STM32GUI_使用 TouchGFX 动态图片功能实现动态更换表盘背景功能

关键字：TouchGFX，GUI，动态图片，更换表盘背景

1. 引言

自从 2013 年第一款侧重于 GUI 应用的 STM32F429x【搭载 Chrom-ART 图形加速和 LTDC 控制器】开始，ST 提供了 STM32MCU + X-Cube-TouchGFX 一站式 GUI 开发平台，越来越多的客户使用 STM32 + TouchGFX 开发智能手表/智能家居控制面板等嵌入式设备.

对于智能手表应用，由于可以通过无线方式与手机进行通信，因此动态表盘背景更新也成为一个很好增加用户体验的功能(如可以根据节日更新表盘背景).

下面我们用一个例程来介绍下如何使用 TouchGFX 动态位图来实现这一个功能.

2. 例程开发步骤如下

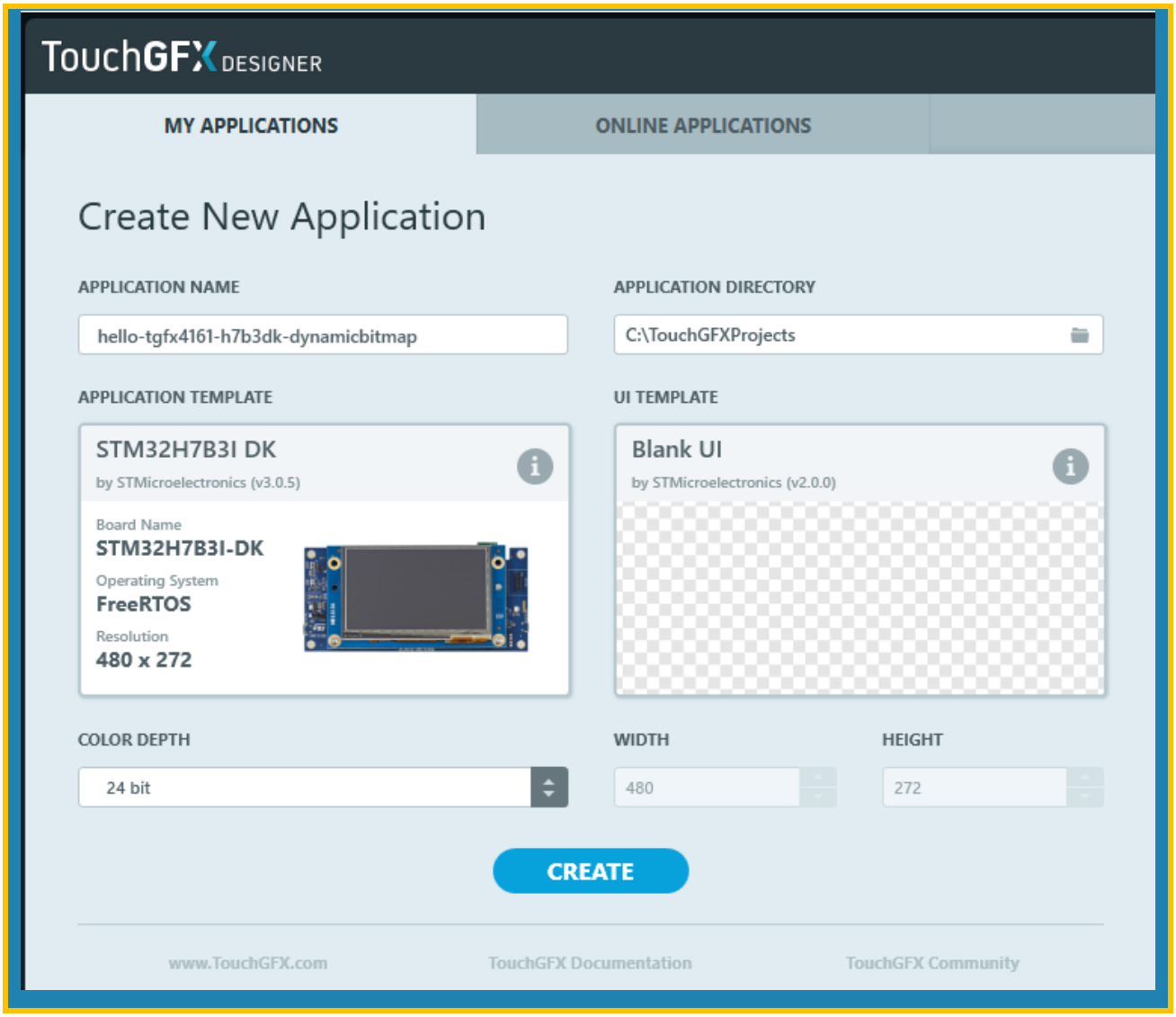
2.1 开发环境安装 ：

- TouchGFX4.16.1 (本文使用 4.16.1 举例，其他版本操作过程相同)
  - 环境安装请参考网址: https://support.touchgfx.com/docs/introduction/installation
- VSCode
- STM32H7B3 探索板

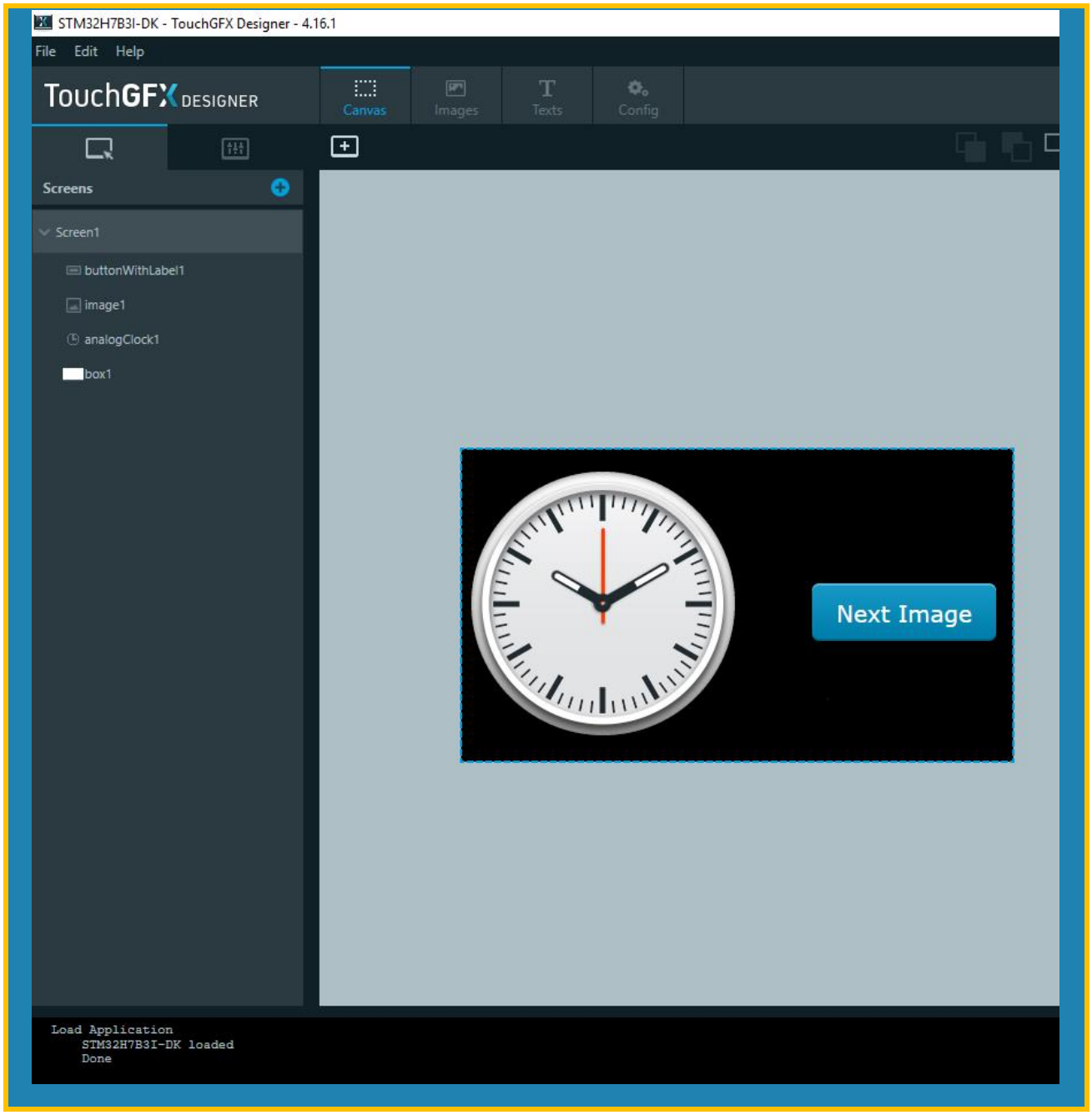
2.2 例程设计流程

2.2.1.打开 TouchGFX Designer 4.16.1 :

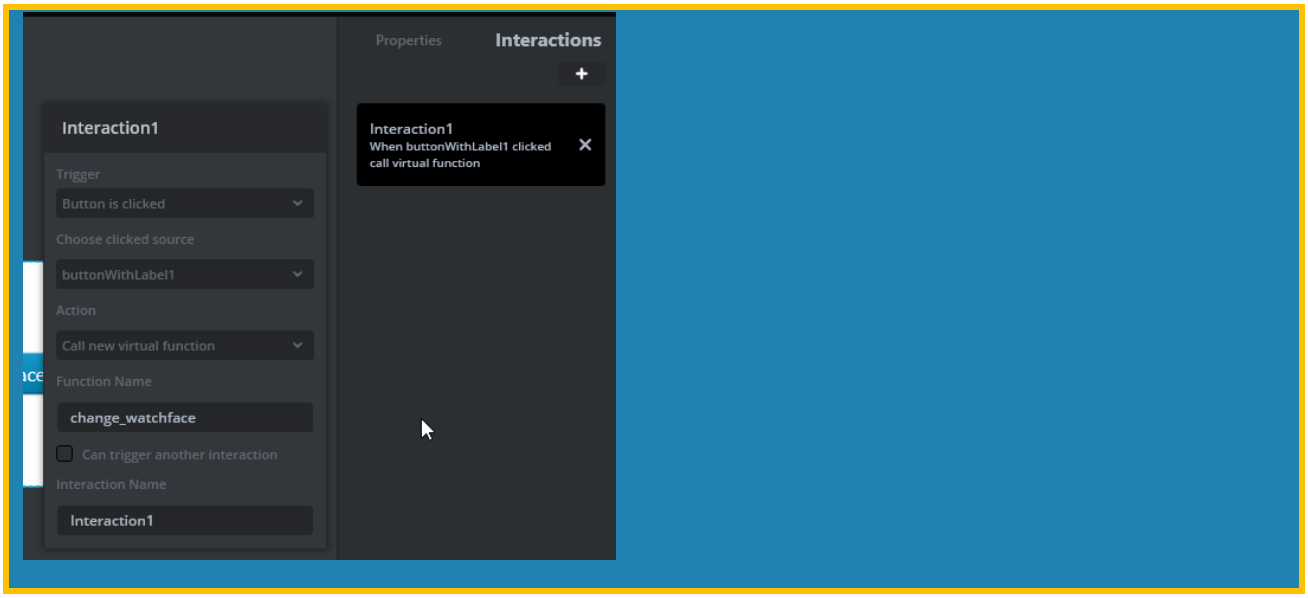
2.2.2 选择 STM32H7B3 探索板,生成工程 :



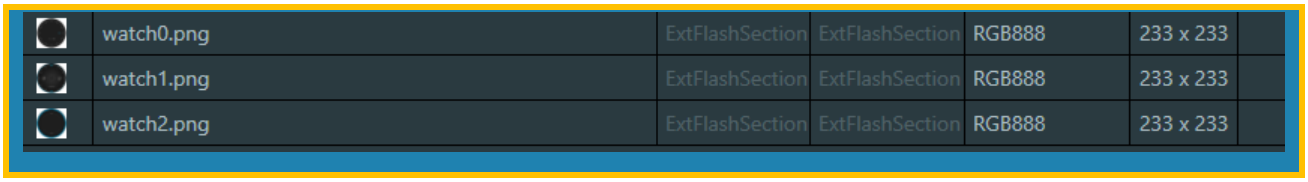
2.2.3 设计界面,依次增加 box/ analogClock / image / button



2.2.4 为 button 增加交互,按键按下 call 虚函数 change_dimage，用于触发更换表盘背景内容事件



2.2.6 将准备好的表盘图片拖入到 TouchGFX Designer 的 Image 中，并修改图片格式为 RGB888



2.2.7 由于创建动态位图时，像素内存从位图缓存中分配。因此，必须先配置位图缓存，然后才能创建动态位图，这是无法在 TouchGFX 设计器或生成器中完成的手动过程.

1) 使用 vscode 打开代码，添加动态图片相关代码,在 FrontendApplication.cpp 中为动态 Bitmap 申请内存:

- PC 仿真环境

```
#include <gui/common/FrontendApplication.hpp>

FrontendApplication::FrontendApplication(Model& m, FrontendHeap& heap)
   : FrontendApplicationBase(m, heap)
{
#ifdef SIMULATOR
      //PC 仿真环境
   const uint32_t cacheSize = 233*233*3*2;
   uint16_t* const cacheStartAddr = (uint16_t*)malloc(cacheSize);
   Bitmap::setCache(cacheStartAddr, cacheSize, 4);
#else
      //MCU 环境
   /*
   RAM          0x24000000       0x00040000       xrw
   RAMFB        0x24040000       0x000c0000       xrw
   BMP_CACHE      0x24040000+0xC000
   */
   // Place cache start address in SDRAM at address 0x24040000+0xC0000;
   uint16_t* const cacheStartAddr = (uint16_t*)(0x24040000+0xC000);
   const uint32_t cacheSize = 233*233*3*2;
   Bitmap::setCache(cacheStartAddr, cacheSize, 4);
#endif
```

- MCU 环境 :

```
#include <gui/common/FrontendApplication.hpp>

FrontendApplication::FrontendApplication(Model& m, FrontendHeap& heap)
    : FrontendApplicationBase(m, heap)
{
#ifdef SIMULATOR
    //PC 仿真环境
    const uint32_t cacheSize = 233*233*3*2;
    uint16_t* const cacheStartAddr = (uint16_t*)malloc(cacheSize);
    Bitmap::setCache(cacheStartAddr, cacheSize, 4);
#else
    //MCU 环境
    /*
    RAM             0x24000000          0x00040000          xrw
    RAMFB           0x24040000          0x000c0000          xrw
    BMP_CACHE       0x24040000+0xC000
    */
    // Place cache start address in SDRAM at address 0x24040000+0xC0000;
    uint16_t* const cacheStartAddr = (uint16_t*)(0x24040000+0xC000);
    const uint32_t cacheSize = 233*233*3*2;
    Bitmap::setCache(cacheStartAddr, cacheSize, 4);
#endif
}
```

2) 在 Screen1View.hpp 定义变量和虚函数：

```
#ifndef SCREEN1VIEW_HPP
#define SCREEN1VIEW_HPP

#include <gui_generated/screen1_screen/Screen1ViewBase.hpp>
#include <gui/screen1_screen/Screen1Presenter.hpp>
#include <touchgfx/widgets/Image.hpp>
class Screen1View : public Screen1ViewBase
{
public:
    Screen1View();
    virtual ~Screen1View() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
    virtual void handleTickEvent();
    virtual void change_dimage();
    unsigned char image_index;
protected:
private:
    Image image;
    unsigned int tick_cnt ;

    unsigned int width ;
    unsigned int height;
    BitmapId bmpId;
};

#endif // SCREEN1VIEW_HPP
```

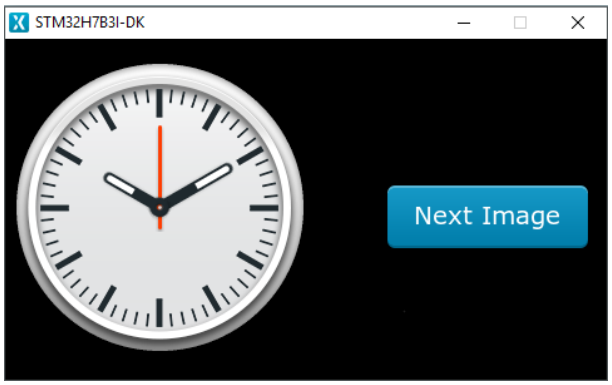
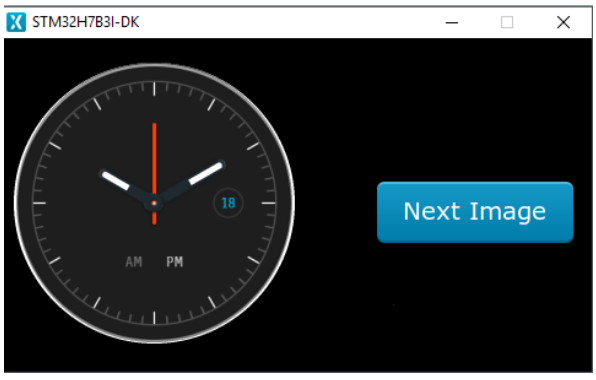
3) 在 Screen1View.cpp 中进入 screen 函数 setupScreen 函数添加创建动态图片，并获得 bmpId, 供其他控件调用：

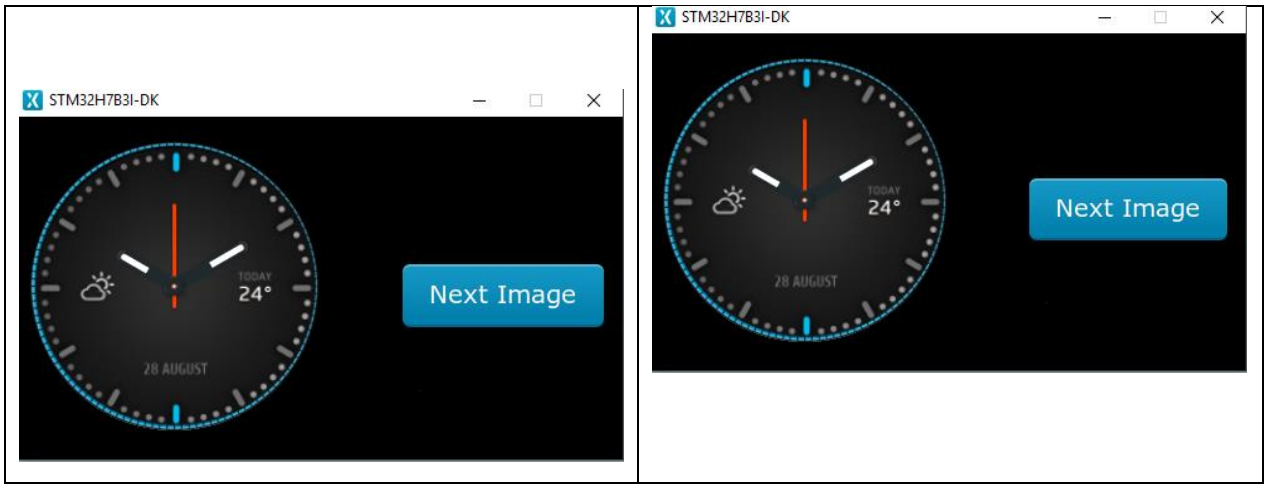
```
#include <gui/screen1_screen/Screen1View.hpp>
#include "string.h"
extern const unsigned char image_watch0[] ;
extern const unsigned char image_watch1[] ;
extern const unsigned char image_watch2[] ;
Screen1View::Screen1View()
{

}

void Screen1View::setupScreen()
{
    Screen1ViewBase::setupScreen();
    width = 233;
    height = 233;
    bmpId = Bitmap::dynamicBitmapCreate(width, height, Bitmap::RGB888);
}
```

```
void Screen1View::tearDownScreen()
{
    Screen1ViewBase::tearDownScreen();
}

void Screen1View::handleTickEvent()
{
    if(tick_cnt++>60)
    {
        tick_cnt = 0;

    }
}

void Screen1View::change_dimage()
{
    if (image_index == 0)
    {
        memcpy(Bitmap::dynamicBitmapGetAddress(bmpId), image_watch0, width*height * 3);    image_index =
1;
    }
    else
    if (image_index == 1)
    {
        memcpy(Bitmap::dynamicBitmapGetAddress(bmpId), image_watch1, width*height * 3);    image_index =
2;
    }
    else
    if (image_index == 2)
    {
        memcpy(Bitmap::dynamicBitmapGetAddress(bmpId), image_watch2, width*height * 3);    image_index =
0;
    }
    analogClock1.setBackground(bmpId, width, height);
    analogClock1.invalidate();
}
```

2.2.8 编译运行结果如下：

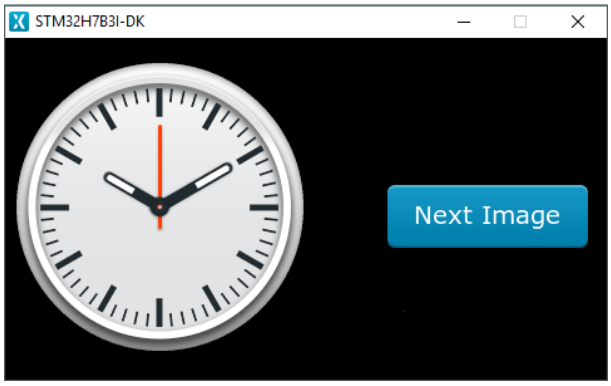
- PC 仿真环境：

.默认表盘	动态表盘 0
	
动态表盘 1	动态表盘 2

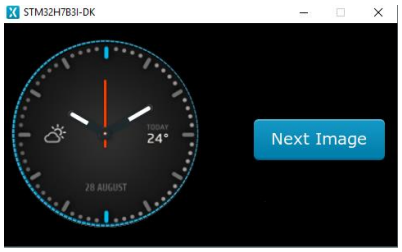


- MCU 环境：

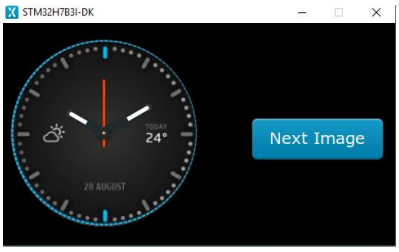
1.默认表盘



动态表盘 1



动态表盘 2



2.小结

通过以上的示例我们可以看到, 使用 TouchGFX 提供的动态图片功能， 很容易就可以实现将一个数据流转换为 TouchGFX 引擎可用的素材, 并显示到界面上.

当然， 除了可以实现离线更换表盘,还可以实现远程 OTA 为用户推送更多交互体验（如更新壁纸/主题）等功能， 不断为终端用户提供更多体验和服务.

版本历史

日期	版本	变更
2022 年 04 月 11 日	1.0	首版发布

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。