AN13730

如何使用GUI Guider在内存受限的MCU上开发LVGL GUI示例

第0版 — 2022年9月9日 应用笔记

文档信息

信息	内容
关键词	GUI Guider 1.3.1、LVGL、LPC55S06
摘要	本应用笔记介绍了用于支持外部SPI Flash的LVGL文件系统机制的使用，以及用于支持硬件按钮进行界面切换的LVGL输入设备机制的使用。

1 介绍

一个有吸引力的GUI依赖于精心设计的图像和字体。GUI示例越复杂，需要的图像和字体资源就越多，从而导致消耗更多的内存资源。如果某个设计中所选的MCU没有足够的片内Flash和片内RAM来存储图像和字体，那么就不得不使用片外Flash和片外RAM。

幸运的是，LVGL提供了文件系统机制来支持外部存储设备，如SD卡或串行Flash等。本应用笔记中使用**LPC55S06**作为目标MCU，以电动自行车用户界面（**E-Bike** UI）的实现为例，介绍了如何使用LVGL文件系统来支持低成本的外部串行Flash。本应用笔记中使用的外部串行Flash是Winbond **W25Q64**。

除了提供图形功能之外，LVGL还支持一种输入设备机制。本应用笔记介绍了如何使用硬件按钮作为LVGL输入设备来实现界面切换。

2 LPC55S06概述

LPC55S0x/LPC550x是基于Arm Cortex-M33内核的高性价比微控制器系列，适用于嵌入式应用，具有以下特性：

- 运行频率高达**96 MHz**
- 用于隔离安全和非安全代码的TrustZone选项
- 浮点单元（FPU）和内存保护单元（MPU）
- 多达**96 kB的片上RAM**
- 多达**256 kB的片上Flash**
- CAN-FD
- 五个通用定时器
- SC定时器/PWM
- RTC/报警定时器
- 24位多速率定时器（MRT）
- 窗口看门狗定时器（WWDT）
- 代码看门狗
- **高速SPI（50 MHz）**
- 八个灵活的串行通信外设（每个都可以支持USART、SPI、I2C或I2S接口）
- 16位每秒2.0 M采样ADC，能够同步转换
- 温度传感器。

上面列出的与显示性能密切相关的MCU的特性包括：系统频率、Flash容量、RAM容量和SPI通信速率。本示例使用高速SPI连接到外部串行Flash。

3 LVGL概述

LVGL是一个开源图形库，提供了创建嵌入式GUI所需的所有功能，带有易于使用的图形元素、美观的视觉效果和低内存占用。

关键特性：

- 开源且在MIT许可证下免费使用
- 用C（兼容C++）语言编写，并装载在GitHub上
- 超过30个功能强大、完全可定制的小部件，如按钮、图像按钮、复选框、开关、滑块、标签、弧形、条形、线条、画布、图像、滚轮、滑条、仪表盘、表格、文本区域、动画、日历、图表、列表、菜单、消息框、选项卡视图等
- 支持任意分辨率的显示，提供GPU支持、多显示器支持
- 支持多种类型的输入设备，如：
  - 指针类输入设备，如触摸板或鼠标
  - 键盘式输入设备，如普通键盘或简单数字键盘
  - 具有左/右旋转和按压功能的编码器
  - 分配给界面上的特定点的外部硬件按钮
- 绘图功能，如：
  - 抗锯齿
  - 阴影
  - 直线、圆弧、多边形
  - 遮罩
- 文本功能，如：
  - 支持UTF-8
  - 字距调整
  - 自动换行和自动文本滚动
  - 支持阿拉伯语和波斯语
  - 字体压缩
  - 亚像素渲染
  - 在线和离线字体转换器
  - 自定义字体引擎接口
  - FreeType集成示例
  - 提供多语言支持
- 图像功能，如：
  - 各种颜色格式：RGB、ARGB、色度键控、索引、仅alpha
  - 实时图像重新着色
  - 实时缩放和旋转
  - 图像可以存储在Flash或文件中（如SD卡）
  - 在线和离线图像转换器
  - 用于缓存的图像解码器接口
  - PNG集成示例
- 样式，如：
  - 级联样式（如CSS）
  - 在多个窗口组件中重用样式
  - 局部样式，用于简单变化
  - 主题，用于提供默认外观
  - 状态更改时的转换（动画）
- 支持Micropython
- 丰富的演示示例和文档
- 由恩智浦开发的免费UI设计工具GUI Guider提供支持。

详细信息请参阅LVGL页面。

本应用笔记中使用的LVGL版本为**8.0.2**。

4 GUI Guider概述

GUI Guider是恩智浦推出的一款用户友好的图形用户界面开发工具，可利用开源的LVGL图形库快速开发高质量的显示界面。GUI Guider的拖放式编辑器可以轻松利用LVGL的许多功能（如小部件、动画和样式），只需最少的编码或完全无需编码即可创建一个GUI。

只需点击一个按钮，就可以在模拟环境中运行应用程序，或将其导出到目标工程。从GUI Guider生成的代码可以轻松地添加到MCUXpresso IDE、IAR Embedded Workbench或Keil uVision工程中。GUI Guider加速了开发过程，让用户可以无缝地将嵌入式用户界面添加到应用程序中。

GUI Guider可以免费用于恩智浦的通用MCU和跨界MCU。它包含了几个内置工程模板，适用于多种受支持的MCU平台。

更多细节，请参考GUI Guider。

本应用笔记使用的GUI Guider版本是**1.3.1**。

5 电动自行车（E-bike）示例概述

电动自行车（E-Bike）示例是一个GUI 应用，具有3个界面，分别命名为Overview、Ride 1和Ride 2，如图2、图3和图4所示。在这些界面的底部，有三个带有 **<**、**>** 和 **^**标签的按钮，分别用于将当前界面切换到上一个界面、下一个界面或主界面。**Overview**是系统重启后显示的第一个界面，因此在后文中我们称之为主界面。

图1所示为硬件平台，专门为电动自行车示例定制。

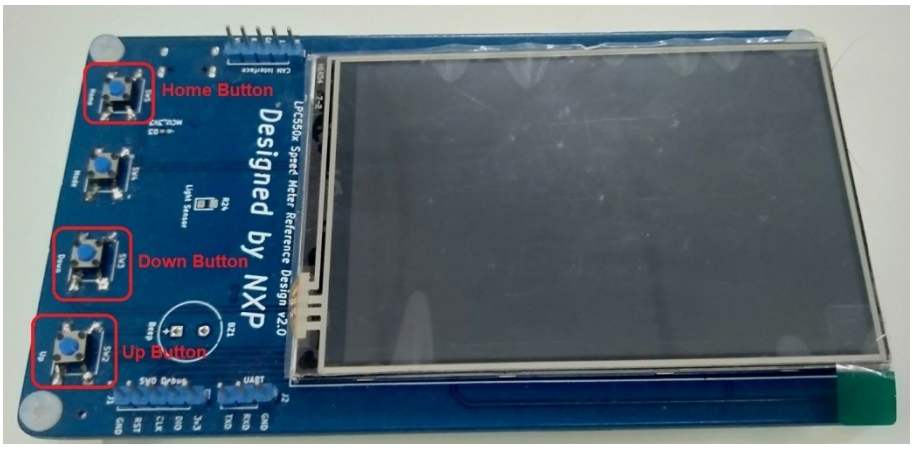


图1. 硬件平台

“主界面（Home）”按钮用于从另一个活动界面切换回主界面。“向下（Down）”按钮用于切换到下一个界面，“向上（Up）”按钮用于切换到上一个界面。

例如，假设当前活动界面是 Ride 1，

- 如果按下“向下（Down）”按钮，则会切换到 Ride 2 界面。
- 如果按下“主界面（Home）”按钮或“向上（Up）”按钮，则会切换到主界面。

图2. Overview（主）界面

图3. Ride 1界面

图4. Ride 2界面

要了解如何使用GUI Guider进行GUI设计，包括创建工程、添加小部件、设置小部件属性、添加事件、模拟、生成代码以及下载到目标板并运行等步骤，请参阅*Smart Home Demo on GUI Guider for LPC546xx*（文档AN13694）。

6 外部串行Flash支持

如上所述，LPC55S06上的片上Flash为256 kB，不足以存储本示例中使用的图像资源。接下来将逐步介绍如何设置本设计以使用外部Flash。

1. 针对与高速SPI接口相关的复用功能配置引脚，如表1和图5所示。

表1. 连接到外部串行Flash的IO的复用功能

SPI	引脚	复用功能
MOSI	PIO0_26	9
SSEL1	PIO1_1	5
SCK	PIO1_2	6
MISO	PIO1_3	6

2. 使用**HS_SPI_Init**函数初始化高速SPI接口，如图5所示。

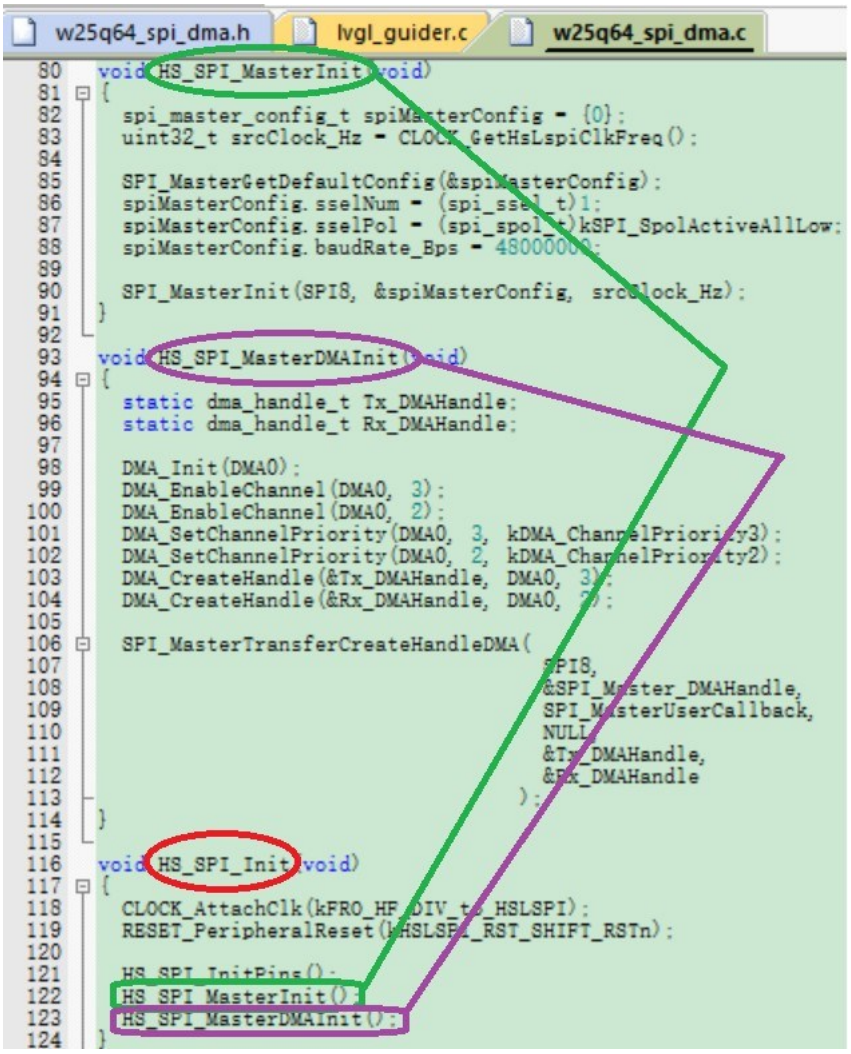


图5. 高速SPI的初始化

3. 将用于驱动外部串行Flash的源文件和头文件添加到生成的代码工程中。在此示例中，假设已经使用GUI Guider完成了GUI页面设计，并基于Keil IDE生成了一个代码工程。图6所示为代码工程文件夹中存储的驱动程序文件的目录。图7所示为Keil项目中的驱动程序文件所在的组。

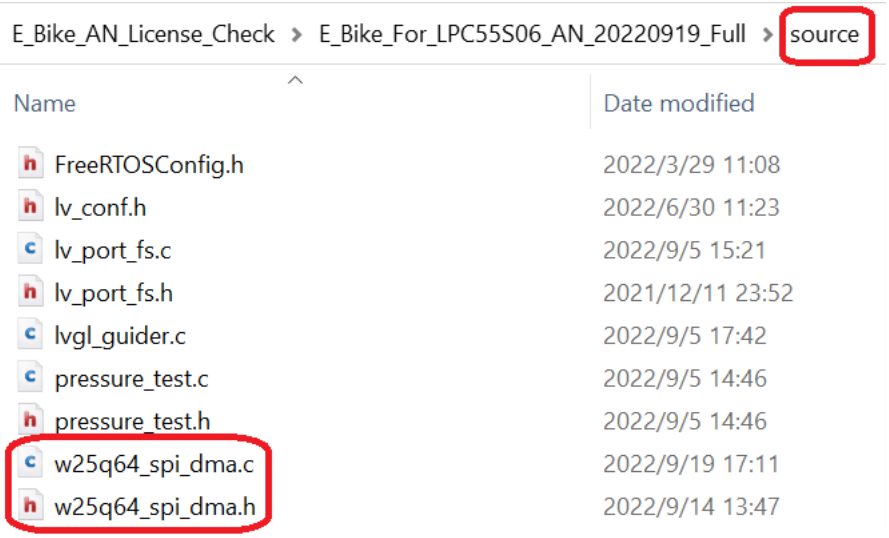


图6. 外部串行Flash驱动程序的目录

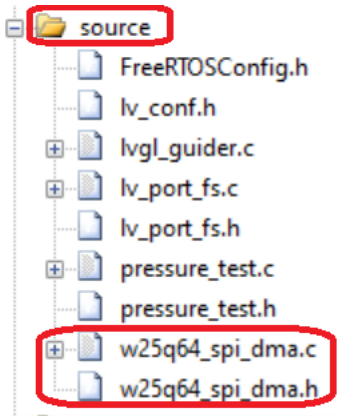


图7. 外部串行Flash驱动程序所在的Keil组

4. 将用于驱动外部串行Flash的初始化函数的调用添加到main()函数中，如图8所示。

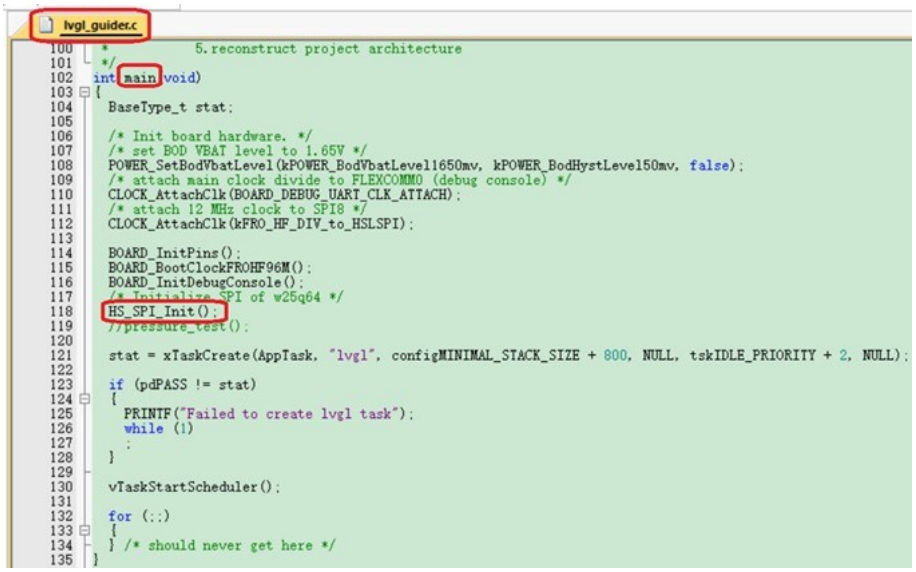


图8. 调用初始化函数到main()函数中

7 图像资源的外部存储

该示例使用独立的外部操作将图像加载到Flash中。要将本示例中使用的图像资源下载到外部串行Flash，请准备图像文件，然后使用调试探头（SEGGER J-Link）完成下载。

1. 使用在线图像转换器将 BMP、JPG 或 PNG 图像转换为二进制格式。图像转换器网址：Online image converter - BMP, JPG or PNG to C array or binary。

图9. LVGL在线图像转换器的网址

转换操作的一个关键点是为图像转换器选择正确的颜色格式和输出格式。GUI Guider 1.3.1 版本支持两种颜色格式，如图10所示。此处，该演示选择**True Color Alpha**。

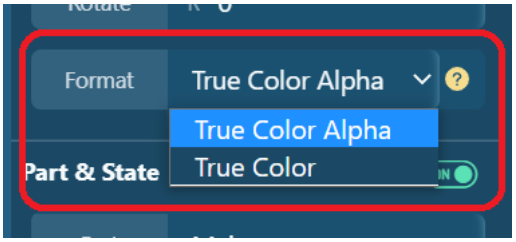


图10. GUI Guider支持的颜色格式

关于输出格式，请参阅代码工程中名为 **lv_conf.h** 的配置文件。此处，该示例选择 **16-bit color depth without swap**。

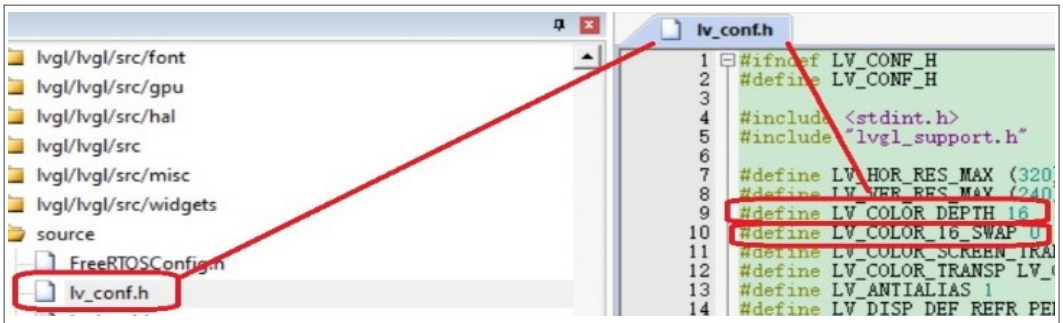


图11. GUI Guider支持的color depth

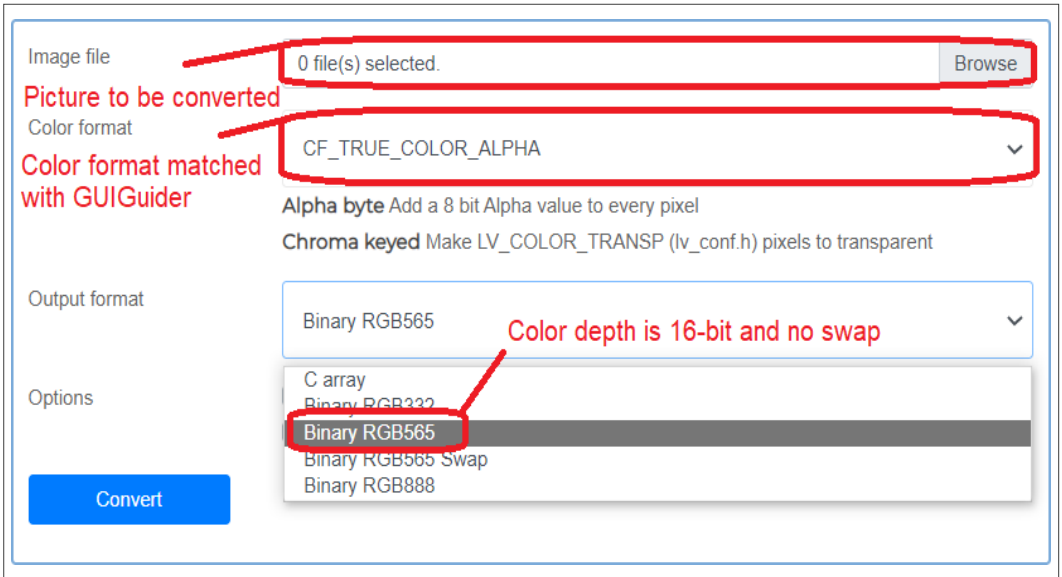


图12. 图像转换器的颜色格式和输出格式配置

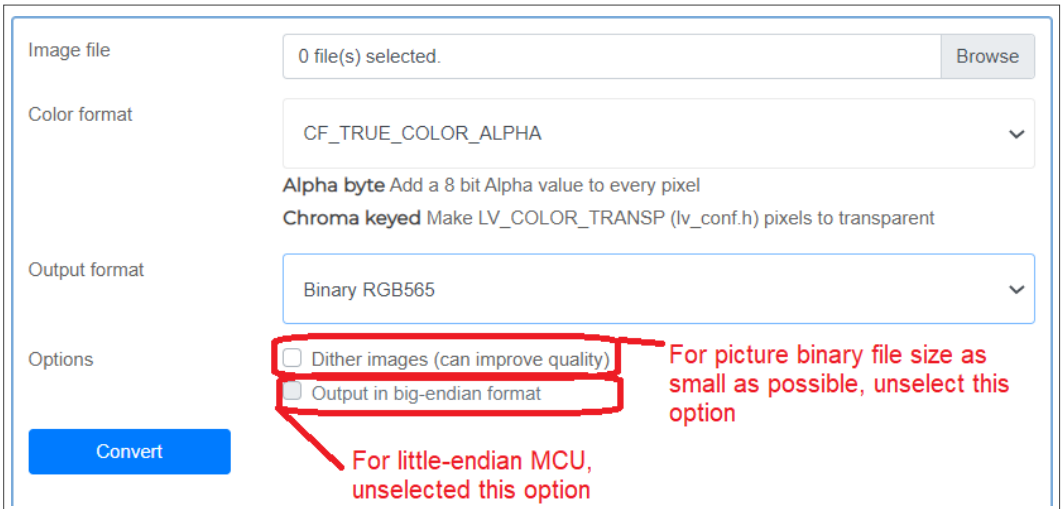


图13. 图像转换器的抖动和字节序配置

1. 使用名为 **MultipleBinFileMergeTool.cpp** 的二进制合并工具，将步骤1中生成的二进制图像文件合并为名为 **mergeBinFile.bin** 的单个二进制文件，如图14所示。

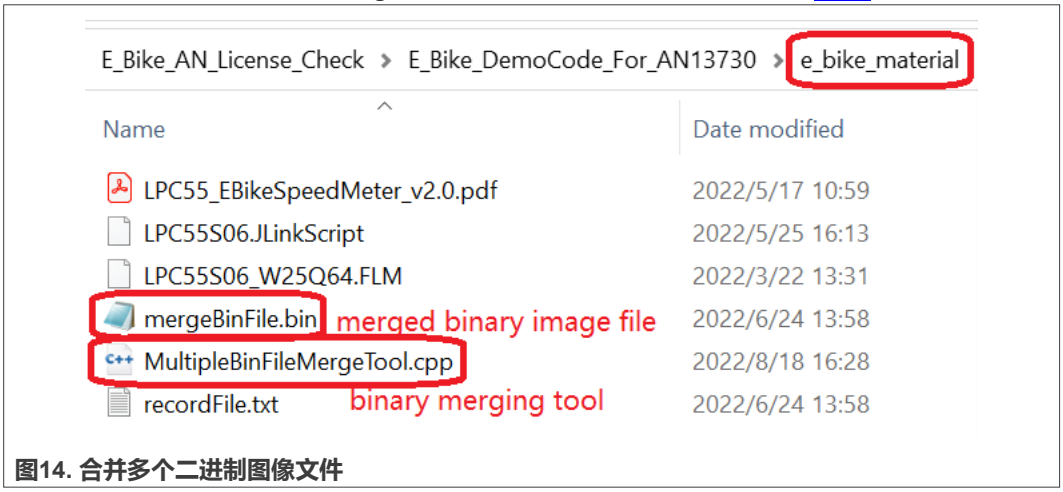


图14. 合并多个二进制图像文件

2. 将步骤2中生成的合并后的二进制图像文件下载到外部串行Flash中。要实现此操作，需要为J-Link创建一个Flash驱动程序和用于执行编程操作的J-Flash实用程序，具体步骤如下所述。
   a. 要使用J-Link探针对设计中的Flash存储器进行编程，需要一个驱动程序或算法文件（称为FLM文件）。将外部串行Flash的编程算法文件放置到SEGGER驱动程序安装目录中的针对恩智浦器件的特定文件目录下，如图15所示。

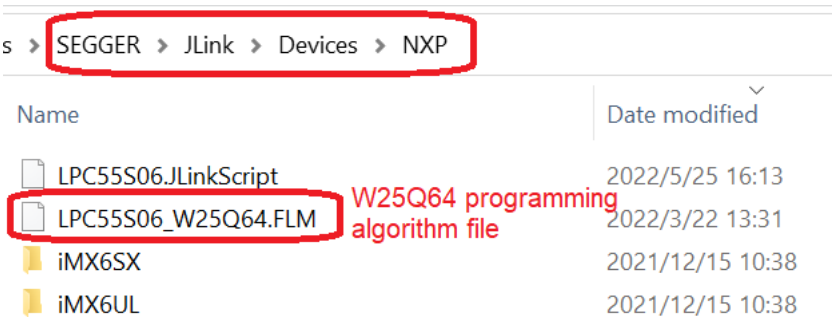


图15. 外部串行Flash的编程算法文件目录

b. 在J-Link安装目录中找到*JLinkDevices.xml*。例如，*JLinkDevices.xml* 放置在图16所示的目录中。J-Link驱动程序使用此文件来识别所有受支持的闪存设备并查找其关联的驱动程序。

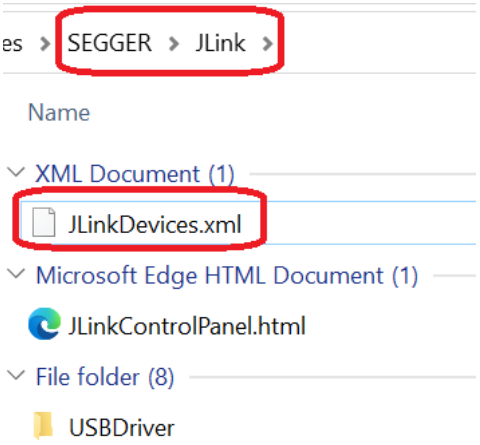


图16.JLinkDevices.xml所在目录

在*JLinkDevices.xml*文件的末尾，添加外部串行Flash的算法索引条目，如图17所示。索引条目如下：

```
<Device>
 <ChipInfo Vendor="NXP" Name="LPC55S06_SPIFlash_W25Q64"
 WorkRAMAddr="0x20000000" WorkRAMSize="0x8000"
 Core="JLINK_CORE_CORTEX_M33" />
 <!-- MCU does not have memory mapped flash area, instead
 a virtuell address is used. -->
 <FlashBankInfo Name="EXTSPI" BaseAddr="0xC0000000"
 MaxSize="0x400000" Loader="Devices/NXP/
LPC55S06_W25Q64.FLM"
 LoaderType="FLASH_ALGO_TYPE_CMSIS" />
<Device>
```

图17. 外部Flash的编程算法索引位置

注意：Loader="Devices/NXP/LPC55S06_W25Q64.FLM"表示Flash编程算法文件的文件路径。用户可以根据自己的需要修改文件名和文件路径，但要确保它们与实际的文件名和文件目录保持一致。

c. 启动该目录中的*J-Flash.exe*，如图18所示。启动J-Flash后，主界面如图19所示。

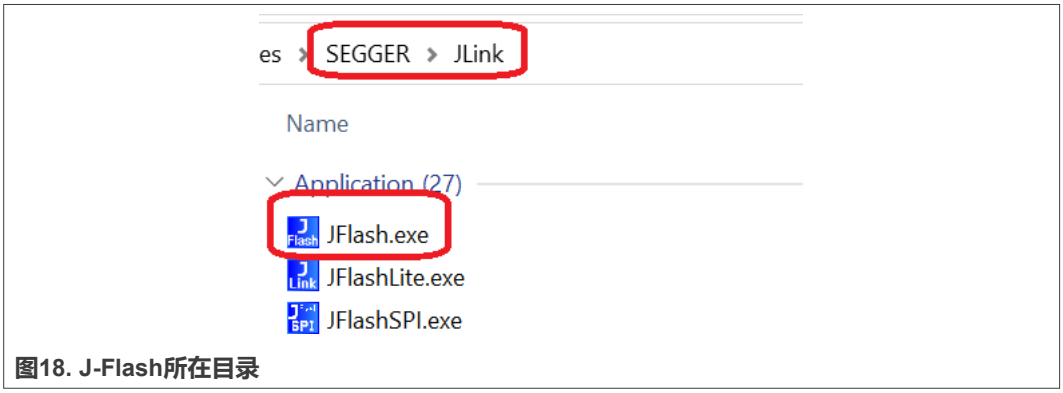


图18. J-Flash所在目录

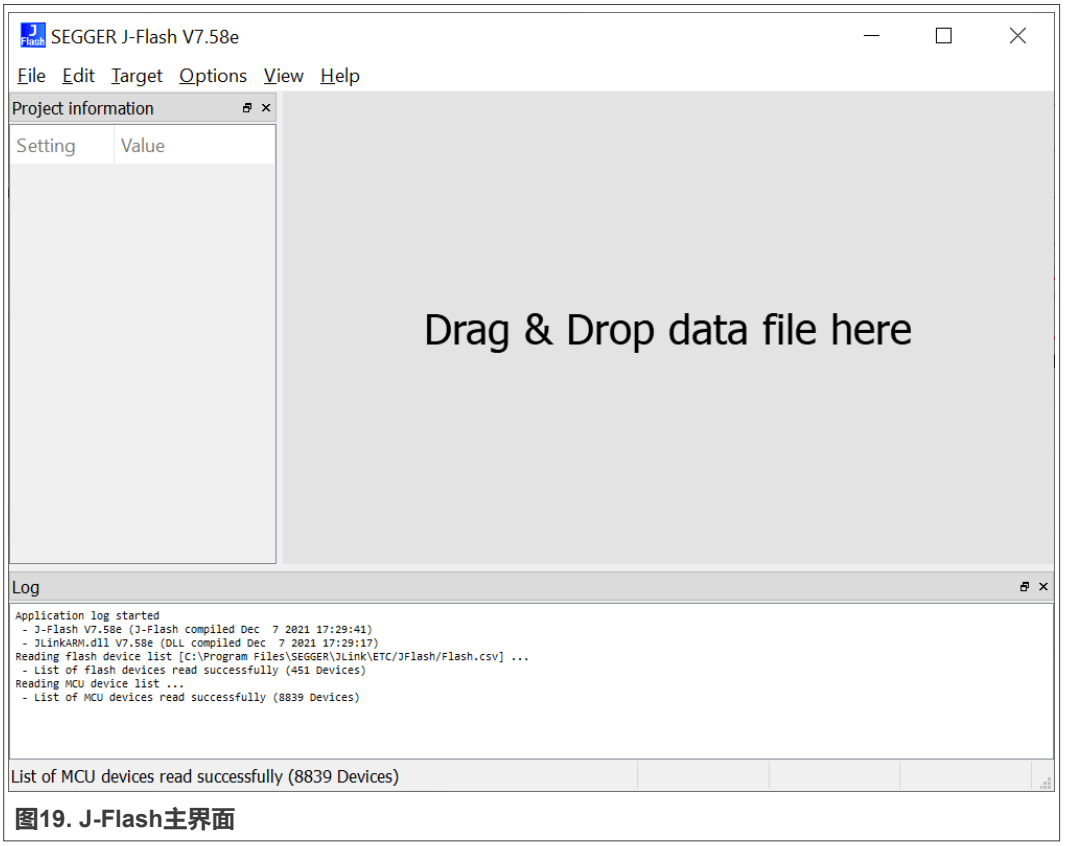


图19. J-Flash主界面

d. 点击 **File -> New project**，弹出**Create New Project（创建新项目）**对话框，如图20所示。根据实际情况选择**Target interface（目标接口）**和**Speed（速度）**。这里我们选择**SWD**和**4000 kHz**。

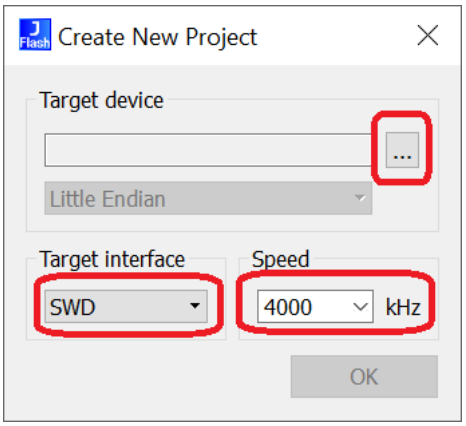


图20. 创建新项目对话框

e. 点击“**...**”按钮，弹出**Target Device Settings（目标器件设置）**对话框，如图21所示。由于目标器件是LPC55S06，因此使用**LPC55S06**作为关键词来搜索所需的目标器件。

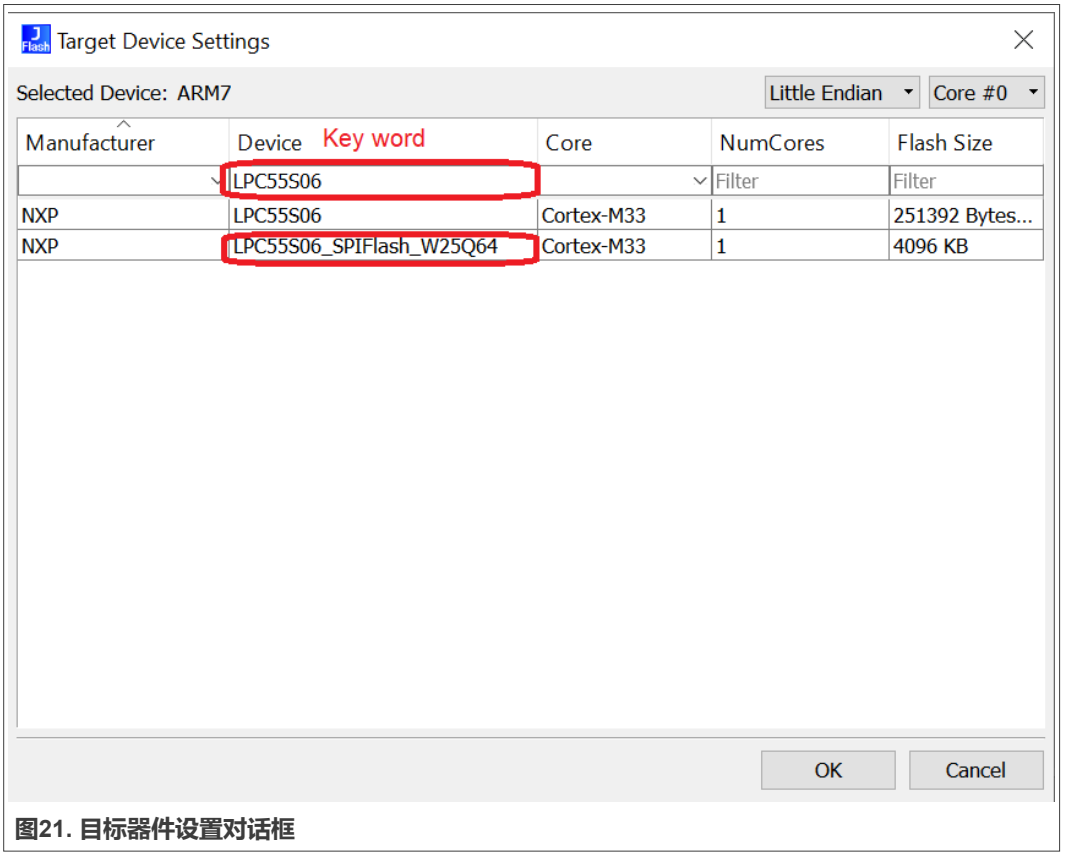


图21. 目标器件设置对话框

选择**LPC55S06_SPIFlash_W25Q64**，它就是步骤b中在*JLinkDevices.xml*文件中添加的闪存编程算法索引。

f. 在图21中，点击**OK**返回到图20。在图20中，点击**OK**完成项目创建，如图22所示。

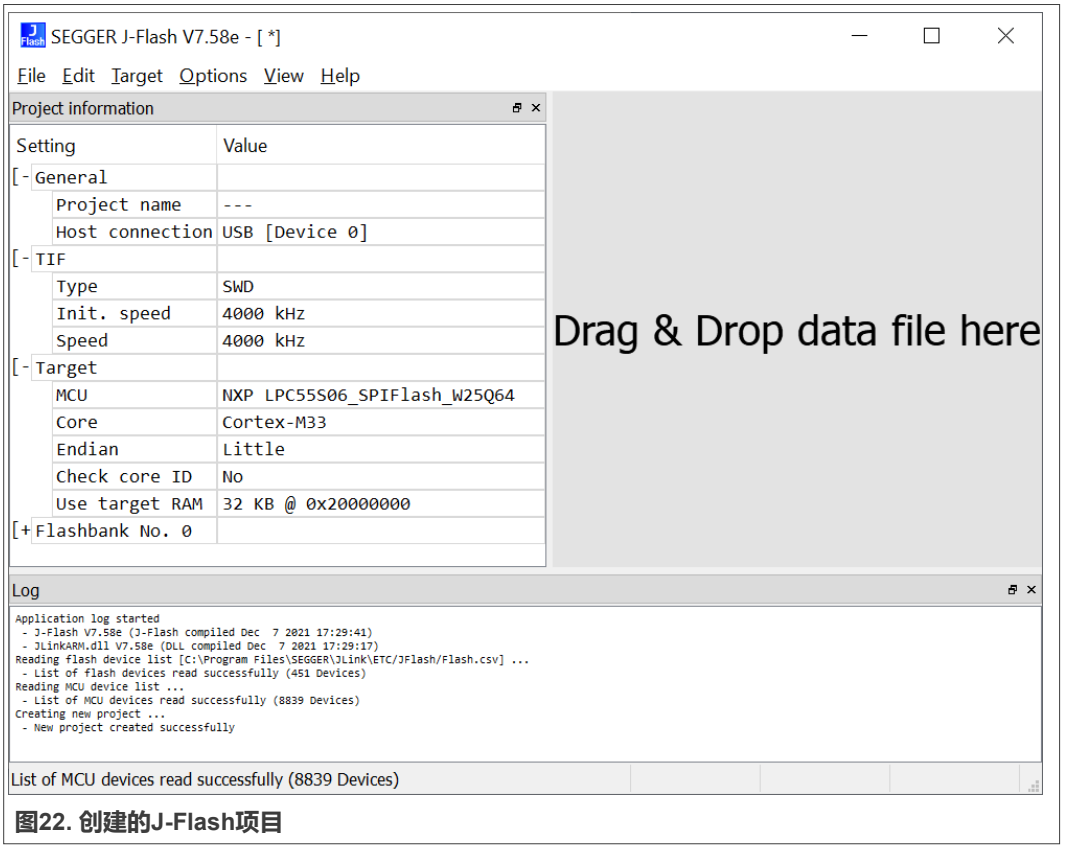


图22. 创建的J-Flash项目

g. 点击**Options -> Project settings**，弹出**Project settings（项目设置）**对话框，如图23所示。点击**MCU**，启用**Use J-Link script file（使用J-Link脚本文件）**，并选择LPC55S06的脚本文件。该示例中使用的脚本文件是**LPC55S06.JLinkScript**，其内容如图24所示。

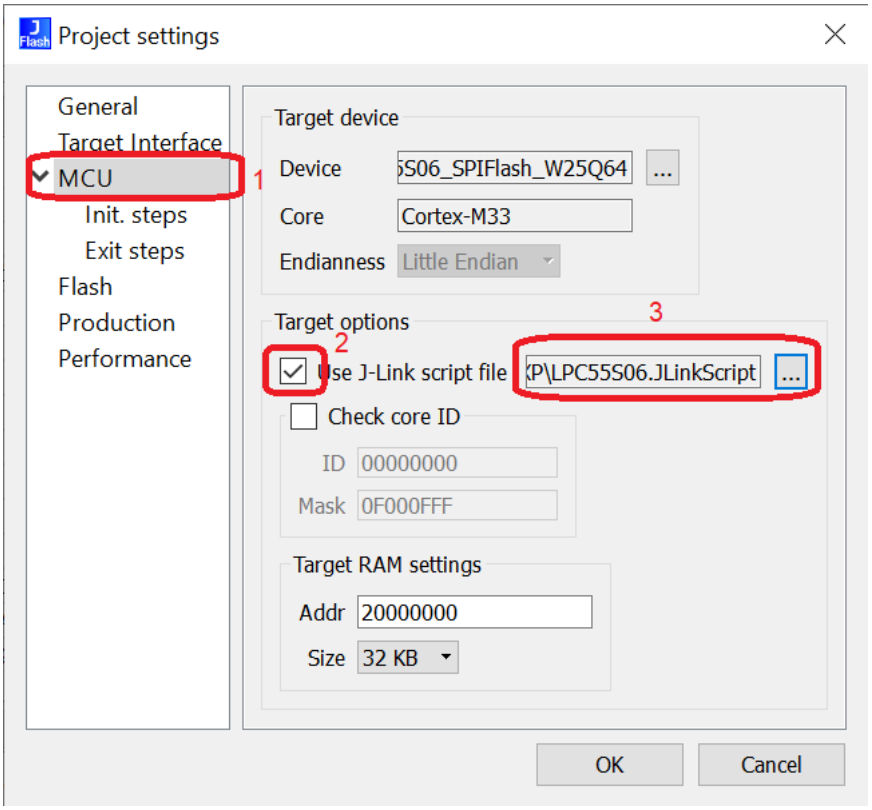


图23. 使用脚本配置J-Flash项目

LPC55S06.JLinkScript

```
*
*       ResetTarget
* This device requires a special reset as default reset does not work for blank device.
*/
// Overwrite ResetTarget for blank device download only.
void ResetTarget(void) {}


/*********************************************************************
*
*       InitTarget
*/
void InitTarget(void) {
  int v;

  JLINK_SYS_Report("******************************************************");
  JLINK_SYS_Report("J-Link script: LPC55xx Cortex-M33 core J-Link script");
  JLINK_SYS_Report("******************************************************");
  JLINK_CORESIGHT_Configure("IRPre=0;DRPre=0;IRPost=0;DRPost=0;IRLenDevice=4");
  CPU = CORTEX_M33;      // Pre-select that we have a Cortex-M33 connected
  JTAG_AllowTAPReset = 0;   // J-Link is allowed to use a TAP reset for JTAG-chain auto-detection

  JTAG_SetDeviceId(0, 0x6BA02477);  // 4-bits IRLen

  // Select ISP-AP
  JLINK_CORESIGHT_WriteDP(2, 0x020000f0);
  v = JLINK_CORESIGHT_ReadAP(3);
  JLINK_SYS_Report1("DAP-IDCODE:", v);
  JLINK_CORESIGHT_WriteDP(2, 0x02000000);
  JLINK_CORESIGHT_ReadDP(0);

  // Active DebugMailbox
  JLINK_CORESIGHT_WriteAP(0, 0x21);
  JLINK_CORESIGHT_ReadAP(0);

  // Enter Debug Session
  JLINK_CORESIGHT_WriteAP(1, 0x07);
  JLINK_CORESIGHT_ReadAP(0);
}
```

图24. J-Flash项目设置的脚本文件

h. 要保存创建的工程，请点击**File** -> **Save project as…**。

i. 要将合并后的二进制图像文件**mergeBinFile.bin**加载到J-Flash中，请点击**File** -> **Open data file…**。

j. 要在PC上的J-Flash工具和电动自行车（E-Bike）硬件平台上的调试器之间建立连接，请点击**Target** -> **Connect**。

k. 要擦除整个外部串行Flash，请点击**Target** -> **Manual Programming**->**Erase Chip**。

l. 要将合并后的二进制图像文件**mergeBinFile.bin**烧录到电动自行车硬件平台上的外部串行Flash中，请点击**Target** -> **Manual Programming** -> **Program**。

8 SRAM3支持

当开发GUI应用程序时，可能会遇到如图25所示的错误。

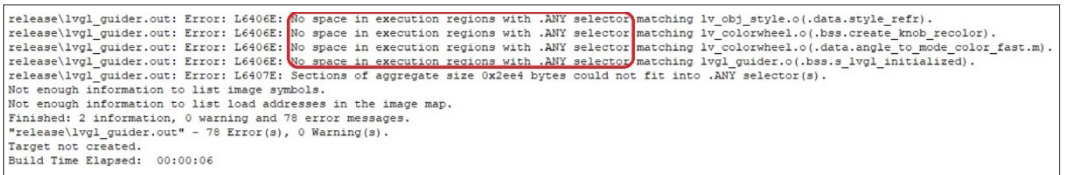


图25. 无空间错误

要解决此错误，首先要了解错误的根本原因。无空间错误的根本原因是Flash和/或RAM的容量不足以存储GUI应用程序。导致Flash和/或RAM容量不足的原因有两个：

- 第一个是MCU的内存资源没有得到充分利用。
- 第二个是虽然MCU的内存资源已经被充分利用，但由于所选MCU的存储容量较小，仍然不足以存储整个GUI应用程序。

要判断一个GUI应用程序是否充分利用了内存资源，可以检查其描述内存资源分配的文件，例如Keil IDE的分散加载（或链接器）文件。本文档以**LPC55S06**为例，说明如何判断内存资源是否被充分利用。

要获取LPC55S06的内存资源，请参见*LPC55S0x/LPC550x用户手册*（文档UM11424），如图26所示。LPC55S06共有256 kB的**片上Flash**，其中系统保留**12 kB**，用户应用程序使用剩余的**244 kB**。LPC55S06共有96 kB的**片上RAM**，包括**16 kB的SRAMX、32 kB的SRAM 0、16 kB的SRAM 1、16 kB的SRAM 2和16 kB的SRAM 3**。

要使用GUI Guider创建一个GUI应用程序，需要生成一个基于Keil IDE的代码工程，然后打开分散加载文件，查看内存分配情况，如图27所示。对于RAM，分散加载文件指定了从`0x20000000`到`0x2000FFFF`的RAM地址空间作为数据段，不包括SRAM3（0x20010000-0x20013FFF）。**RAM地址空间未被充分利用。**至于Flash，所有的244 kB片上Flash都用于代码段，其大小为`0x0003CE00`加上`0x00000200`后的值。**因此，Flash地址空间被充分利用。**

由于我们发现RAM没有被充分利用，我们可以将SRAM 3和SRAM 0、1、2一起启用，把它们用作数据段。可用的RAM空间则从64 kB增加到80 kB。关于如何启用SRAM 3，请参见*SRAM3 Usage in LPC55(s)06*（文档AN13628）。

Table 4. Memory map overview

AHB port	Non-secure start address	Non-secure end address	Secure start address	Secure end address	Function [1]
0	0x0000 0000	0x0003 FFFF	0x1000 0000	0x1003 FFFF	Flash memory, on CM33 code bus. The last 17 pages (12KB) are reserved on the 256 KB flash devices resulting in 244 KB internal flash memory.
	0x0300 0000	0x0301 FFFF	0x1300 0000	0x1301 FFFF	Boot ROM, on CM33 code bus.
1	0x0400 0000	0x0400 3FFF	0x1400 0000	0x1400 3FFF	SRAM X on CM33 code bus, 32 KB. SRAMX_0 (0x1400 0000 to 0x1400 0FFF) and SRAMX_1 (0x1400 1000 to 0x1400 1FFF) are used for Casper (total 8 KB). If CPU retention used in power-down mode, SRAMX_2 (0x1400 2000 to 0x1400 25FF) is used (total 1.5 KB) by default in power API and this is user configurable within SRAMX_2 and SRAMX_3.
2	0x2000 0000	0x2000 7FFF	0x3000 0000	0x3000 7FFF	SRAM 0 on CM33 data bus, 32 KB.
3	0x2000 8000	0x2000 BFFF	0x3000 8000	0x3000 BFFF	SRAM 1 on CM33 data bus, 16 KB.
4	0x2000 C000	0x2000 FFFF	0x3000 C000	0x3000 FFFF	SRAM 2 on CM33 data bus, 16 KB.
5	0x2001 0000	0x2001 3FFF	0x3001 0000	0x3001 3FFF	SRAM 3, 16 KB.
6	0x4000 0000	0x4001 FFFF	0x5000 0000	0x5001 FFFF	AHB to APB bridge 0. See Section 2.1.6.
	0x4002 0000	0x4003 FFFF	0x5002 0000	0x5003 FFFF	AHB to APB bridge 1. See Section 2.1.6.
7	0x4008 0000	0x4008 FFFF	0x5008 0000	0x5008 FFFF	AHB peripherals. See Section 2.1.7.
8	0x4009 0000	0x4009 FFFF	0x5009 0000	0x5009 FFFF	AHB peripherals. See Section 2.1.7.
9	0x400A 0000	0x400A FFFF	0x500A 0000	0x500A FFFF	AHB peripherals. See Section 2.1.7.

244KB is available for user code

16KB

96KB RAM is available for user code

图26. LPC55S06内存资源

LPC55S06_flash.scf

```
#endif

#define  m_interrupts_start             0x00000000
#define  m_interrupts_size              0x00000200

#define m_text_start                    0x00000200
#define m_text_size                     0x0003CE00

#define  m_data_start                   0x20000000
#define  m_data_size                    0x00010000

#define  m_sramx_start                  0x04000000
#define  m_sramx_size                   0x00004000

#define  m_sram3_start                  0x20010000
#define  m_sram3_size                   0x00004000

LR_m_text m_interrupts_start m_interrupts_size+m_text_size {    ; load region size_region

  VECTOR_ROM m_interrupts_start m_interrupts_size { ; load address = execution address
    * (.isr_vector,+FIRST)
  }

  ER_m_text m_text_start FIXED m_text_size { ; load address = execution address
    * (InRoot$$Sections)
    .ANY (+RO)
  }

  RW_m_data m_data_start m_data_size-Stack_Size-Heap_Size { ; RW data
    .ANY (+RW +ZI)
  }
  ARM_LIB_HEAP +0 EMPTY Heap_Size {     ; Heap region growing up
  }
  ARM_LIB_STACK m_data_start+m_data_size EMPTY -Stack_Size { ; Stack region growing down
  }

}
```

图27. LPC55S06分散加载文件

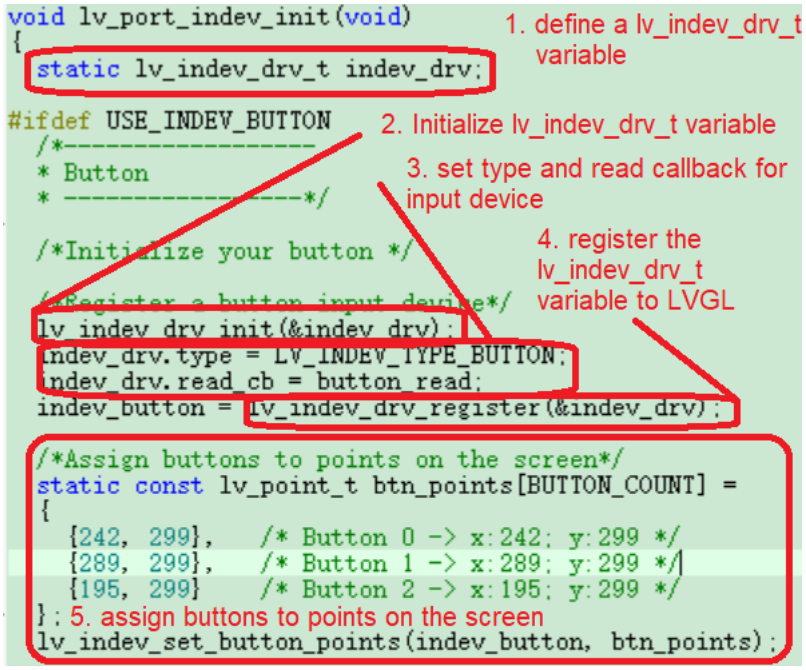
9 用于界面切换的硬件按钮

本节介绍如何使用硬件按钮实现界面切换。LVGL支持以下类型的输入设备：

- 指针类输入设备，如触摸板或鼠标
- 键盘式输入设备，如普通键盘或简单数字键盘
- 具有左/右旋转和按压功能的编码器
- 分配给界面上特定点的外部硬件按钮

要使用硬件按钮实现界面切换，请按照以下步骤操作：

1. 要注册一个输入设备，请初始化**lv_indev_drv_t**变量，如图28所示。




```
void lv_port_indev_init(void)
{
  static lv_indev_drv_t indev_drv;

#ifdef USE_INDEV_BUTTON
  /*------------------
  * Button
  * -----------------*/

  /*Initialize your button */

  /*Register a button input device*/
  lv_indev_drv_init(&indev_drv);
  indev_drv.type = LV_INDEV_TYPE_BUTTON;
  indev_drv.read_cb = button_read;
  indev_button = lv_indev_drv_register(&indev_drv);

  /*Assign buttons to points on the screen*/
  static const lv_point_t btn_points[BUTTON_COUNT] =
  {
    {242, 299},    /* Button 0 -> x:242; y:299 */
    {289, 299},    /* Button 1 -> x:289; y:299 */
    {195, 299}     /* Button 2 -> x:195; y:299 */
  };
  lv_indev_set_button_points(indev_button, btn_points);
```

图28. 注册一个输入设备

2. 实现与按钮读取相关的函数，包括`button_read`，`button_get_pressed_id`和`button_is_pressed`，如图29，图31和图32所示。

```
/* Will be called by the library to read the button */
static void button_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data)
{
  static uint8_t last_btn = 0;

  /* Get the pressed button's ID */
  int8_t btn_act = button_get_pressed_id();

  if(btn_act >= 0)
  {
    data->state = LV_INDEV_STATE_PR;
    last_btn = btn_act;
  }
  else
  {
    data->state = LV_INDEV_STATE_REL;
  }

  /* Save the last pressed button's ID */
  data->btn_id = last_btn;
}
```

图29. `button_read`的实现

```
/* Button counts */
#define BUTTON_COUNT            3
```

图30. 设置按钮数量

```
/* Get ID  (0, 1, 2 ..) of the pressed button */
static int8_t button_get_pressed_id(void)
{
  uint8_t i;

  /*
     Check to buttons see which is being
     pressed (assume there are 3 buttons)
  */
  for(i = 0; i < BUTTON_COUNT; i++)
  {
    /* Return the pressed button's ID */
    if(button_is_pressed(i))
    {
      return i;
    }
  }

  /* No button pressed */
  return -1;
}
```

图31. button_get_pressed_id的实现

```
/*Test if `id` button is pressed or not*/
static bool button_is_pressed(uint8_t id)
{
  /*Your code comes here*/
  switch(id)
  {
    case 0:
    {
      if(GPIO_PinRead(GPIO, 1, 23) == 0)
      {/* Home */
        return true;
      }
    }
    break;
    case 1:
    {
      if(GPIO_PinRead(GPIO, 1, 5) == 0)
      {/* Down */
        return true;
      }
    }
    break;
    case 2:
    {
      if(GPIO_PinRead(GPIO, 1, 21) == 0)
      {/* Up */
        return true;
      }
    }
    break;
    default:
    {
    }
    break;
  }

  return false;
}
```

图32. button_is_pressed的实现

更多关于输入设备的信息，请参阅input device in LVGL。

10 外部串行Flash的文件系统支持

LVGL有一个**文件系统**抽象模块，能够附加任何类型的文件系统。在此示例中，我们为外部串行Flash构建了一个简单的文件系统，使其可以通过文件API函数操作存储在外部串行Flash上的图像文件。关于LVGL中的文件系统的更多详细信息，请参见 File System in LVGL。

10.1 获取文件系统模板文件并添加到代码工程中

为了快速移植文件系统，LVGL提供了一个文件系统模板文件。首先，克隆位于GitHub上的LVGL图形库。LVGL图形库的GitHub链接请点击此处。

LVGL文件系统模板文件为*lv_port_fs_template.c*，对应的头文件为*lv_port_fs_template.h*。这两个文件的目录如图33所示。

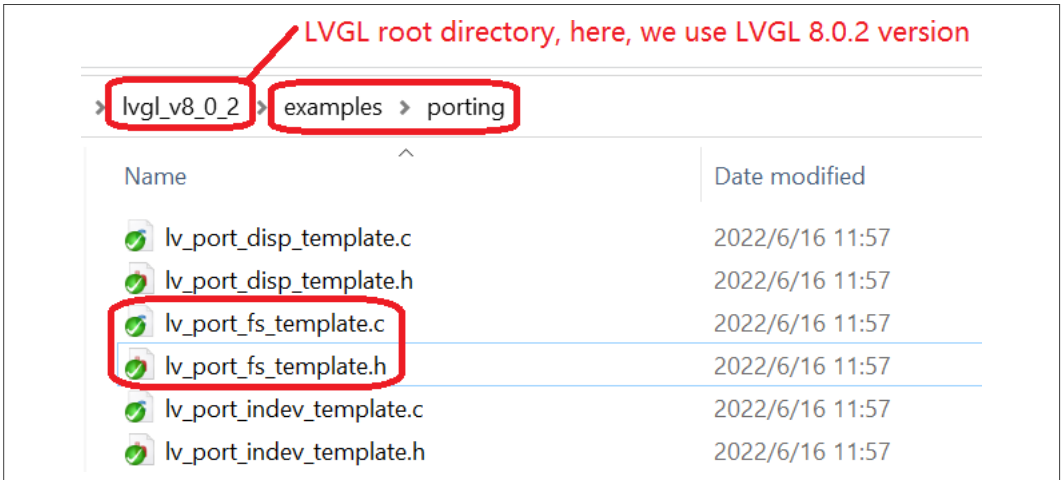


图33. 文件系统模板文件目录

将这两个文件复制到代码工程目录下，并将它们重命名为`lv_port_fs.c`和`lv_port_fs.h`，如图34所示。

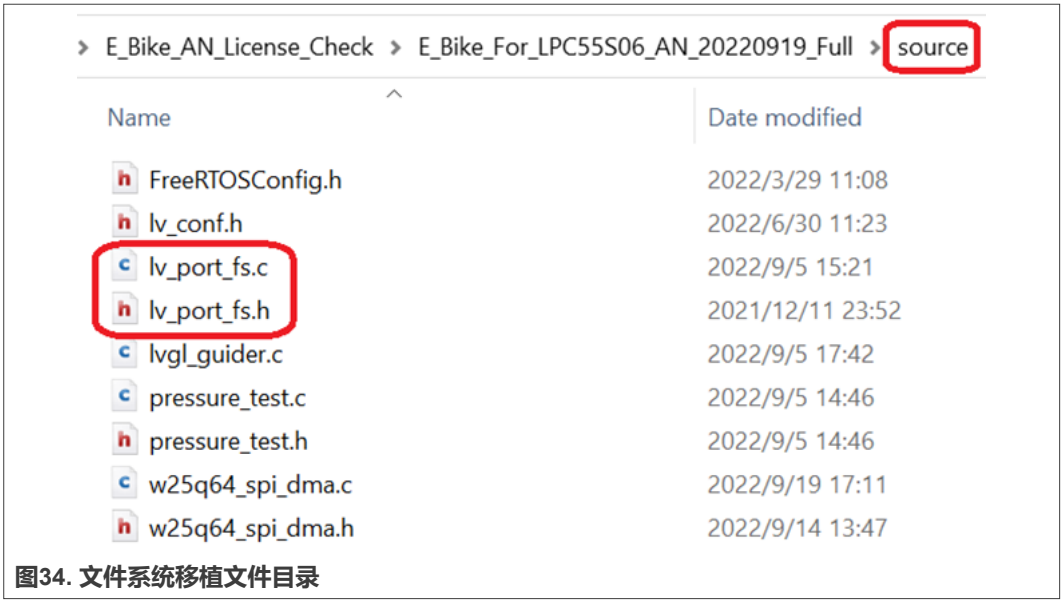


图34. 文件系统移植文件目录

将`lv_port_fs.c`和`lv_port_fs.h`添加到Keil工程的源码组中，如图35所示。

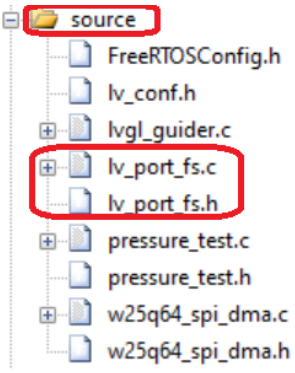


图35. 加文件系统移植文件到Keil工程

10.2 实现文件操作函数

要实现文件操作函数，请参见附带的代码工程中的*lv_port_fs.c*文件。值得注意的文件操作函数包括**lv_port_fs_init**、**fs_init**、**fs_open**、**fs_close**、**fs_read**、**fs_seek**、**fs_tell**和**fs_size**。

这里我们重点介绍**fs_open**函数。**fs_open**函数由几个**if**语句组成，每个**if**语句对应一个二进制图像文件，如图36所示。

```
static void* fs_open(struct _lv_fs_drv_t * drv, const char * path, lv_fs_mode_t mode)
{
  lv_fs_res_t res = LV_FS_RES_NOT_IMP;
  uint32_t* file_p = NULL;

  if (mode == LV_FS_MODE_WR)
  {
  else if (mode == LV_FS_MODE_RD)
  {
  else if (mode == (LV_FS_MODE_WR | LV_FS_MODE_RD))
  {
  /* For Read SPI Flash directly without FileSystem */
  /* ebike_bg.bin */
  if (0 == strcmp(path, "/ebike_bg.bin"))
  {
  /* ebike_header_bg.bin */
  if (0 == strcmp(path, "/ebike_header_bg.bin"))
  {
  /* ebike_gps_arrow.bin */
  if (0 == strcmp(path, "/ebike_gps_arrow.bin"))
  {
    //The location addr of the image in the flash
    FIL.base_addr = 534248;
    //Always 0 at this moment
    FIL.offset = 0;
    //The size of the image
    FIL.size = 4036;
    file_p = (uint32_t*)&FIL;
  }
```

图36. fs_open的部分实现片段

如图36所示，**ebike_gps_arrow.bin**是本示例中使用的二进制图像文件。**base_addr**和**offset**是存储ebike_gps_arrow.bin的外部串行Flash的基地址和地址偏移量。在本示例中，**ebike_gps_arrow.bin**存储在地址534248处，因此可以指定**base_addr为534248**，**offset为0**。**Size**则是**ebike_gps_arrow.bin**的文件大小，这里是4036字节。

这意味着，如果想在GUI应用程序中添加一个新的图像，那么就要将这个图像文件转换为二进制格式，将其下载到外部串行Flash，并添加一个**if**语句。在此**if**语句中，指定基地址、偏移量和文件大小。

要显示已存储在外部串行Flash中的图像文件，请使用图37所示的代码将图像数据从外部Flash加载到LVGL中。

```
//Write codes Overview_image_GPS_icon
ui->Overview_image_GPS_icon = lv_img_create(ui->Overview);
lv_obj_set_pos(ui->Overview_image_GPS_icon, 274, 60);
lv_obj_set_size(ui->Overview_image_GPS_icon, 28, 48);

//Write style state: LV_STATE_DEFAULT for style_overview_image_gps_icon
static lv_style_t style_overview_image_gps_icon_main_main_default;
if (style_overview_image_gps_icon_main_main_default.prop_cnt > 1)
  lv_style_reset(&style_overview_image_gps_icon_main_main_default);
else
  lv_style_init(&style_overview_image_gps_icon_main_main_default);
lv_style_set_img_recolor(&style_overview_image_gps_icon_main_main_defau
lv_style_set_img_recolor_opa(&style_overview_image_gps_icon_main_main_d
lv_style_set_img_opa(&style_overview_image_gps_icon_main_main_default,
lv_obj_add_style(ui->Overview_image_GPS_icon, &style_overview_image_gps
lv_obj_add_flag(ui->Overview_image_GPS_icon, LV_OBJ_FLAG_CLICKABLE);
lv_img_set_src(ui->Overview_image_GPS_icon,"F:/ebike_gps_arrow.bin");
lv_img_set_pivot(ui->Overview_image_GPS_icon, 0,0);
lv_img_set_angle(ui->Overview_image_GPS_icon, 0);
```

图37. 将外部串行Flash中的图像数据加载到LVGL中

11 总结

本应用笔记重点介绍了LVGL和GUI Guider在内存受限的MCU上的应用，包括外部串行Flash的支持、将图像存储到外部串行Flash、充分利用内存资源、使用硬件按钮切换界面，以及文件系统的支持。

与本示例相关的信息，包括代码、图像和图像合并工具，都与本应用笔记一起提供。

注意：*除了链接器文件方面的差异之外，IAR和MCUXpresso IDE的J-Link的加载方式相同。对于MCUXpresso IDE，链接器文件可能比较复杂，而对于IAR，链接器文件应该很简单。*

12 Legal information

12.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

12.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

12.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

目录

For more information, please visit: http://www.nxp.com.cn
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 9 September 2022
Document identifier: AN13730