使用 TouchGFX 二进制翻译(Binary Translation)功能实现动态更新翻译

关键字：TouchGFX, GUI, Binary Translation, TouchGFX 二进制翻译

1. 引言

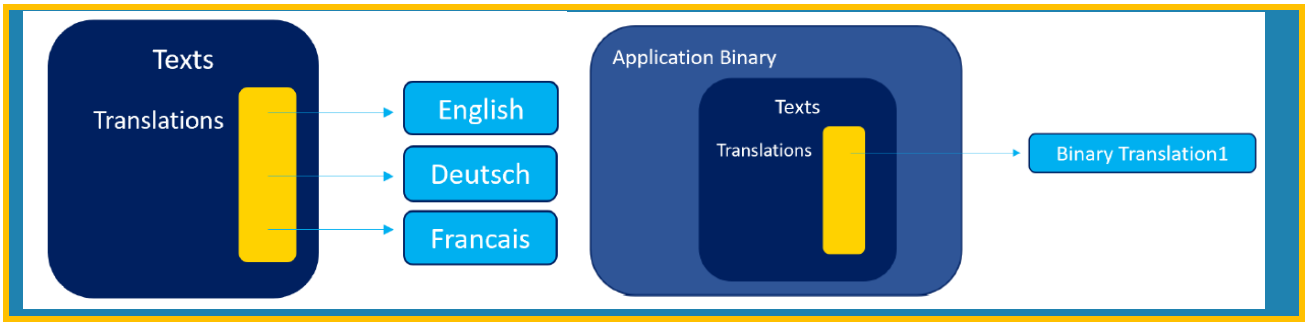
从 2013 年第一款侧重于 GUI 应用的 STM32F429x【内嵌 Chrom-ART 图形加速和 LTDC 控制器】开始，ST 就提供了 STM32MCU + X-Cube-TouchGFX 一站式 GUI 开发平台，越来越多的客户使用 STM32 + TouchGFX 开发智能手表/智能家居控制面板等嵌入式设备。

在嵌入式GUI应用中，设备厂家希望提供功能支持用户根据不同国家来定义使用的字体翻译, 从技术上支持将字体翻译和应用程序分离开。这样可以在嵌入式设备有限的存储空间上，根据不同国家的需要，来更新和升级字体翻译的类型。

2. TouchGFX Binary translation 介绍

TouchGFX 二进制翻译功能将文本翻译与应用程序分开。二进制翻译是保存在单独的二进制文件中，这些文件可以被编程到闪存或存储在 SD 卡上，这在处理大量翻译时为应用程序开发人员提供了更大的灵活性。

TouchGFX 中的 Text 类包含一个指针数组，其中包含指向应用程序中每种语言的翻译表的指针, 翻译表原则上是该语言中使用的所有字符串的集合。在运行时使用二进制翻译时，需要将映射从编译时翻译更改为二进制翻译（二进制翻译必须存放在可寻址内存中（闪存或 RAM））



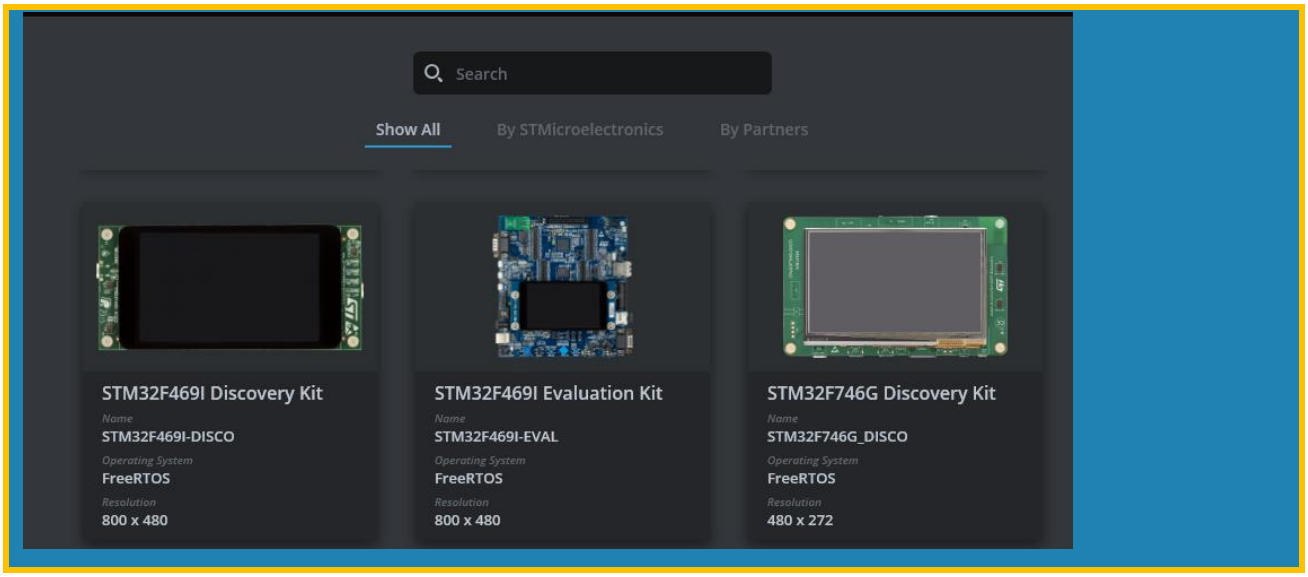
3. 例程开发步骤如下

3.1. 打开 TouchGFX Designer 4.18.0

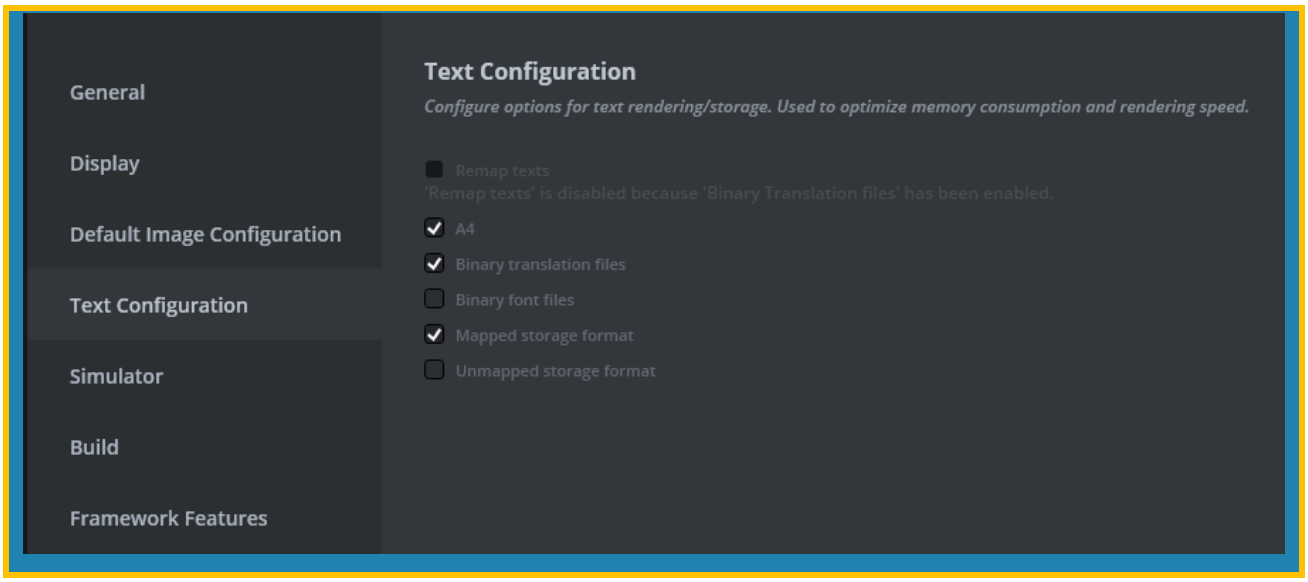
- TouchGFX4.18.0 (本文使用 4.18.0 举例，其他版本操作过程相同)
  - 环境安装请参考网址: https://support.touchgfx.com/docs/introduction/installation
- VSCode

3.2.打开 TouchGFX Designer 4.18.0

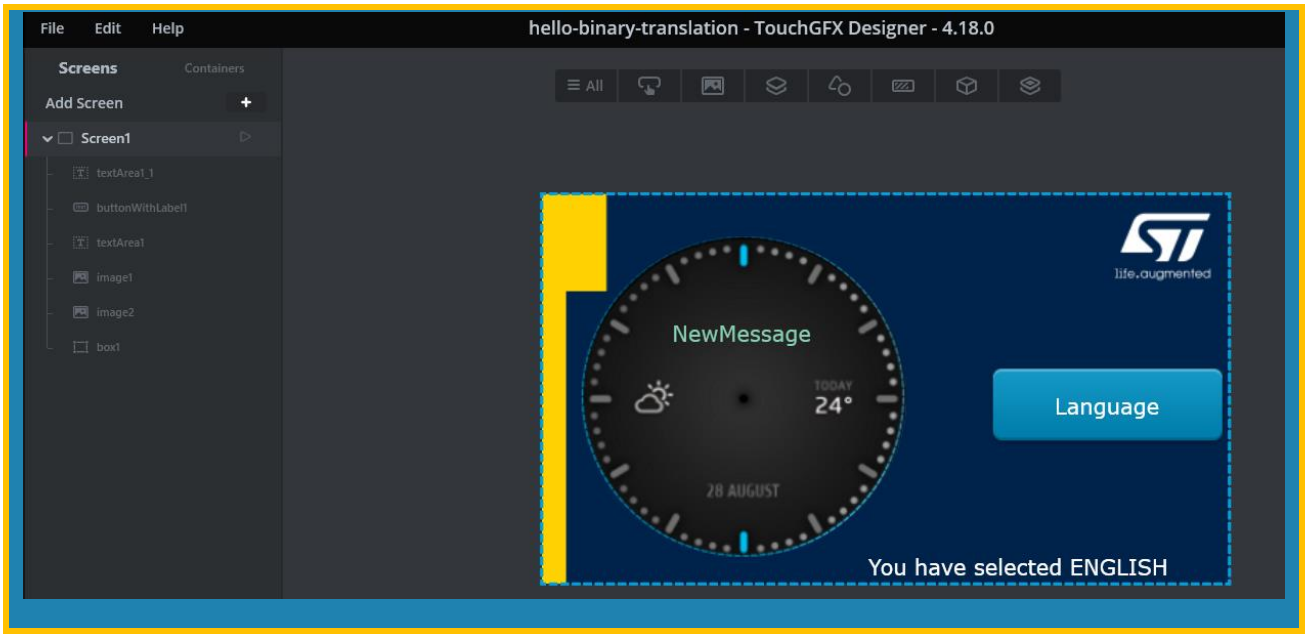
3.2.选择 STM32F746DK 探索板,生成工程：



3.3 选择 “文本配置”，选择 “Binary translation files”

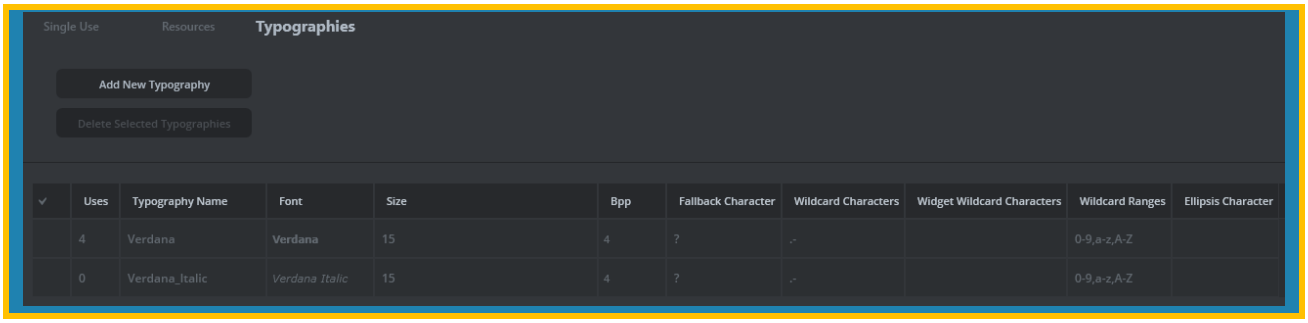


3.4 设计 Screen1 (添加 Images/TextArea/Buttons)

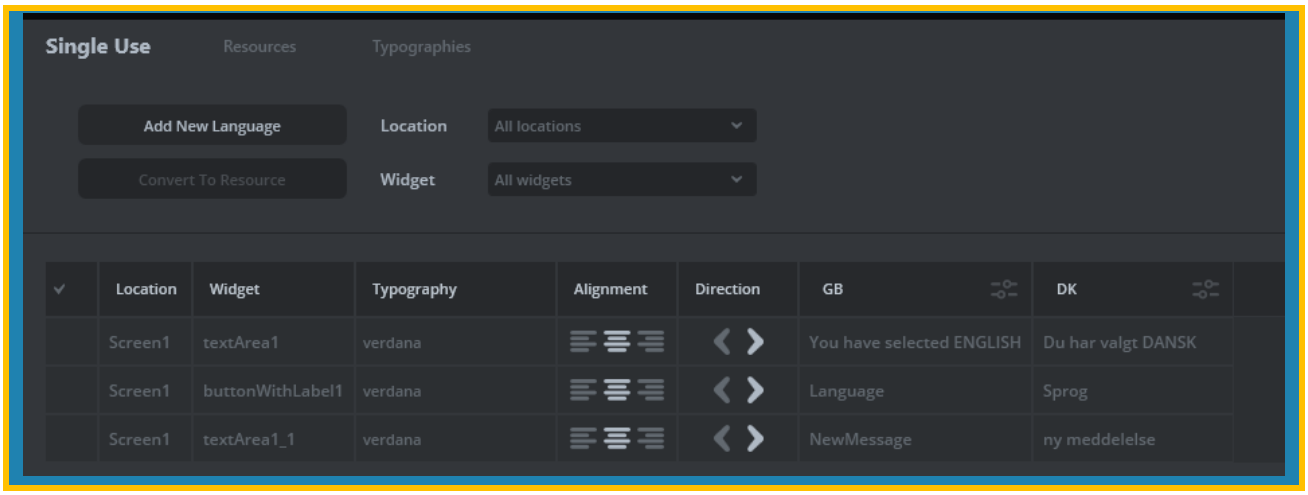


3.5 设置 Typographies 语言：

- Typographies 设置：

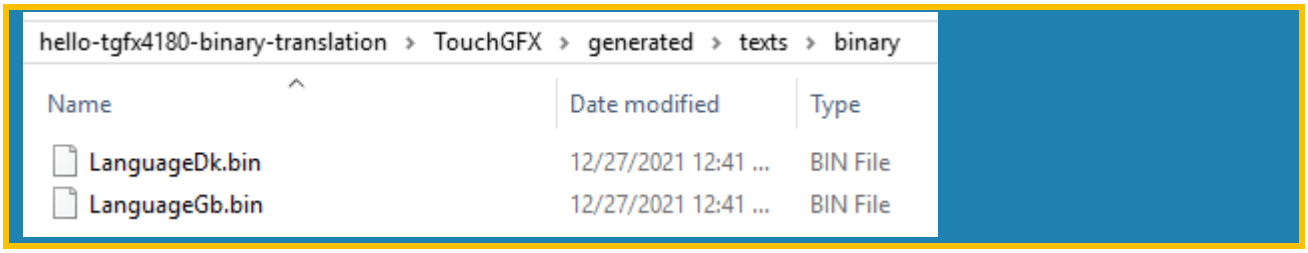


-Single Use 设置，添加 DK 语言翻译(本示例以丹麦语举例):



3.6 TouchGFX Designer 生成项目

可以在以下目录找到GB和DK对应的二进制翻译文件：



3.7 在代码中安装二进制翻译

当使用二进制翻译时，编译到应用程序中的翻译为空。因此在使用文本之前，需要将二进制翻译加载到内存中，并在 TouchGFX 中进行安装。

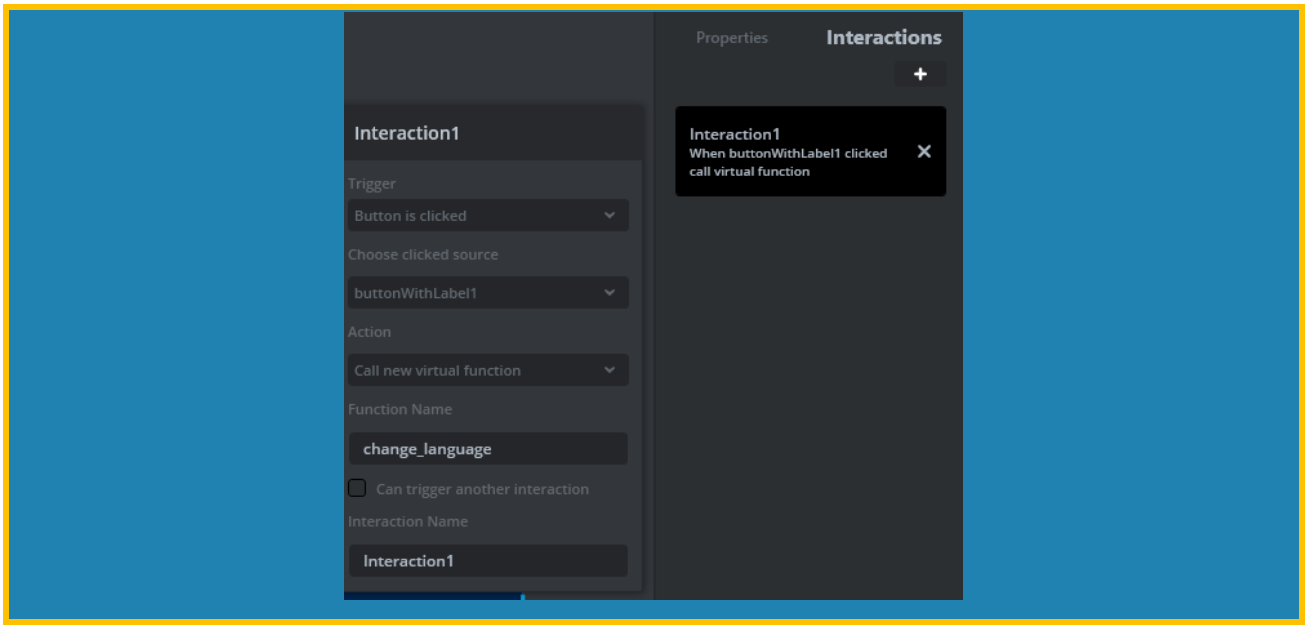
- 在FrontApplication.cpp安装默认的二进制翻译，此处默认使用 LanguageGb.bin 文件:

```
#include <gui/common/FrontendApplication.hpp>
#include <stdio.h>
#include <stdlib.h>
#include <touchgfx/Texts.hpp>
#include <texts/TextKeysAndLanguages.hpp>
#include <touchgfx/Utils.hpp>



//read the file into this array in internal RAM
uint8_t translation[10000];

FrontendApplication::FrontendApplication(Model& m, FrontendHeap& heap)
: FrontendApplicationBase(m, heap)
{
    //read the binary font from a file
```

```
#ifdef __GNUC__
    FILE* font = fopen("generated/texts/binary/LanguageGb.bin", "rb");
#else
    FILE* font = 0;
    fopen_s(&font, "../../generated/texts/binary/LanguageGb.bin", "rb");
#endif

    if (font)
    {
        //touchgfx_printf("open font file ok !!!\n");
        //read data from the file
        fread(translation, 1, 10000, font);
        fclose(font);

        //replace empty translation with the binary data
        Texts::setTranslation(GB, translation);

        //always change language to get TouchGFX changed from the
        //empty translation compiled in to the binary translation
        Texts::setLanguage(GB);
    }
    else{
        //touchgfx_printf("open font file fail !!!\n");
    }
}
```

3.7 添加 Button 交互，用于选择使用不同的二进制翻译

- 添加交互，当按键按下，切换不同语言:



- 添加代码 Screen1View.hpp

```
#ifndef SCREEN1VIEW_HPP
#define SCREEN1VIEW_HPP
```

```
#include <gui_generated/screen1_screen/Screen1ViewBase.hpp>
#include <gui/screen1_screen/Screen1Presenter.hpp>
#include <stdio.h>
#include <stdlib.h>
#include <touchgfx/Texts.hpp>
#include <texts/TextKeysAndLanguages.hpp>
#include <fonts/FontCache.hpp>
#include <fonts/CachedFont.hpp>
#include <texts/TypedTextDatabase.hpp>
#include <fonts/GeneratedFont.hpp>
#include <touchgfx/Utils.hpp>
class Screen1View : public Screen1ViewBase
{
public:
    Screen1View();
    virtual ~Screen1View() {}
    virtual void setupScreen();
    virtual void tearDownScreen();
        virtual void change_language();
        //read the file into this array in internal RAM
        uint8_t translation[10000];
        bool flag = 0;

#ifdef __GNUC__
        FILE* font;
#else
        FILE* font = 0;
#endif
protected:
};

#endif // SCREEN1VIEW_HPP
```

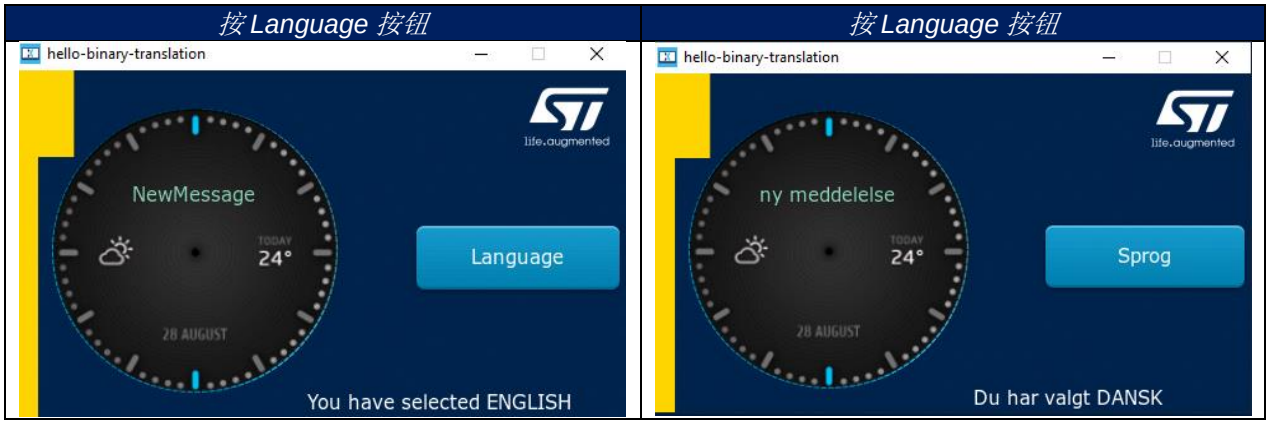
- 添加代码 Screen1View.cpp

```
#include <gui/screen1_screen/Screen1View.hpp>
#include <stdio.h>
#include <stdlib.h>
#include <touchgfx/Texts.hpp>
#include <texts/TextKeysAndLanguages.hpp>
#include <touchgfx/Utils.hpp>
Screen1View::Screen1View()
{

}

void Screen1View::setupScreen()
```

```
{
   Screen1ViewBase::setupScreen();
      invalidate();
}

void Screen1View::tearDownScreen()
{
   Screen1ViewBase::tearDownScreen();
}
void Screen1View::change_language()
{
      if ( flag == 0 )
      {
            flag = 1;
            //read the binary font from a file
            #ifdef __GNUC__
                  font = fopen("generated/texts/binary/LanguageDk.bin", "rb");
            #else
                  fopen_s(&font, "../../generated/texts/binary/LanguageDk.bin", "rb");
            #endif
      }
      else{
            flag=0;
            //read the binary font from a file
            #ifdef __GNUC__
                  font = fopen("generated/texts/binary/LanguageGb.bin", "rb");
            #else
                  fopen_s(&font, "../../generated/texts/binary/LanguageGb.bin", "rb");
            #endif
      }
      if (font)
      {
            //touchgfx_printf("open font file ok !!!\n");
            //read data from the file
            fread(translation, 1, 10000, font);
            fclose(font);

            //replace empty translation with the binary data
            Texts::setTranslation(GB, translation);

            //always change language to get TouchGFX changed from the
            //empty translation compiled in to the binary translation
            Texts::setLanguage(GB);
      }
      else{
            //touchgfx_printf("open font file fail !!!\n");
      }
      invalidate();
```

```
}
```

3.8 编译运行结果如下：



4. 小结

从以上的演示示例可以看到,通过TouchGFX Designer的简单配置和调用提供的API，就可以轻松实现翻译的动态更新。

厂家和用户可以根据场景需求灵活使用本功能。比如，厂家可以根据用户的不同国家来提供OTA服务，让用户动态更新自己的语言翻译类型。这样由于各二进制翻译占用同一块内存区域，厂家的硬件成本也会得到降低。

具体示例实现过程，请参考示例代码：

《hello-tgfx4180-binary-translation-v1.0-202112.7z》

参考文献

文件编号	文件标题	版本号	发布尔日期
Binary Translation	https://support.touchgfx.com/4.13/docs/development/ui-development/touchgfx-engine-features/using-binary-translations		

文档中所用到的工具及版本

TouchGFX Designer 4.18.0

Visual Code 1.63.2

LAT 中的附件

《 hello-tgfx4180-binary-translation-v1.0-202112.7z》

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。