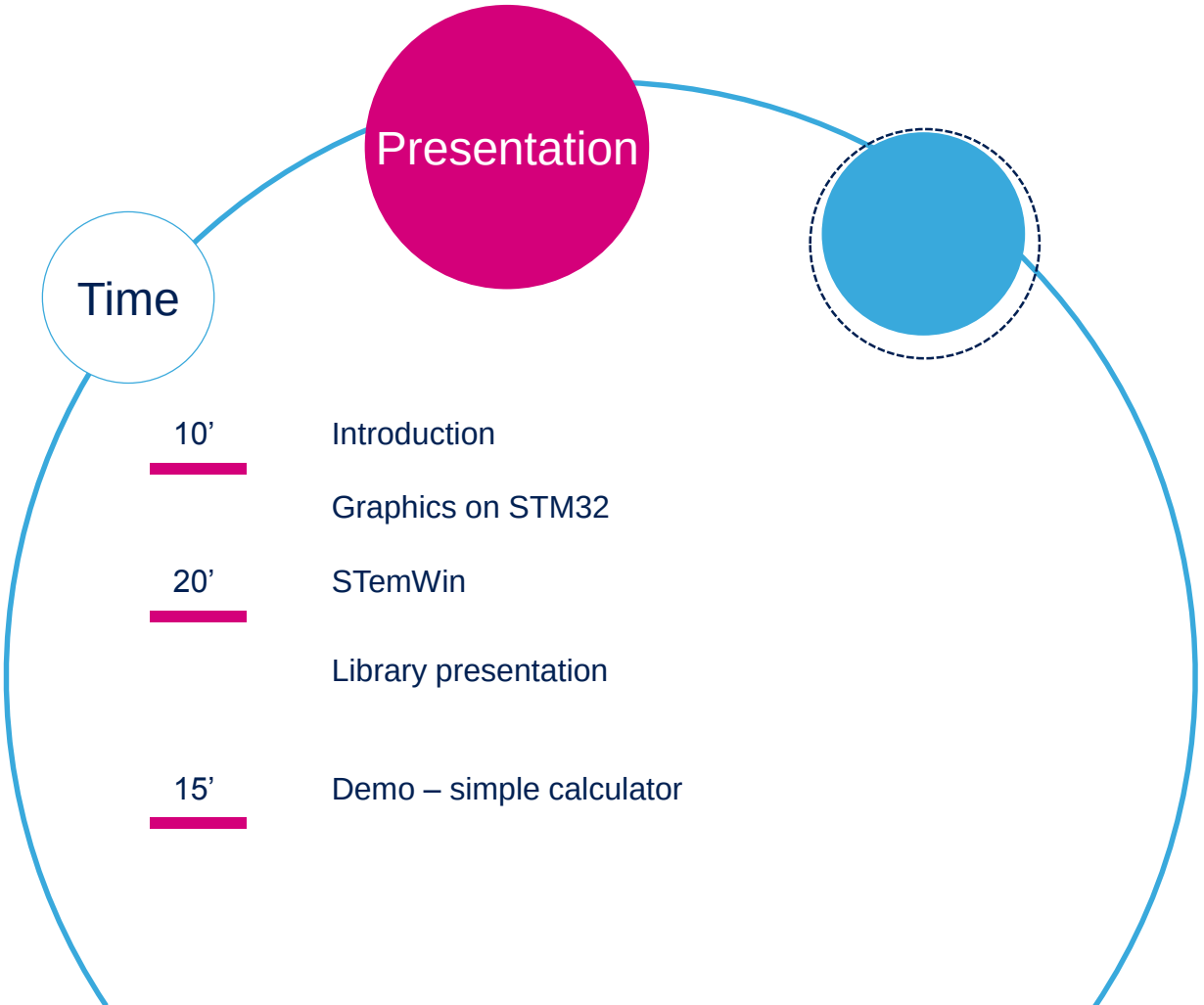
# STemWin

Hands-On on STM32F7

life.augmented

# Agenda



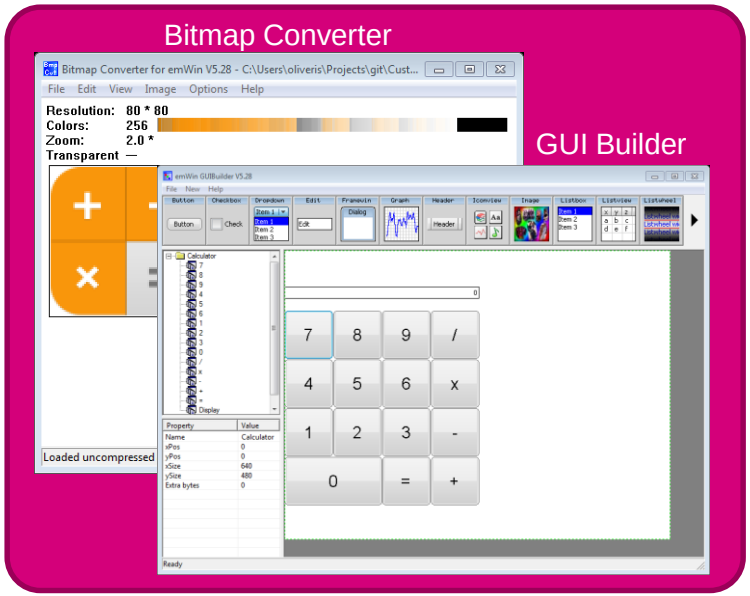

Presentation
Time
10’
Introduction
Graphics on STM32
20’
STemWin
Library presentation
15’
Demo – simple calculator

# Introduction

# STemWin – UI Development Kit



## Embedded user interface development made simple





STM32F756NG

- Chrom-ART Accelerator™
- ART Accelerator™
- Floating Point Unit (FPU)
- TFT LCD Controller
- Cortex-M7

STemWin Firmware Library

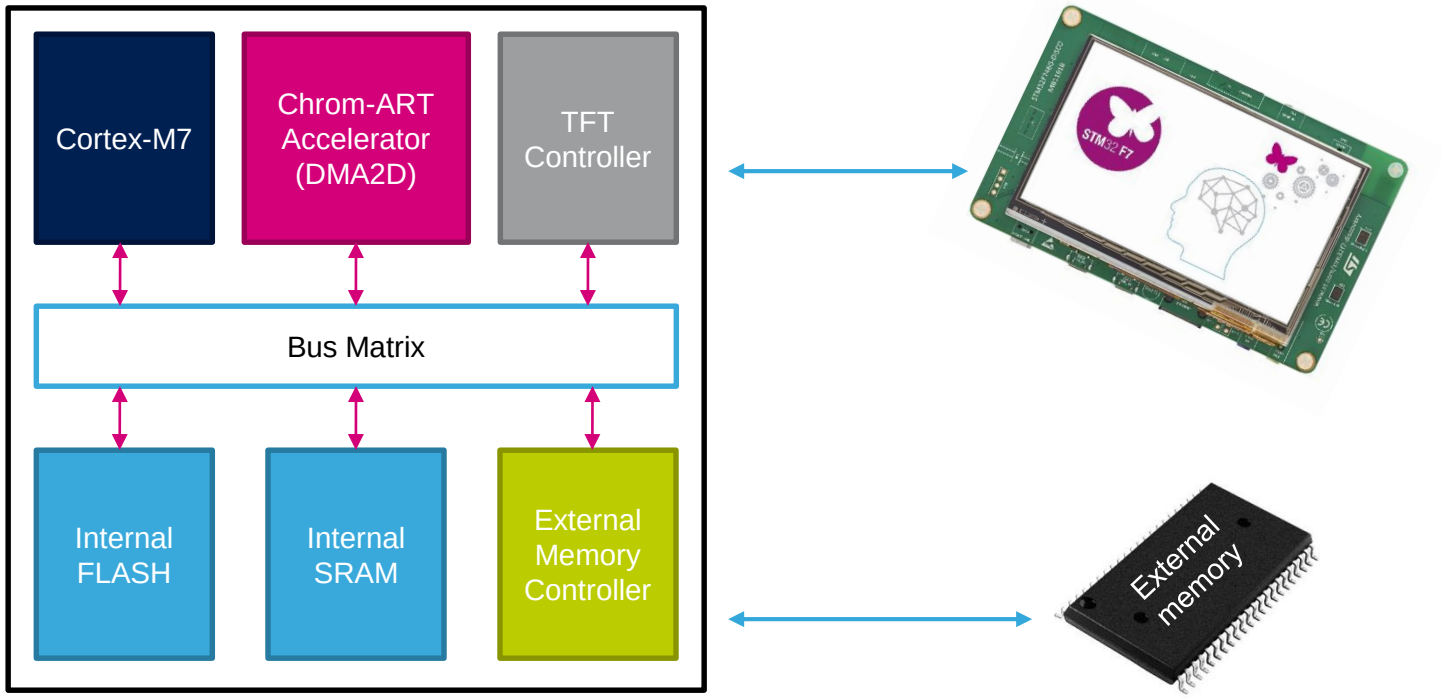
IDE

life.augmented

# Graphics on STM32



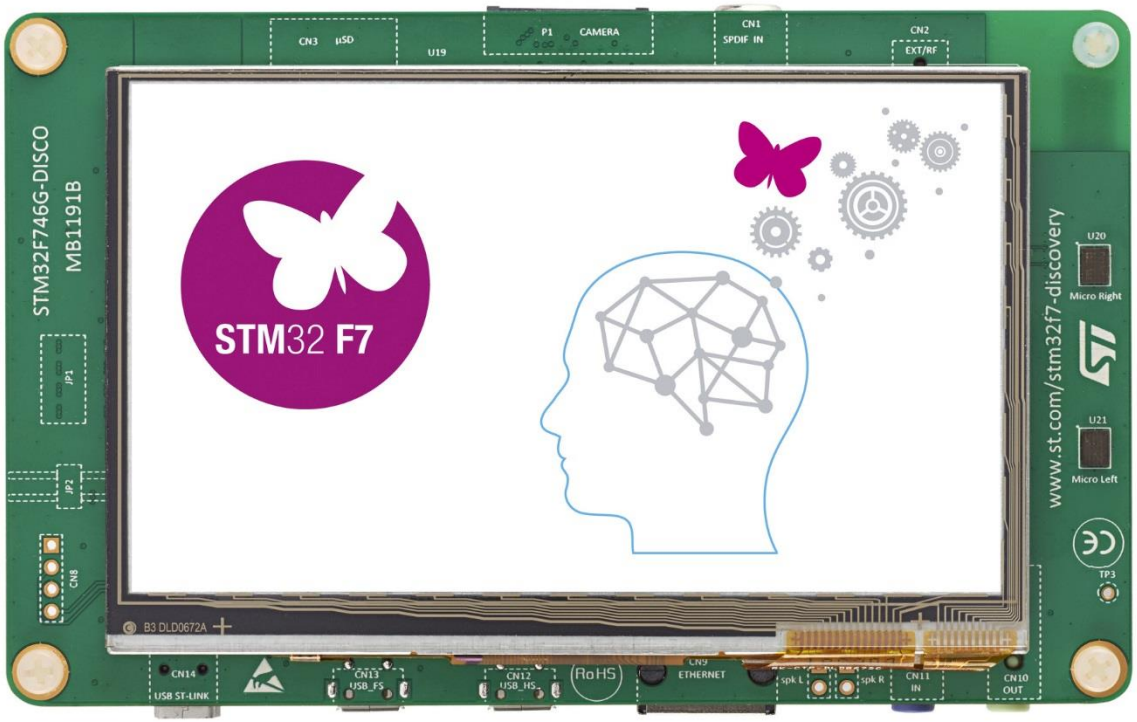
## STM32F756 Architecture




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

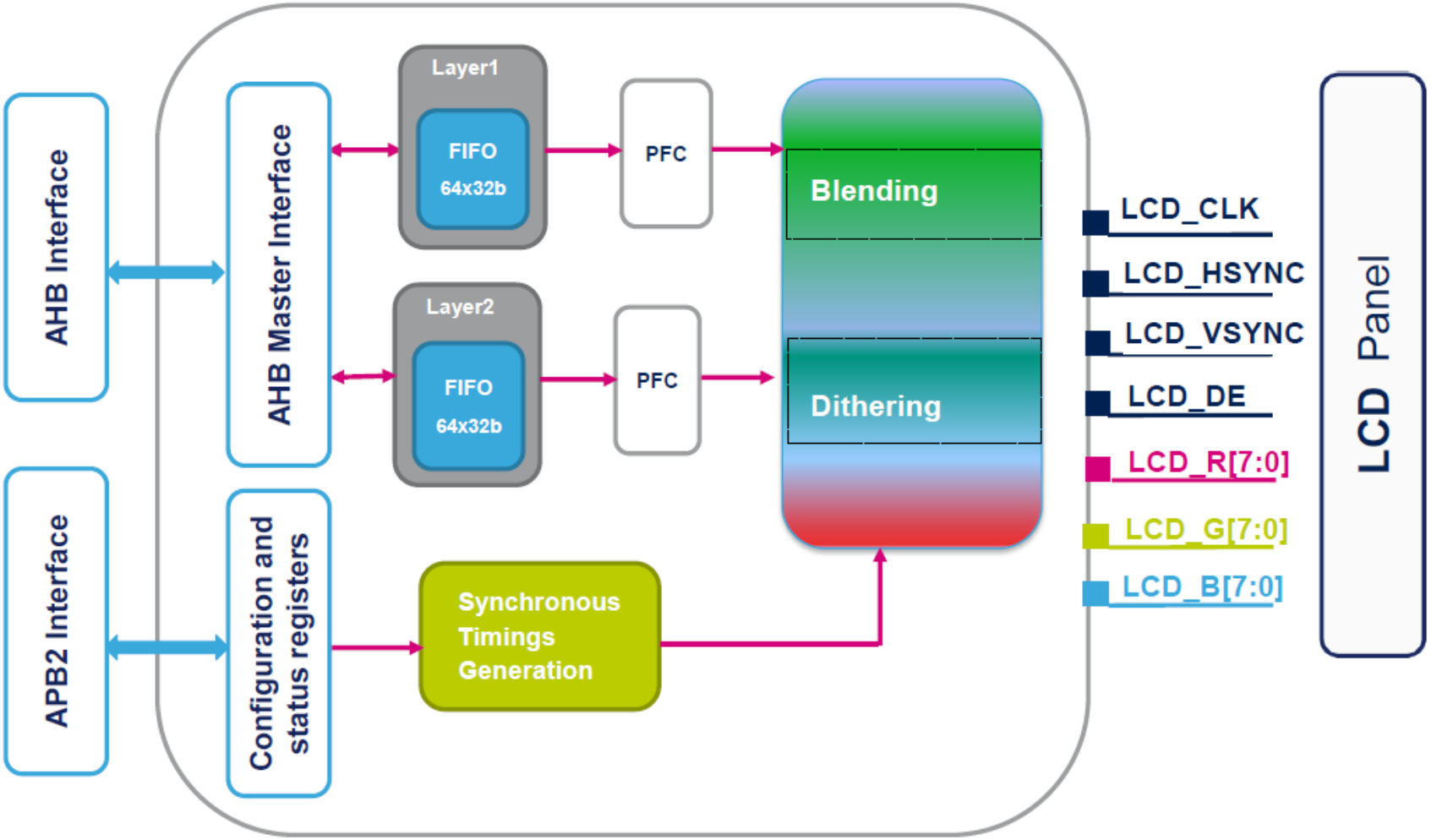
# LCD-TFT Display controller (LTDC)





# LCD-TFT Controller Architecture





# LCD Memory Requirements



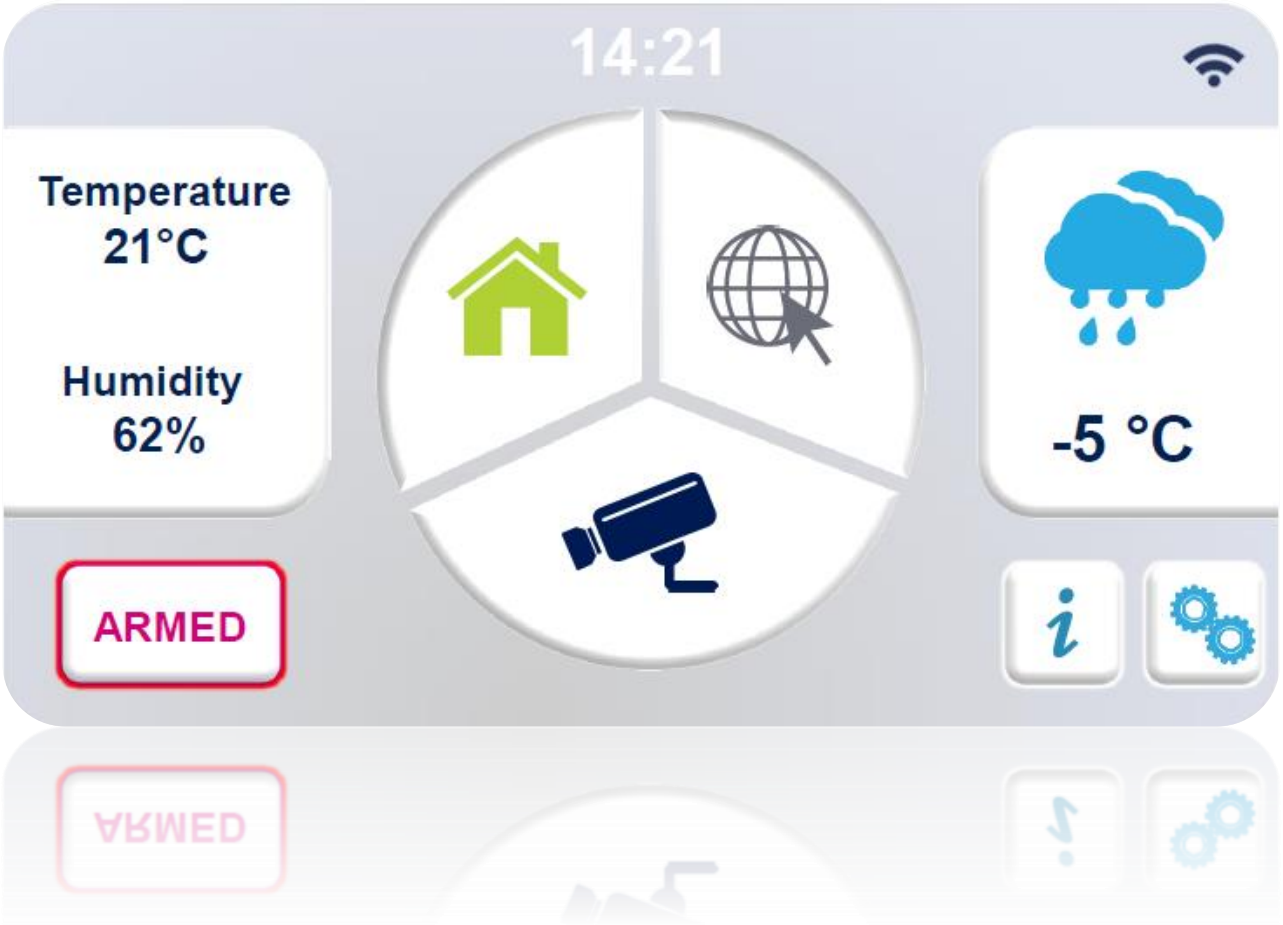
- Frame buffer size
  - The panel size and bits per pixel determine the amount of memory needed to hold the graphic buffer.
  - Memory Requirement (KiloBytes) = (bpp * Width * Height) / 8
  - In many cases, more memory is needed. E.g. double buffering: one graphic buffer to store the current image while a second buffer used to prepare the next image.

| Panel Resolution | Total Pixel (K) | bpp ( Bit per pixel) | Required memory (KB) |
|---|---|---|---|
| 320x240 (QVGA) | 76.8 | 16 | 153.6 |
| | | 8 | 76.8 |
| 480x272 (WQVGA) | 130.5 | 16 | 216.1 |
| 640x480 (VGA) | 307.2 | 16 | 614.4 |
| 800x600 (SVGA) | 480 | 16 | 960 |

# Chrom-ART Accelerator™



Hardware acceleration for Graphics content creation

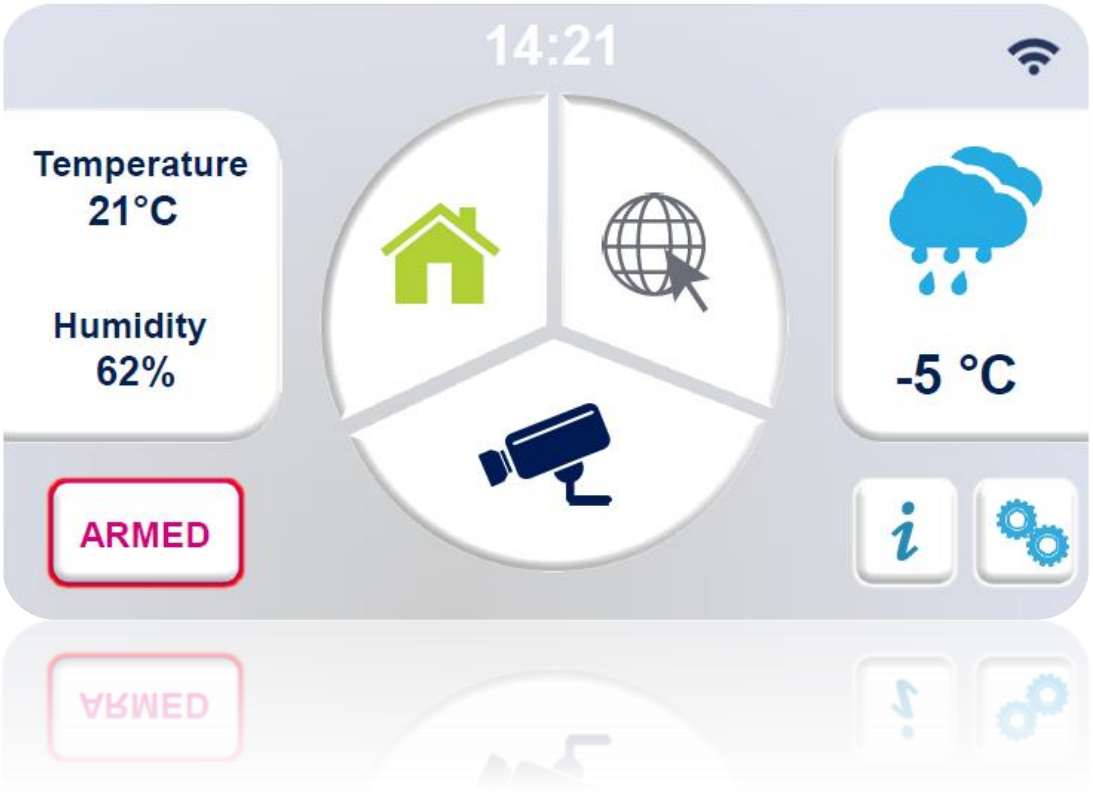



How the frame buffer is generated in order to creating “cool” graphical UI to be displayed through the TFT controller?

# Frame buffer construction (1/3)



The frame buffer is generated drawing successively all the graphics objects




Graphics object type:

- A **Bitmap** with its own color coding (different from the final one), compressed or not
- A **Vectorial Description** (a line, a circle with a texture...etc....)
- A **Text**

# Frame buffer construction (2/3)



**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

# Frame buffer construction (3/3)



- To generate the 24bpp frame buffer:
  - 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


- Many copy operations with pixel conversions can be done by the CPU, but it's very time consuming → HW acceleration helps

# Chrom-ART Accelerator™

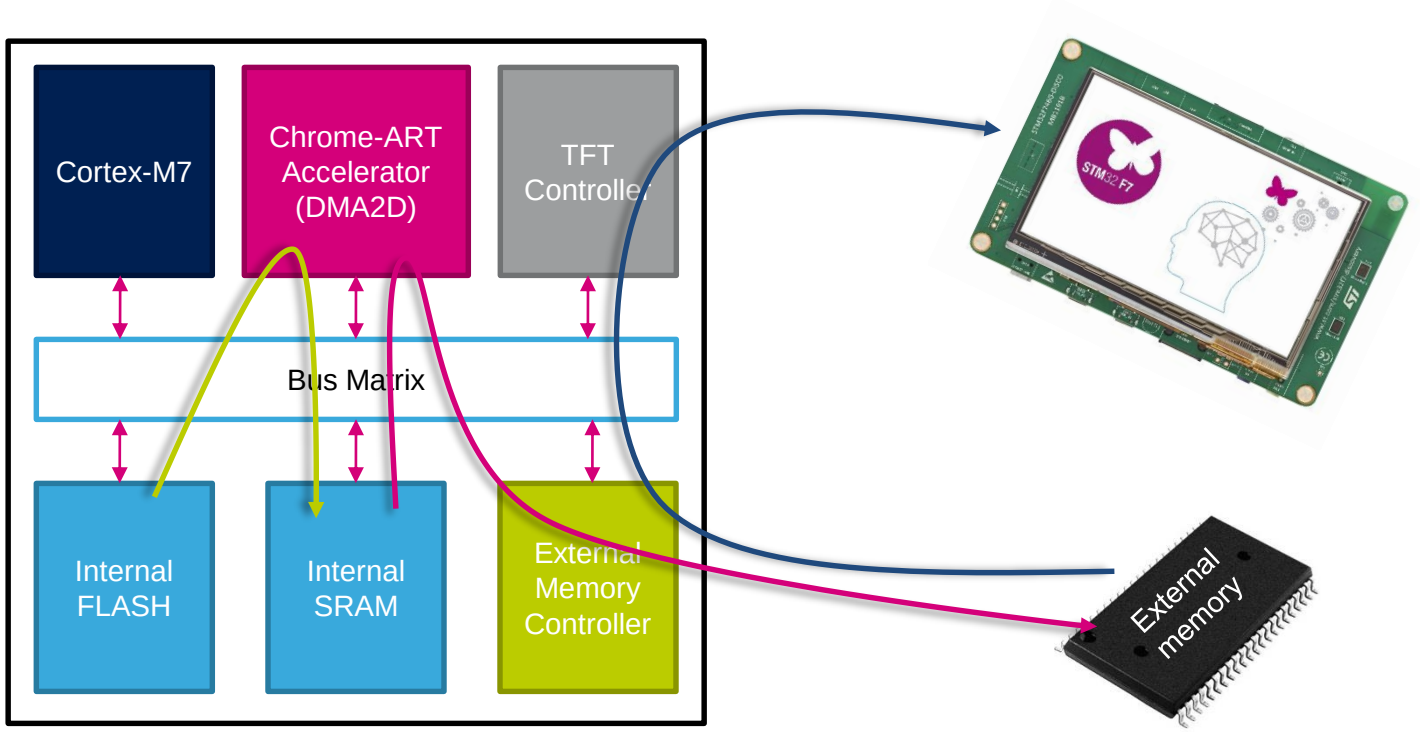


## Overview

- The Chrom-ART combines both a DMA2D and graphic 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 into another part of a graphic, or simply fill a part of a graphic 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

# STemWin

# Contents



- Objectives of the STemWin solution
- Basic 2D library
- PC SW Tools
- Window manager
- Configuration ?

# Why STemWin ?



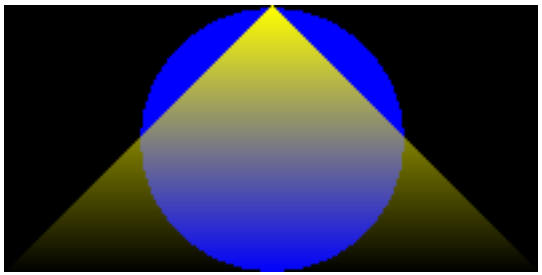
Provide a high level graphics library which:

- Is easy to use
- Is flexible to customize
- Provides graphic objects from simple 2D elements to complex window driven objects
- You can use the ST LTDC and Chrome-ART HW features without detailed knowledge
- Is the industry standard in embedded field
- Is FREE of charge!

# Basic 2D library



- With STemWin you can easily draw basic vector objects as
  - Lines, rectangles, arcs, polygons, graphs, bitmaps, JPEGs, PNGs and more
- Objects are drawn by a foreground color
  - `GUI_SetColor(GUI_RED);`
- STemWin supports Alpha blending – objects can overlay each other.

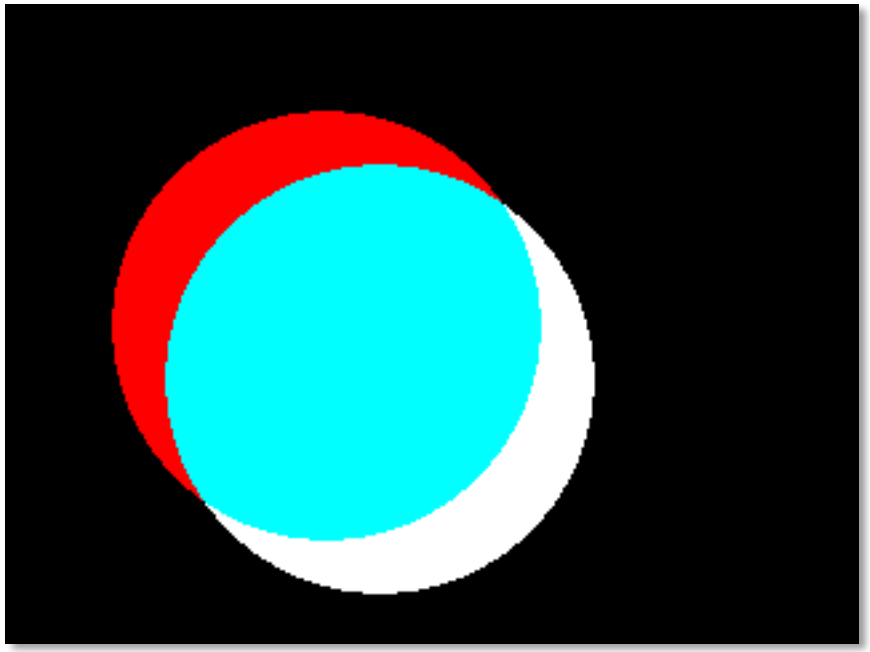


- Text and values can be easily written on the screen

# Basic 2D library - examples



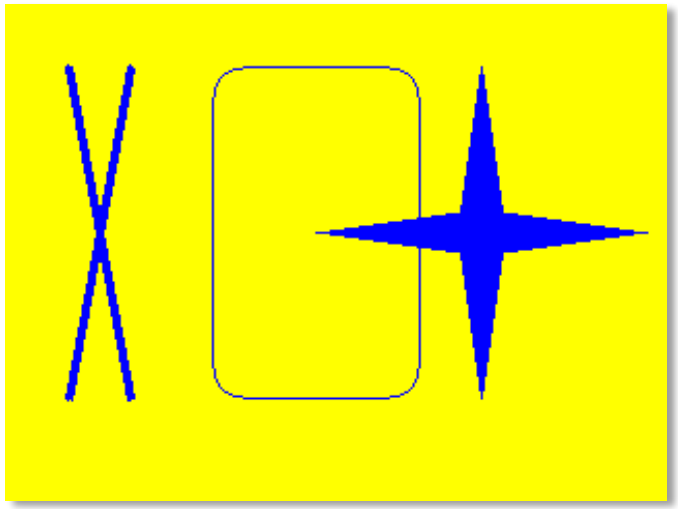
```
GUI_SetColor(GUI_RED);
GUI_SetDrawMode(GUI_DRAWMODE_NORMAL);
GUI_FillCircle(120, 120, 80);
GUI_SetDrawMode(GUI_DRAWMODE_XOR);
GUI_FillCircle(140, 140, 80);
```



# Basic 2D library - examples



```
GUI_SetBkColor(GUI_YELLOW);
GUI_Clear();
GUI_SetColor(GUI_BLUE);
GUI_SetPenSize(4);
GUI_DrawLine(30, 30, 60, 190);
GUI_DrawLine(30, 190, 60, 30);
GUI_DrawRoundedRect(100, 30, 200, 190, 15);
{
        const GUI_POINT aPoints[8] = {
                { 230, 30},{ 240, 100},
                { 310, 110}, { 240, 120},
                { 230, 190}, { 220, 120},
                { 150, 110}, { 220, 100},
        };
        GUI_FillPolygon(&aPoints, 8, 0, 0);
}
```



# Basic 2D library - examples



```
GUI_DrawPie(100, 100, 50, 0, 60, 0);
GUI_DrawPie(100, 100, 50, 120, 180, 0);
GUI_DrawPie(100, 100, 50, 240, 300, 0);

GUI_DrawArc(200, 200, 40, 0, 0, 180);
GUI_DrawArc(200, 200, 50, 0, 30, 90);
GUI_DrawArc(200, 200, 60, 0, 30, 90);
GUI_DrawArc(200, 200, 70, 0, 30, 90);
```

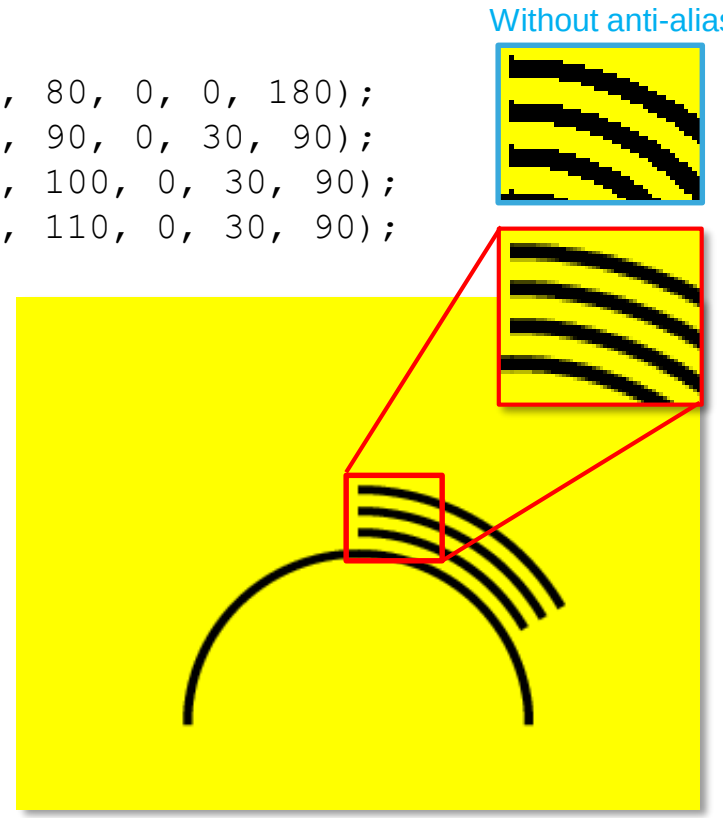
# Basic 2D library - Antialiasing



- Antialiasing smoothens curves and diagonal lines by "blending" the background color with that of the foreground.
  - Text
    - Font converter is required for creating AA fonts.
  - Circles
  - Arcs
  - Lines
  - Polygons

```
GUI_AA_SetFactor(4);
GUI_AA_DrawArc(160, 200, 80, 0, 0, 180);
GUI_AA_DrawArc(160, 200, 90, 0, 30, 90);
GUI_AA_DrawArc(160, 200, 100, 0, 30, 90);
GUI_AA_DrawArc(160, 200, 110, 0, 30, 90);
```

The higher the number of shades used between background and foreground colors, the better the antialiasing result (and the longer the computation time).

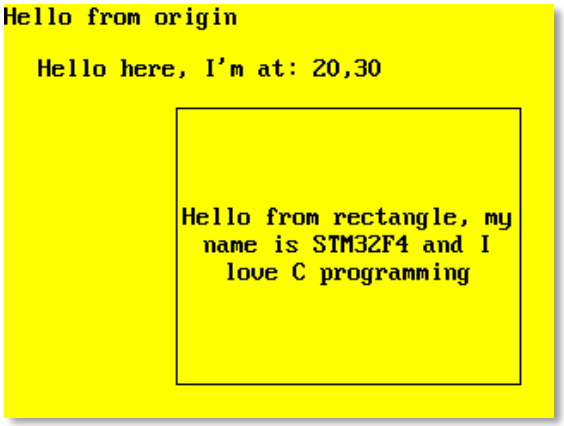


# Basic 2D library – text



- STemWin enables you to add text of any font into your GUI
- Several API functions are available to ease of use for text
  - Display text at specific position
  - Manage text inside a rectangle

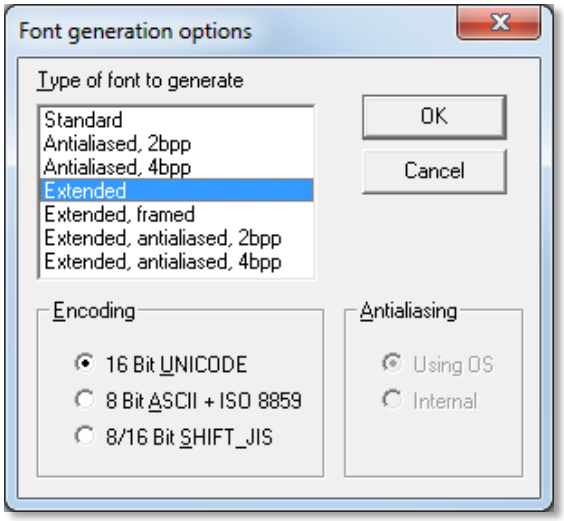
```
GUI_SetFont(&GUI_Font8x16);
GUI_DispString("Hello from origin");
GUI_DispStringAt("Hello here, I'm at: 20,30", 20,30);
{
  GUI_RECT pRect = {100, 60, 300, 220};
  GUI_DrawRect(100, 60, 300, 220);
  GUI_DispStringInRectWrap("Hello from
    rectangle, my   name is STM32F4 and I love
    C programming", &pRect, GUI_TA_VCENTER |
    GUI_TA_HCENTER, GUI_WRAPMODE_WORD);
}
```

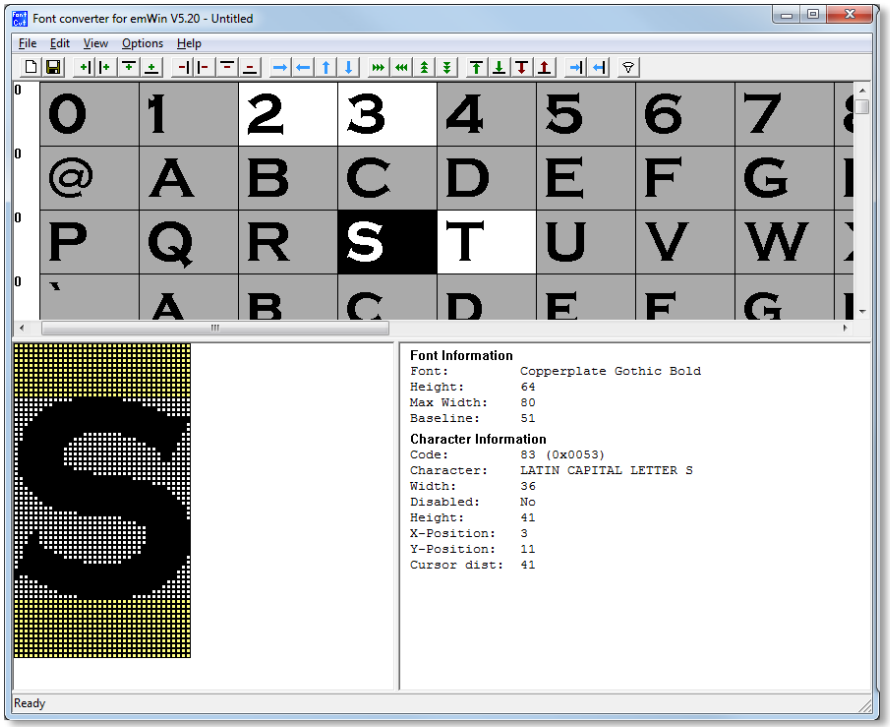


# PC SW Tools – Font converter



- You can manage your fonts in-application
  - Create fonts from any Windows font
  - Mange the number of characters so that you save only those you need → save ROM space
  - Export as .c files




life.augmented

# PC SW Tools – Font converter



## Using generated font

- Using the font we have generated is very easy
  - Include the generated .c file into the project
  - Include the external font declaration to all modules which will use it
  - Use GUI_SetFont() function to point STemWin to this font

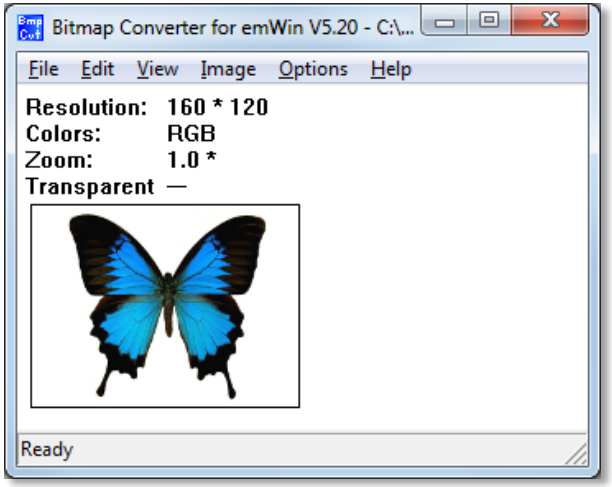
```
extern GUI_CONST_STORAGE GUI_FONT GUI_FontCopperplateGothicBold64;
extern GUI_CONST_STORAGE GUI_FONT GUI_FontCopperplateGothicBold64_aa;
..
GUI_RECT pRect1 = {0, 60, 319, 119};
GUI_RECT pRect2 = {0, 120, 319, 180};
GUI_SetFont(&GUI_FontCopperplateGothicBold64);
GUI_DispStringInRect("STM32", &pRect1, GUI_TA_VCENTER | GUI_TA_HCENTER);
GUI_SetFont(&GUI_FontCopperplateGothicBold64_aa);
GUI_DispStringInRect("STM32", &pRect2, GUI_TA_VCENTER | GUI_TA_HCENTER);
```

Drawing anti-aliased text takes much more time! (and memory as well)

# PC SW Tools – Bitmap converter



- Static bitmaps or streamed bitmaps are supported
  - **Static bitmap** is completely available during drawing (e.g. stored in Flash memory)
  - **Streamed bitmaps** are available only by parts received by the MCU (e.g. reception from data card).

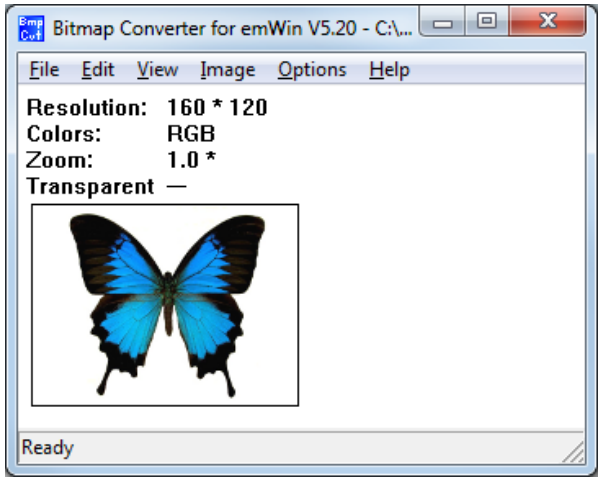


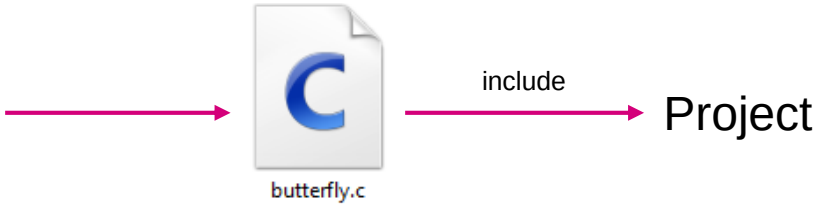

- Supported formats
  - Internal bitmap format
  - BMP
  - JPEG
  - PNG
  - GIF
- **BmpCvt.exe** can be used for bitmap conversion and storage to .c files
- **Bin2c.exe** can be used to store any binary in .c form, e.g. complete bitmap or jpeg files

# PC SW Tools – Bitmap converter



## Using generated image file



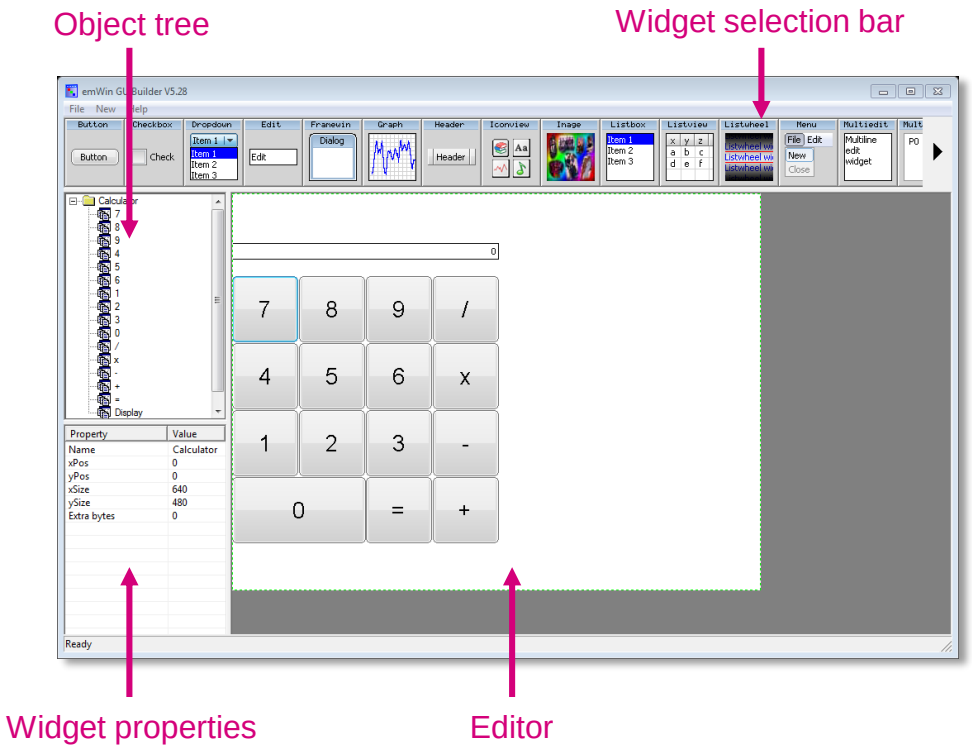

```
extern GUI_CONST_STORAGE GUI_BITMAP bmbutterfly;
GUI_DrawBitmap(&bmbutterfly, 30, 30);
```

Storing a bitmap in the same format as the LCD increases drawing performance → no pixel conversion needed, but may need more storage space.

# PC SW Tools – GUIBuilder



- Dialog creation made simple
- No C knowledge is needed
- Support to Drag & Drop
- Fine tune UI elements by editing its properties



# Window Manager



## What is the **W**indow **M**anage?

- Management system for a hierarchic window structure
  - Each layer has its own desktop window. Each desktop window can have its own hierarchy tree of child windows.
- Callback mechanism based system
  - Communication is based on an event driven callback mechanism. All drawing operations should be done within the WM_PAINT event.
- Foundation of widget library
  - All widgets are based on the functions of the WM.
- Basic capabilities:
  - Automatic clipping
  - Automatic use of multiple buffers
  - Automatic use of memory devices
  - Automatic use of display driver cache
  - Motion support

# Window Manager



## callback mechanism

The callback mechanism requires a callback routine (*window procedure*) for each window. These routines have to support the following:

- **Painting the window**
  - Each window has to draw itself. This should be done when receiving a **WM_PAINT** message.
- **Default message handling**
  - Plain windows need to call the function WM_DefaultProc() to avoid undefined behavior of the window.

Further the WM needs to 'stay alive'. This can be done within a simple loop after creating the windows. It has nothing to do but calling `GUI_Delay()` or `GUI_Exec()` which does the following:

- **PID management**
- **Key input management**
- **Timer management**

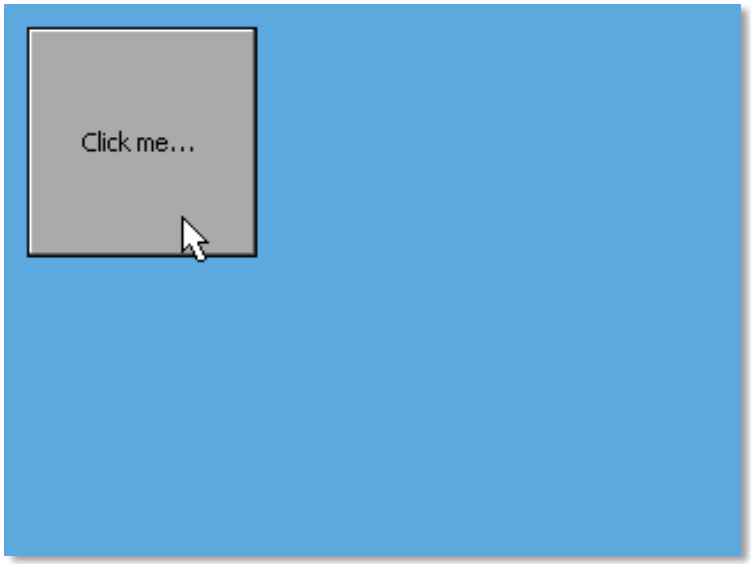
```
hButton = BUTTON_Create( 10, 10, 100, 100, GUI_ID_BUTTON0, WM_CF_SHOW);
BUTTON_SetText(hButton, "Click me...");
WM_SetCallback(WM_HBKWIN, myCbBackgroundWin);
WM_SetCallback(hButton, myCbButton);
```

Initialization – called only once!

```
... ... ...
static void myCbBackgroundWin(WM_MESSAGE *pMsg) {
  int NCode, Id;
  switch (pMsg->MsgId) {
  case WM_NOTIFY_PARENT:
    Id = WM_GetId(pMsg->hWinSrc);  /* Id of widget */
    NCode = pMsg->Data.v;          /* Notification code */
    if ((Id == GUI_ID_BUTTON0) && (NCode == WM_NOTIFICATION_RELEASED))
     buttonClicked = 0;
    break;
  case WM_PAINT:
    GUI_SetBkColor(STBLUE);
    GUI_Clear();
    break;
  }
}
```

Redraw code

On release WM calls parent callback function with message informing about child touch release




```
... ... ...
static void myCbButton(WM_MESSAGE *pMsg) {
  switch (pMsg->MsgId) {
  case WM_TOUCH:
    if (((GUI_PID_STATE*)(pMsg->Data.p))->Pressed == 1)
    buttonClicked = 1;
    break;
  case WM_SIZE:
    // add some code
    break;
  default:
    BUTTON_Callback(pMsg);
  }
}
```

On click WM calls Button callback function

Default callback must be called to achieve proper functionality

# Window Manager



## Widget library

- Widget = **Wi**ndow + Ga**dget**
- Currently the following widgets are supported:
  - Button, Checkbox, Dropdown, Edit, Framewin, Graph, Header, Iconview, Image, Listbox, Listview, Listwheel, Menu, Multiedit, Progbar, Radio, Scrollbar, Slider, Text, Treeview
- Creating a widget can be done with one line of code.
- There are basically 2 ways of creating a widget:
  - **Direct creation**

    For each widget there exist creation functions:

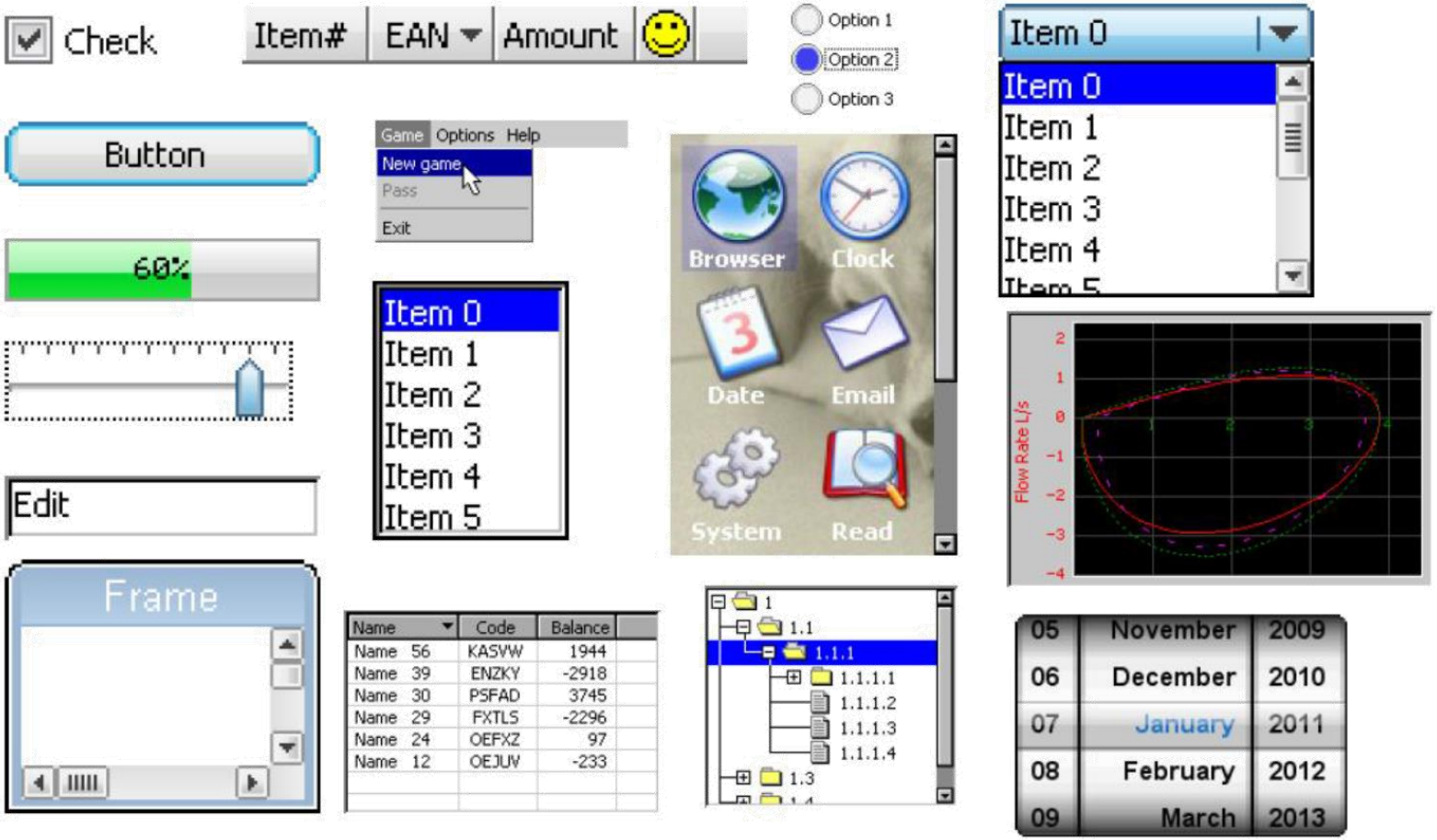
    <WIDGET>_CreateEx() // Creation without user data.

    <WIDGET>_CreateUser() // Creation with user data.
  - **Indirect creation**
    - A widget only needs to be created indirectly if it is to be included in a dialog box.
      <WIDGET>_CreateIndirect() // Creates a widget to be used in dialog boxes.

# Window Manager



## Available Widgets



# Demo

# STemWin Example project



- Download
  - **www.st.com -> search “stm32cube”**

| | | |
|---|---|---|
| STM32CubeF4 | Embedded Software | Embedded software for STM32F4 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards: STM32 Nucleo, Discovery kits and Evaluation boards) |
| STM32CubeF7 | Embedded Software | Embedded software for STM32F7 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards: STM32 Nucleo, Discovery kits and Evaluation boards) |

| Part Number ▲ | Software Version | Marketing Status | Supplier | Order from ST |
|---|---|---|---|---|
| STM32CubeF7 | 1.3.0 | Active | ST | Get Software |

# Thank You

life.augmented