AN14263

如何基于框架实现LVGL GUI人脸识别功能

第1版—2024年4月19日 **应用笔记**

文档信息

信息	内容
关键词	人脸识别、LVGL GUI、框架
摘要	本应用笔记介绍了如何基于框架启用AI&ML视觉算法模型进行人脸识别，通过一个简单的LVGL GUI示例在SLN-TLHMI-IOT板上实现人脸识别功能。

1 概述

恩智浦推出了一款名为SLN-TLHMI-IOT的解决方案开发套件，专注于智能HMI应用。它可在一个恩智浦i.MX RT117H MCU上实现具有ML视觉、语音和图形用户界面的智能HMI。这个解决方案的软件基于SDK，构建在一个称为框架的设计上，支持视觉和语音功能的灵活设计及定制。为了帮助用户更好地使用此软件平台，提供了一些基础文档，例如软件开发用户指南。该指南介绍了应用程序的基本软件设计和架构，涵盖了此解决方案的所有组件（包括框架），可帮助开发人员更轻松、高效地使用SLN-TLHMI-IOT实现其应用程序。

有关该解决方案和相关文档的更多详细信息，请访问“基于i.MX RT117H、具有ML视觉、语音和图形用户界面（UI）的恩智浦EdgeReady智能HMI解决方案”的网页。

然而，对于开发人员来说，参考这些基本指南来实现他们的智能HMI应用仍非易事。我们计划推出一系列应用笔记，来帮助一步一步地学习基于该框架的开发。本应用笔记基于“利用框架实现LVGL GUI摄像头预览”（文档AN14147）。

本应用笔记介绍了如何基于框架启用AI&ML视觉算法模型，并在SLN-TLHMI-IOT板上通过一个简单的LVGL GUI示例，利用GUI界面上的摄像头预览来实现人脸识别功能。

在本应用笔记中，该示例展示了一个带有摄像头预览功能的LVGL GUI界面，以及一些可触发人脸注册、识别和删除的按钮。已注册的人脸数据通过一个小文件系统存储在Flash中。

本应用笔记概述了以下内容：

- 基于框架启用人脸识别功能。
- 通过Flash上的文件系统为框架添加人脸数据库支持。
- 实现LVGL GUI应用。

通过上述介绍，本文可帮助开发人员完成以下内容：

- 更深入地了解框架和智能HMI解决方案软件。
- 通过LVGL GUI应用开发基于框架的AI&ML人脸识别功能。

1.1 框架概述

此解决方案软件主要围绕框架架构的使用而设计，该架构由以下几个不同的部分组成：

- 设备管理器 – 核心部件
- 硬件抽象层（HAL）设备
- 消息/事件

如图1所示，该框架的机制概述如下：

设备管理器负责管理系统使用的设备。每种设备类型（输入、输出等）都有其特定类型的设备管理器。设备管理器在设备注册后启动，在初始化和启动注册设备后，等待并检查消息以将数据传输到设备和其他管理器。

HAL设备是基于底层驱动程序代码编写的，通过提取许多底层细节，有助于提高代码的可理解性。

事件是在不同设备间通过其管理器传递信息的一种方式。当一个事件被触发时，首先接收到此事件的设备会将该事件传递给它的管理器，然后依次通知被指定接收该事件的其他管理器。

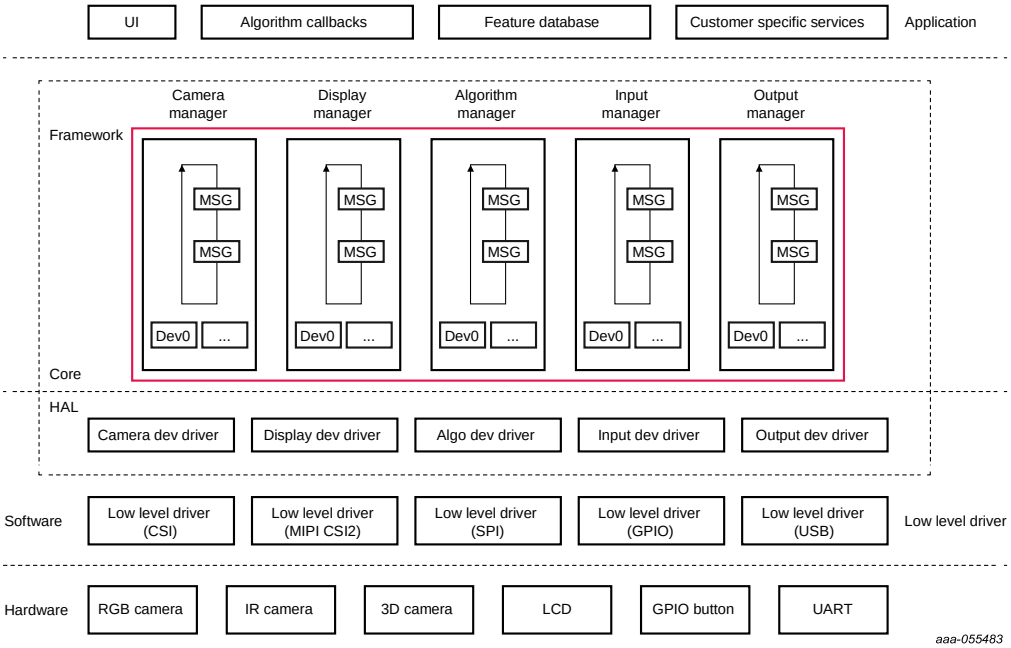

图1. 在框架上的智能HMI软件架构

该框架的架构设计围绕三个主要目标：

1. 易于使用
2. 灵活性/可移植性
3. 性能

该框架的设计目标是加快视觉和其他机器学习应用的上市速度。为确保其快速上市，软件本身易于了解和修改至关重要。从这一目标出发，该框架的架构做到了既易于修改，又不受限制，也不会以牺牲性能为代价。

有关该框架的更多详细信息，请参阅《智能HMI软件开发用户指南》（文档MCU-SMHMI-SDUG）。

1.2 轻量级多功能图形库（LVGL）

LVGL（轻量级多功能图形库）是一个免费的开源图形库，可提供创建嵌入式GUI所需的一切：具有简单易用的图形元素，美观的视觉效果和低内存占用。

1.3 GUI Guider

GUI Guider是恩智浦推出的一款用户友好型图形用户界面开发工具，可利用开源LVGL图形库快速开发高质量显示屏。GUI Guider的拖放编辑器可让您轻松使用LVGL的许多功能，如窗口组件、动画和样式，以最少的编码或根本无需编码即可创建GUI。

只需单击一个按钮，您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中。从GUI Guider生成的代码可轻松添加到工程中，从而加快开发进程，并支持将嵌入式用户界面无缝添加到您的应用程序中。

GUI Guider可免费与恩智浦的通用和跨界MCU一起使用，并包含适用于多个支持平台的内置工程模板。

如需了解有关LVGL和在GUI Guider上进行GUI开发的更多信息，请查阅轻量级多功能图形库和GUI Guider。

2 开发环境

首先，准备并搭建基于框架实现示例的硬件和软件环境。

硬件环境

验证示例的硬件环境搭建：

- 基于NXP i.MX RT117H的智能HMI开发套件（SLN_TLHMI_IOT套件）
- 带有9引脚Cortex-M适配器和V7.84a或更新版本驱动程序的SEGGER J-Link

软件环境

开发示例的软件环境搭建：

- MCUXpresso IDE V11.7.0
- GUI Guider V1.6.1-GA
- `lvgl_gui_camera_preview_cm7` – 第二个应用笔记的示例代码，可作为开发的基础软件。如需了解详细信息，请参阅：https://mcuxpresso.nxp.com/appcodehub。
- RT1170 SDK V2.13.0 – 作为开发的代码资源。
- SLN-TLHMI-IOT软件V1.1.2 – 在恩智浦GitHub代码库中发布的智能HMI源代码，可作为开发的代码资源。如需了解详细信息，请参阅：GitHub - NXP/mcu-smhmi at v1.1.2

有关软件环境的获取和设置，请参阅：SLN-TLHMI-IOT快速入门。

3 框架上的视觉架构

此框架上的视觉架构如图2所示。视觉算法HAL `(OASIS_HAL)` 有以下流程：

- 在接收到来自输出用户界面HAL的相关事件后，通过AI&ML视觉算法模型进行人脸注册和识别。将算法模型的推理结果通知给输出用户界面HAL。
- 在接收到来自输出用户界面HAL的相关事件后，通过调用FaceDB HAL的API，访问（添加、删除……）基于小文件系统的人脸特征数据库。
- 在进行人脸注册和识别时，向摄像头HAL请求摄像头视频帧。

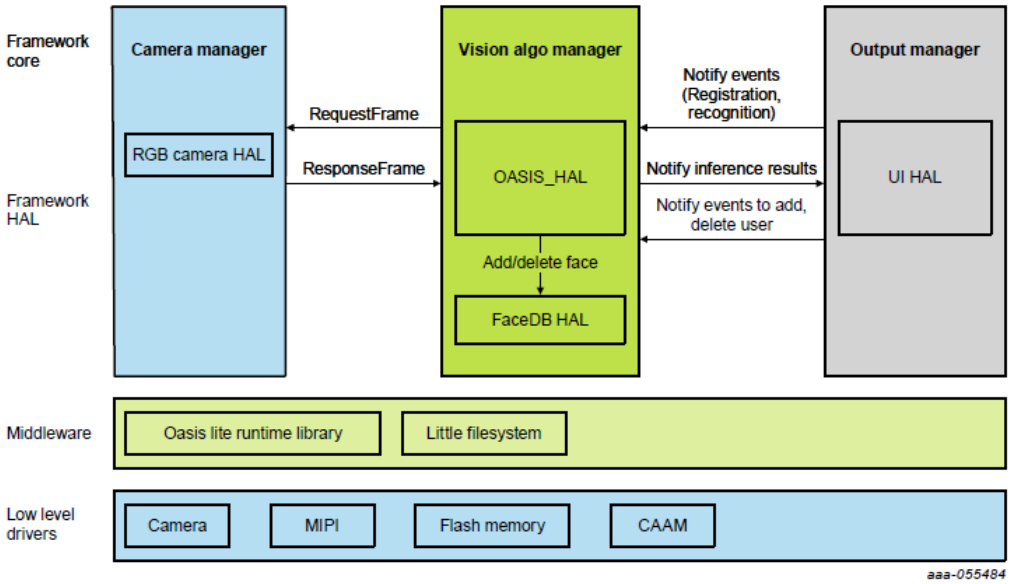

图2. 视觉架构

4 基于框架实现人脸识别功能

在框架上的LVGL GUI人脸识别示例（示例稍后提供）是基于《利用框架实现LVGL GUI摄像头预览》（文档AN14147）中的示例代码实现的。

为了演示示例中的人脸识别功能，GUI应用程序（见图3主界面）的基本功能设计如下：

- 当点击**Registration**或**Recognition**按钮时，GUI应用程序会向输出用户界面HAL触发人脸注册或识别事件。当人脸注册成功后，输出用户界面HAL会通知一个将用户添加到视觉算法HAL的事件。
- 在识别出用户人脸后单击**Delete User**按钮，GUI应用程序会向输出用户界面HAL触发删除用户的事件。
- 当单击按钮和图像之外的界面时，GUI应用程序会向输出用户界面HAL触发停止运行oasis算法的事件。

图3. LVGL GUI人脸识别应用主界面

软件准备

准备用于实现示例的软件包。

- 克隆基础软件`lvgl_gui_camera_preview_cm7`。将工程名称和主文件名更改为`lvgl_gui_face_rec_cm7`。
- 由于框架核心的源代码从1.1.2版开始在GitHub上公开，因此需要在软件中更新框架。
- 将框架文件夹替换从GitHub上复制的V1.1.2，但`inc\`下的`fwk_log.h`和`fwk_common.h`文件除外，因为它们已根据应用笔记系列进行了修改。这些操作如图4所示：

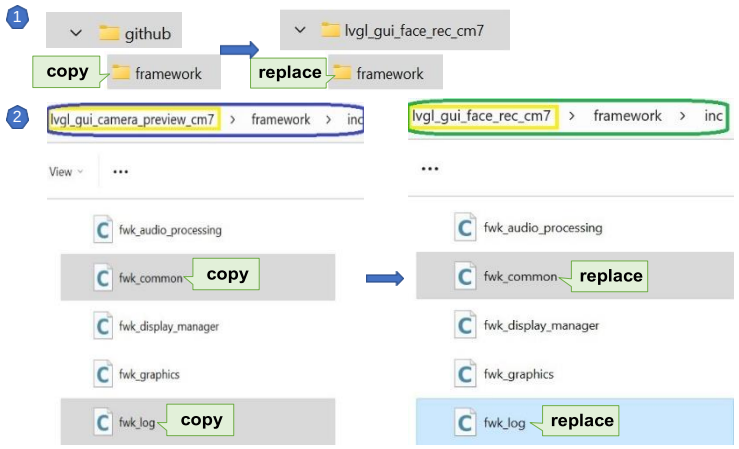

图4. 更新框架

- 删除`libs`组下的`framework_cm7`文件夹，并删除`Project > Properties > C/C++ Build > settings > Tool Settings > MCU C++ Linker > Libraries`中配置的库`framework_cm7`及其搜索路径，因为核心的源代码已提供。

4.1 基于框架启用人脸识别功能

人脸识别功能建立在以静态库形式提供的ML视觉算法模型之上，即恩智浦的oasis lite运行时库。该库是一个小型、高效、定制和优化的AI库。该模型包括人脸检测、人脸识别、玻璃检测和活体检测。

它主要提供`OASISLT_run_extended()`这个`API`来运行人脸识别管道，同时通过事件回调将结果更新给调用者，并通过在初始化时调用另一个`API`即`OASISLT_init()`，来指定源框架信息、回调和库使用的内存池后，通过人脸数据库回调添加/更新/删除数据库中的人脸。

API调用和回调函数在框架的视觉算法HAL中实现。

4.1.1 添加视觉算法模型库

1. 将包含库和相关头文件的`oasis`文件夹从智能`HMI\coffee_machine\cm7\libs\`复制到示例软件的`libs`文件夹。
2. 在`Project > Properties > C/C++ Build > settings > Tool Settings > MCU C compiler > Includes`和`MCU C++ compiler > Includes`中添加此头文件的搜索路径：

```
"${workspace_loc:/${ProjName}/libs/oasis/include}"
```

3. 添加库和其搜索路径，依次点击Project > Properties > C/C++ Build > settings > MCU C++ + Linker > Libraries：

```
liboasis_lite2D_DEFAULT_117f_ae.a
"${workspace_loc:/${ProjName}/libs/oasis}"
```

并添加宏定义来启用Project > Properties > C/C++ Build >settings > Tool Settings > MCU C compiler > Preprocessor和MCU C++ compiler > Preprocessor功能：

```
SMART_TLHMI_2D
```

4.1.2 启用视觉算法HAL

视觉算法HAL驱动视觉算法模型工作，并在收到来自用户界面输出HAL的事件后将结果响应给用户界面输出HAL。

要启用它，请克隆现有的类似HAL驱动程序文件，其中可实现以下功能：

- 实现人脸数据库操作和事件处理的回调。
- 通过调用oasis库的API来驱动视觉算法工作。
- 访问用户人脸数据库和应用程序数据库（示例中不需要）。
- 接收事件并将结果发送到输出用户界面HAL中。

实现该示例的HAL的主要工作包括：

- 克隆现有的类似HAL驱动程序文件并更改相关名称。
- 删除与应用程序数据操作相关的代码。
- 根据示例设计更新用于处理输出用户界面HAL事件的定义和函数。
- 添加oasis初始化所需的配置。

具体步骤如下：

1. 克隆hal_vision_algo_oasis_coffeemachine.c。将文件名更改为hal_vision_algo_oasis_guifacerec.c。并将文件中的所有CoffeeMachine字符串替换为GUIFaceRec。
2. 删除包含与应用程序数据库相关的字符串coffeedb（不区分大小写）的代码，例如#include hal_sln_coffeedb.h。
3. 修改函数HAL_VisionAlgoDev_OasisGUIFaceRec_InputNotify()以处理来自输出用户界面HAL的事件。
   - 将事件定义kEventFaceRecId_RegisterCoffeeSelection更改为kEventFaceRecId_RegisterUserFace，并将结构字符串regCoffeeSelection更改为regGUIFaceRec，以便在事件处理时将新的人脸特征数据添加到数据库。
   - 为了显示示例中人脸识别操作的标准流程，请根据状态定义修改kEventFaceRecID_OasisSetState的处理方式：

```
kOASISLiteState_Registration
kOASISLiteState_Recognition
kOASISLiteState_Stopped
```

4. 添加和修改上一步中提到的事件定义。
   - 将头文件smart_tlhmi_event_descriptor.h从smart HMI\coffee_machine\cm7\source\event_handlers\复制到软件示例的source文件夹。更新文件如下：
   - 将枚举类型_event_smart_tlhmi_id中的事件定义从kEventFaceRecId_RegisterCoffeeSelection更改为kEventFaceRecId_RegisterUserFace，并将_event_smart_tlhmi结构体中的结构字符串从regCoffeeSelection更改为regGUIFaceRec。

因此，将regCoffeeSelection的结构体register_coffee_selection_event_t更改为register_gui_facerec_event_t。

- 删除咖啡机应用程序所用的其他内容，例如有关语音的代码行：#include "hal_event_descriptor_voice.h"。
- 将kOASISLiteState_Stopped和kOASISLiteState_Running类型添加到工程中framework>hal>vision下hal_vision_algo.h中的枚举类型oasis_lite_state_t，如下所示：

```
typedef enum _oasis_lite_state
{
    kOASISLiteState_Running,
    kOASISLiteState_Stopped,
    kOASISLiteState_Recognition,
    kOASISLiteState_Registration,
    kOASISLiteState_DeRegistration,
    kOASISLiteState_RemoteRegistration,
    kOASISLiteState_Count
} oasis_lite_state_t;
```

- 使用上述更新的结构体oasis_lite_state_t来完善此工程中framework>hal>vision下hal_event_descriptor_face_rec.h中的结构体oasis_state_event_t，如下所示：

```
typedef struct _oasis_state_event_t
{
    oasis_lite_state_t state;
} oasis_state_event_t;
```

5. 将所有kEventInfo_Remote更改为kEventInfo_Local，以便将事件从视觉算法HAL发送到在同一内核上运行的其他HAL，因为示例中使用的是单核而非双核。
6. 在OASISLT_init()中添加并修改以下oasis初始化的配置：
   - 在board_define.h中为视频帧添加宏定义和内存分区：

```
#define OASIS_RGB_FRAME_WIDTH      800
#define OASIS_RGB_FRAME_HEIGHT    600
#define OASIS_RGB_FRAME_SRC_FORMAT kPixelFormat_YUV1P444_RGB
#define OASIS_RGB_FRAME_BYTE_PER_PIXEL 3

#define AT_FB_SHMEM_SECTION_ALIGN(var, alignbytes) \
    __attribute__((section(".bss.$fb_sh_mem,\"aw\",%nobits @"))) var
 __attribute__((aligned(alignbytes)))
```

   - 在Project > Properties > C/C++ Build > MCU Settings中将内存分配配置到上述内存分区fb_sh_mem，如图5所示：

Type	Name	Alias	Location	Size	Driver
Flash	BOARD_FLASH	Flash	0x30000000	0x800000	MIMXRT1170_S...
RAM	BOARD_SDRAM	RAM	0x80000000	0x1000000	
RAM	NCACHE_REGION	RAM2	0x81000000	0x400000	
RAM	SRAM_DTC_cm7	RAM3	0x20000000	0x80000	
RAM	SRAM_ITC_cm7	RAM4	0x0	0x40000	
RAM	SRAM_OC1	RAM5	0x20240000	0x80000	
RAM	SRAM_OC2	RAM6	0x202c0000	0x80000	
RAM	SRAM_OC_ECC1	RAM7	0x20340000	0x10000	
RAM	SRAM_OC_ECC2	RAM8	0x20350000	0x10000	
RAM	SRAM_OC_cm7	RAM9	0x20360000	0x20000	
RAM	res_sh_mem	RAM10	0x81400000	0x100000	
RAM	fb_sh_mem	RAM11	0x81500000	0x200000	

图5. fb_sh_mem分区的内存分配

- 在lvgl_gui_face_rec_cm7.cpp中声明全局变量g_DTCOPBuf：

```
AT_NONCACHEABLE_SECTION_ALIGN_DTC
(uint8_t g_DTCOPBuf[DTC_OPTIMIZE_BUFFER_SIZE], 4);
```

- 继续添加上述变量中使用的定义：
- 在board_define.h中定义上述部分：

```
#define AT_NONCACHEABLE_SECTION_ALIGN_DTC(var, alignbytes) \
__attribute__((section(".bss.$SRAM_DTC_cm7,\"aw\",%nobits @"))) var
__attribute__((aligned(alignbytes)))
```

- 包括头文件hal_vision_algo.h，包含lvgl_gui_face_rec_cm7.cpp中所含app_config.h中的宏定义DTC_OPTIMIZE_BUFFER_SIZE。

7. 将s_debugOption变量设置为true，以显示人脸识别的进度状态。
8. 在Project > Properties >C/C++ Build> settings > Tool Settings > MCU C compiler > Includes和MCU C++ compiler > Includes中添加视觉HAL头文件的搜索路径：

```
"${workspace_loc:/${ProjName}/framework/hal/vision}"
```

9. 在board_define.h中添加以下定义，以启用视觉算法HAL：

```
#define ENABLE_VISIONALGO_DEV_Oasis_GUIFaceRec
```

4.1.3 启用输出用户界面HAL

输出用户界面HAL将事件通知给视觉算法HAL，并对视觉算法HAL的推理结果作出响应。在GUI应用程序中，这些事件通常由此应用程序触发，并将结果显示在此应用程序上。

要启用该功能，请克隆现有的类似HAL驱动程序文件，通常可在该文件中实现以下功能：

- 通知人脸识别和数据库访问事件。
- 实现GUI应用程序的回调来触发事件。
- 处理视觉算法模块的推理结果。
- 通过定时器和人脸引导矩形框控制的进度条，在用户界面上显示事件处理的过程和结果。

为本文档所用示例实现HAL的主要工作包括：

- 克隆现有的类似HAL驱动程序文件并更改相关名称。
- 删除与此应用程序相关的代码。

- 根据示例设计更新事件通知和结果响应的函数。
- 添加GUI应用程序的回调来触发事件。

具体步骤如下：

1. 克隆hal_output_ui_coffee_machine.c。将文件名更改为hal_output_ui_guifacerec.c。
2. 将文件中的所有CoffeeMachine字符串替换为GUIFaceRec。
3. 删除与咖啡机应用程序相关的代码。
   - 删除WakeUp()和StandBy()函数及相关代码（可搜索字符串wake_up和standby）。
   - 删除HAL_OutputDev_UiGUIFaceRec_Input Notify()中与预览模式事件处理相关的代码。
   - 除预览模式功能的gui_set_virtual_face()之外，删除函数UI_xxx_Callback()、含gui_字符串的代码以及与咖啡机GUI相关的界面。
   - 删除与咖啡机应用程序相关的变量s_IsWaitingAnotherSelection和s_IsWaitingRegisterSelection所涉及的代码。

     ```
     #include "hal_voice_algo_asr_local.h",
     #include "hal_event_descriptor_voice.h"
     ```

   - 删除与语音、音频和语言相关的代码。例如：
4. 对于各种事件通知，在删除它们之前，参考函数StopFaceRec()、RegisterCoffeeSelection()和DeregisterCoffeeSelection()，实现新函数OutputManagerNotify()、SetFaceRec()、RegisterGUIFaceRec()和DeregisterGUIFaceRec()。
   - _OutputManagerNotify()实现了基本事件输出，即将事件发送到视觉算法HAL。以下函数可调用它来发送自己的事件。
   - _SetFaceRec()可发送事件kEventFaceRecID_OasisSetState，以触发视觉算法进行人脸注册、识别并停止算法。
   - 当注册成功后,_RegisterGUIFaceRec()会发送smart_tlhmi_event_descriptor.h中定义的事件kEventFaceRecId_RegisterGUIFaceRec，以将人脸特征数据添加到数据库中。
   - 当人脸识别通过时，DeregisterGUIFaceRec()会发送事件kEventFaceRecID_DelUser，从数据库中删除人脸特征数据。
5. 更新代码采取相应操作，包括根据示例设计，在_InferComplete_Vision()函数中调用LVGL GUI应用的API来刷新GUI，以获取人脸注册和识别的推理结果。例如，当人脸注册成功时，
   - 通过调用_FaceRecProcess_Stop()可停止显示进度；
   - 通过调用_SetFaceRec(kOASISLiteState_Stopped)可停止人脸注册；
   - 在GUI上显示成功结果：gui_show_face_rec_result(kFaceRecResult_OK,s_UserId)；
   - 将人脸数据注册到数据库：_RegisterUserFace(s_UserId)；
6. 添加用户界面回调函数来处理从GUI中触发的事件：预览、人脸注册、识别和删除用户等。例如，人脸注册回调函数：

   ```
   void UI_Registration_Callback()
   {
    _SetFaceRec(kOASISLiteState_Registration);
     FaceRecProcess_Start();
   ```

```
}
```

- 添加_FaceRecProcess_Start()和_FaceRecProcess_Stop()函数，以显示不同事件和结果的进度和状态。
- 更新定时器ISR回调函数_SessionTimer_Callback()，以通过调用来处理超时情况：

```
gui_show_face_rec_result(kFaceRecResult_TimeOut,
                         s_UserId);
```

7. 在board_define.h中添加以下定义，以启用用户界面输出HAL：

```
#define ENABLE_OUTPUT_DEV_UiGUIFaceRec
```

注：

为了更好地呈现人脸识别功能，在输出用户界面HAL中保留显示人脸识别过程和结果的功能。此功能描述如下：

- *当开始人脸注册或识别时，人脸引导矩形框显示为蓝色，进度条显示进度。*
- *当人脸注册成功时，人脸引导矩形框显示为红色。*
- *当人脸识别成功时，人脸引导矩形框显示为绿色。*
- *当定时器超时后，操作不成功时，人脸引导矩形框会保持蓝色，进度条显示完整进度。此时，请停止人脸注册或识别。*

进度条和人脸引导矩形框都是以图标显示的，这些图标内置在要编入Flash的二进制资源文件中。在输出用户界面HAL设备初始化时调用的LoadIcons(APP_ICONS_BASE)函数中，设置了指向SDRAM上图标数据的指针。它必须实现对该函数的图标支持。

4.1.4 实现图标的支持

1. 构建将图标与LVGL GUI应用中所用图像相结合的资源：
   - 从smart HMI\coffee machine\resource\icons\克隆4个图标头文件process_bar_240x14.h、virtual_face_blue_420x426.h、virtual_face_green_420x426.h和virtual_face_red_420x426.h，复制到示例软件的resource文件夹下的新文件夹icons中。
   - 将4个图标文件的搜索路径添加到resource文件夹的camera_preview_resource.txt文件中：例如 icon ../resource/icons/process_bar_240x14.h
   - 执行camera_preview_resource_build.bat，以构建图像和图标资源，生成二进制文件camera_preview_resource.bin和信息文件resource_information_table.txt（参见图6）。

```
_NxpFaceRec_alpha_185x55.data = (base + 0);
_NxpVoiceRec_alpha_185x55.data = (base + 30528);
s_Icons[process_bar_240x14] = (base + 0);
s_Icons[virtual_face_blue_420x426] = (base + 6720);
s_Icons[virtual_face_green_420x426] = (base + 364608);
s_Icons[virtual_face_red_420x426] = (base + 722496);

Images Total: 0x00ee80, 61056

Icons Total: 0x107c40, 1080384
```

图6. 在resource_information_table.txt中生成的信息

2. 在app config.h中定义SDRAM的起始地址和图标大小。该地址应紧接在GUI应用程序的图像数据之后。图标的大小在信息文件中生成。

```
#define APP_ICONS_BASE (APP_RES_SHMEM_BASE +
                        APP_LVGL_IMGS_SIZE)
#define APP_ICONS_SIZE 0x107c40
```

3. 通过在app_config.h中将名为res_sh_mem的内存分区重新定义，将其指定大小更新为0x200000：

```
#define RES_SHMEM_TOTAL_SIZE 0x200000
```

并在Project > Properties > C/C++ Build > MCU settings中更新其相应设置。

4. 在主文件lvgl_gui_face_rec_cm7.cpp中的APP_LoadResource()函数中，将图标大小添加到从Flash加载到SDRAM的资源的总大小中。

```
memcpy((void *)APP_LVGL_IMGS_BASE, pLvglImages, APP_LVGL_IMGS_SIZE +
                    APP_ICONS_SIZE);
```

注：*要完成人脸识别功能，需要LVGL GUI应用程序的支持。输出用户界面HAL中的用户界面回调函数由LVGL GUI应用程序调用，用于处理用户界面的事件。另一方面，输出用户界面HAL调用LVGL GUI应用程序的API来更新用户界面，以显示结果和状态。LVGL GUI应用程序的开发相对独立，请参见第4.3节。*

4.1.5 启动人脸识别的HAL设备和管理器

已启用的视觉算法HAL和用户界面输出HAL及其管理器在主文件lvgl_gui_face_rec_cm7.cpp中启动，按照基于框架的开发进行如下转换：

1. 通过添加如下代码行来包含与两个HAL管理器相关的头文件：

```
#include " fwk_output_manager.h "
#include "fwk_vision_algo_manager.h"
```

2. 声明HAL设备：

```
HAL_VALGO_DEV_DECLARE(OasisGUIFaceRec);
HAL_OUTPUT_DEV_DECLARE(UiGUIFaceRec);
```

3. 注册HAL设备：

```
HAL_VALGO_DEV_REGISTER(OasisGUIFaceRec, ret);
HAL_OUTPUT_DEV_REGISTER(UiGUIFaceRec, ret);
```

4. 初始化管理器：

```
FWK_MANAGER_INIT(VisionAlgoManager, ret);
FWK_MANAGER_INIT(OutputManager, ret);
```

5. 启动管理器：

```
FWK_MANAGER_START(VisionAlgoManager, VISION_ALGO_MANAGER_TASK_PRIORITY, ret);
FWK_MANAGER_START(OutputManager, OUTPUT_MANAGER_TASK_PRIORITY, ret);
```

6. 定义管理器任务的优先级：

```
#define VISION_ALGO_MANAGER_TASK_PRIORITY 3
#define OUTPUT_MANAGER_TASK_PRIORITY 1
```

4.2 基于框架添加人脸数据库支持

已注册的人脸特征数据通过一个小文件系统从存储在Flash上的人脸数据库中访问。添加人脸数据库支持的步骤如下所述。

4.2.1 添加Flash存储的驱动程序

将Flash接口FlexSPI驱动程序文件fsl_flexspi.c和fsl_flexspi.h以及数据加密驱动程序文件fsl_caam.c和fsl_caam.h，从SDK_2_13_0_MIMXRT1170-EVK\devices\MIMRX1176\drivers\路径复制到示例软件的drivers文件夹中。

4.2.2 添加板级支持

1. 在board.h中添加用于板载Flash设备所用的FlexSPI定义：

```
#define BOARD_FLEXSPI    FLEXSPI1
#define BOARD_FLEXSPI_CLOCK  kCLOCK_FlexSpi1
#define BOARD_FLEXSPI_AMBA_BASE FlexSPI1_AMBA_BASE
```

2. 将smart HMI\coffee_machine\cm7\source\flash_config\路径下Flash设备的运算符和配置文件flexspi_nor_flash_ops.c、flexspi_nor_flash_ops.h、sln_flash_config.c、sln_flash_config_w25q256jvs.h和andsln_flash_ops.h复制到示例软件的board文件夹。
   - 右键单击文件名并打开其属性，取消勾选C/C++ Build > Settings中的“Exclude resource from build”选项以使其构建到此工程中。
3. 在sln_flash_config.c和flexspi_nor_flash_ops.h中将所含的头文件名sln_flash_config.h更改为sln_flash_config_w25q256jvs.h。
4. 参考咖啡机应用程序，在文件clock_config.c中设置**FlexSPI1**时钟源。

4.2.3 添加适配器和中级支持

1. 将文件sln_flash.c、sln_flash.h、sln_encrypt.c和sln_encrypt.h作为文件系统和应用程序的适配器驱动程序从smart HMI\coffee_machine\cm7\source\路径复制到示例的source文件夹。更新新文件：
   - 取消勾选“Exclude resource from build”选项以进行构建。
   - 将所有包含的头文件名从sln_flash_config.h更改为sln_flash_config_w25q256jvs.h。
2. 将filesystem文件夹（包含小文件系统的API和HAL驱动程序）从smart HMI\coffee_machine\cm7\source\复制到示例软件。并更新新文件夹：
   - 取消勾选“Exclude resource from build”选项以进行构建。

- 在工程设置中为其添加包含路径：“${workspace_loc:/${ProjName}/filesystem}”
- 在sln_flash_littlefs.h文件中，将包含的头文件名sln_flash_config.h更改为sln_flash_config_w25q256jvs.h，以及将fica_definition.h更改为app_config.h。

3. 将littlefs文件夹（包含小文件系统的中间件）从SDK_2_13_0_MIMXRT1170-EVK\middleware\路径复制到示例软件。并更新新文件夹：
   - 取消勾选“Exclude resource from build”选项以进行构建。
   - 在工程设置中为其添加包含路径：

```
"${workspace_loc:/${ProjName}/littlefs}"
```

4.2.4 添加AL驱动程序

有两个HAL设备-文件系统和人脸数据库HAL支持数据库访问功能，且它们无需进行任何更改就已在框架中实现。通过在board_define.h中添加以下定义即可启用：

```
#define ENABLE_FLASH_DEV_Littlefs
#define ENABLE_FACEDB
```

还要更改示例中的人脸数据库名称：

```
#define OASIS_FACE_DB_DIR "oasis_gui_face_rec"
```

4.2.5 添加应用级支持

1. 更新主文件lvgl_gui_face_rec_cm7.cpp：
   - 通过添加以下代码行来包含与Flash文件系统HAL管理器相关的头文件：#include "fwk_flash.h"
   - 声明并注册文件系统HAL设备：

```
HAL_FLASH_DEV_DECLARE(Littlefs);
HAL_FLASH_DEV_REGISTER(Littlefs, ret);
```

   ***注：**在APP_InitFramework()函数中初始化所有设备管理器之前，必须注册文件系统HAL设备。*

   - 调用APP_BoardInit()中的BOARD_ConfigMPU()函数来配置MPU。
2. 通过定义sln_flash_littlefs.h文件中所用的宏定义，在app_config.h文件中设置Flash上的文件系统分配：

```
#define FICA_IMG_FILE_SYS_ADDR (FLASH_IMG_SIZE + RES_SHMEM_TOTAL_SIZE)
#define FICA_FILE_SYS_SIZE (0x280000)
```

4.2.6 配置

某些与Flash相关的代码会在SRAM ITC区域执行，以获得足够的性能。将包含链接器配置的linkscripts文件夹从smart HMI\coffee_machine\cm7\路径复制到示例软件。

4.3 实现LVGL GUI应用程序

基于框架的LVGL GUI应用的开发可调用输出用户界面HAL的API，并向输出用户界面HAL提供API（如需了解输出用户界面HAL的实现，请参见第4.1.3节）。

然而，LVGL GUI应用程序的具体实现取决于此应用程序的要求和设计。本示例中的GUI应用程序按照第4节的开头部分的描述进行设计。

以下是实现的介绍：

1. 自定义代码在GUI Guider提供的`custom.c`中实现，而`custom.h`作为GUI Guider工程与嵌入式系统工程之间的接口。
   - 在`custom.c`中添加名为`gui_xxx()`的新函数，以实现以下功能：
     – 用于输出用户界面HAL和GUI应用程序来更新用户界面。
     – 让GUI应用程序通过从输出用户界面HAL中调用用户界面回调函数来触发事件。

   例如，当相关按钮被点击时，新函数`gui_event_face_rec_action()`会调用用户界面回调函数来处理从GUI应用程序触发的一个人脸注册、人脸识别或删除用户事件。

   ***注：** 在预览模式下，输出用户界面HAL中调用的`gui_set_virtual_face()`函数需要在`custom.c`中实现：*
   - *从`smart HMI\coffee_machine\cm4\custom\custom.c`中克隆`gui_set_virtual_face()`函数。*
   - *在函数中将窗口组件的名称从`home_img_cameraPreview`更改为`screen_img_camera_preview`。*
   - *在`custom.c`中的宏定义`#ifndef RT_PLATFORM`的控制下，对输出用户界面HAL中的所有用户界面回调函数，以同一原型实现，以便与GUI Guider工程兼容，因为输出用户界面HAL中的这些函数与嵌入式平台有关。在`custom.c`中，它们取决于GUI guider的模拟器，与嵌入式平台无关。例如，在运行GUI Guider模拟器时，人脸注册回调的实现如下：*

```
#ifndef RT_PLATFORM
void UI_Registration_Callback()
{
  gui_hide_del_user_btn(true);
  s_InAction = false;
  return;
}
```

   ***注：** 请参考第4.1.3 节第6步中介绍的相同函数原型。*

   在MCUXpresso的工程设置中设置了`RT_PLATFORM`宏定义，如图7所示：

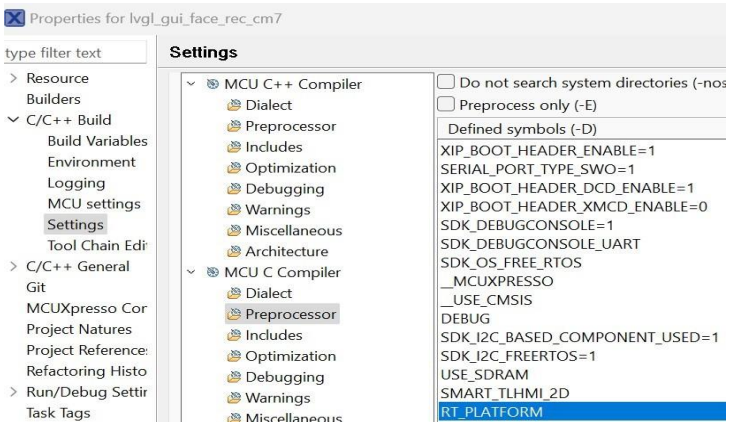

图7. 工程设置中宏定义“RT_PLATFORM”的设置

- 在custom.h中声明所有名为UI_xxx_Callback()和gui_xxx()的函数，并在smart_tlhmi_event_descriptor.h中添加custom.h，以便将GUI API共享到用户界面输出HAL。

2. 在GUI Guider上开发GUI：
   - 克隆基础软件包lvgl_gui_camera_preview_cm7中gui_guider文件夹中的camera preview文件夹（含GUI Guider工程软件）。将新示例的相关名称从camera_preview更改为face_rec。
   - 将上述更新的custom.c和custom.h复制到新GUI Guider工程软件中。
   - 打开GUI Guider上的新face_rec工程。更新如下：
     – 添加标有**Delete User**的新按钮。添加**Hidden**标记，以便在GUI应用程序启动时隐藏该按钮。
     – 在所有按钮**Registration**、**Recognition**和**Delete User**的**Event Setting**中的“Released”触发器上，添加调用API gui_event_face_rec_action()（带有不同事件ID参数）的代码行，用于触发人脸注册、人脸识别和删除用户事件。图8显示了**Registration**按钮事件的代码：

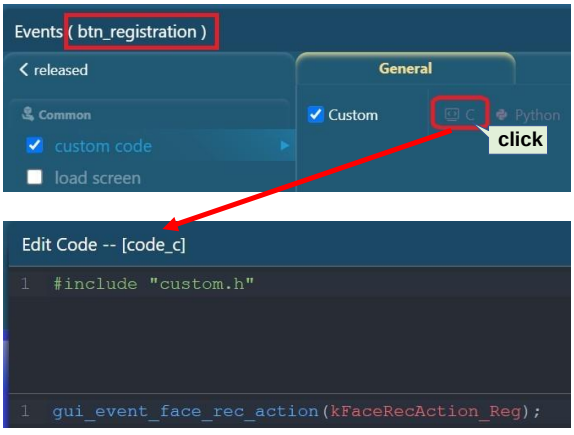

图8. “Registration”按钮事件的编码

3. 将GUI Guider生成的代码更新到MCUXpresso工程。
   - 将MCUXpresso工程软件的generated文件夹中（images文件夹除外）的内容替换为GUI Guider工程软件生成的文件夹中的相应内容。

***注：**有关上述修改的更多详细信息，请通过以下网址查看示例软件：*
https://mcuxpresso.nxp.com/appcodehub。

5 示例工程验证

如需获取包含本应用笔记资源和工具的示例软件包，请访问：https://mcuxpresso.nxp.com/appcodehub。

在MCUXpresso IDE上打开示例工程。构建`.axf`文件并将其烧录到0x30000000地址，并将资源二进制文件`camera_preview_resource.bin`烧录到0x30800000地址。

LVGL GUI人脸识别示例正常工作如下：

- **Preview（预览）：** 开机后，摄像头捕获的视频流会显示在GUI界面上摄像头预览的特定区域。状态标签显示“Preview...”。详见图3。隐藏**Delete User**按钮。当点击按钮和图像以外的区域时，人脸注册或识别操作结束后会显示预览状态，如上图所示。
- **Registration（注册）：**
  - **Startup（启动）：** 点击**Registration**按钮后，开始人脸注册。状态标签变为显示“Registration...”，人脸引导矩形框显示为蓝色，进度条开始显示进度。确保用户的脸部显示在蓝色的人脸引导矩形框内，以便进行注册。
  - **Success（成功）：** 状态标签显示“Registration...OK”和注册用户ID编号，如果在进度条显示满之前人脸注册成功，则人脸引导矩形框会变为红色。
  - **Failure** -> Time out（**失败** -> 超时）：如果当进度条上显示进度已满时，人脸注册仍然失败，则状态标签会显示“Registration...Time out”。
  - **Failure** -> Duplication（**失败** -> 复制）：状态标签显示“Registration...Failed”，如果进度条显示已满时，已注册的人脸被识别，则人脸引导矩形框会变为绿色。
- **Recognition（识别）：**
  - **Startup（启动）：** 点击**Recognition**按钮后，即开始人脸识别。状态标签变为显示“Recognition...”，人脸引导矩形框显示为蓝色，进度条开始显示进度。确保用户的脸部显示在蓝色的人脸引导矩形框中，以便进行注册。
  - **Success（成功）**：状态标签显示“Recognition...OK”和已识别的用户ID编号，如果在进度条显示满之前，人脸识别成功，则人脸引导矩形框变为绿色。此时会出现**Delete User**按钮。这表示只有当用户被识别时才允许删除该用户。
  - **Failure（失败）：** 如果在进度条上显示进度已满时，人脸识别仍然失败，状态标签会显示“Recognition...Time out”。
- **Delete User（删除用户）：** 在人脸识别成功后，点击“Delete User”按钮，状态标签会变成“Delete User...OK”，人脸引导矩形框会变为蓝色，进度条上显示已满。**Delete User**按钮会再次隐藏。已识别的面孔/用户将从数据库中删除。这意味着该人脸/用户在重新注册之前将无法被识别。

6 关于本文中源代码的说明

本文中所示的示例代码具有以下版权和BSD-3-Clause许可：

2024年恩智浦版权所有；在满足以下条件的情况下，可以源代码和二进制文件的形式重新分发和使用本源代码（无论是否经过修改）：

1. 重新分发源代码必须保留上述版权声明、这些条件和以下免责声明。
2. 以二进制文件形式重新分发时，必须在文档和/或随分发提供的其他材料中复制上述版权声明、这些条件和以下免责声明。
3. 未经事先书面许可，不得使用版权所有者的姓名或参与者的姓名为本软件的衍生产品进行背书或推广。

7 修订历史

表1. 修订历史

文档ID	发布日期	说明
AN14263 v.1	2024年4月19日	初始版本

Legal information

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.

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 https://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 document 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.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

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: https://www.nxp.com.cn

Date of release: 19 April 2024
Document identifier: AN14263