AN14250

如何基于框架实现LVGL GUI语音识别功能

第1版—2024年3月26日 **应用笔记**

文档信息

信息	内容
关键词	AN14250、智能HMI、智能TLHMI、框架
摘要	本应用笔记介绍了如何基于框架使用DSMT或VIT模型来进行语音识别，并通过一个简单的LVGL GUI示例在SLN-TLHMI-IOT板上实现此功能。

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情，请访问以下网页：

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 | 恩智浦半导体

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。本文档基于一个应用笔记，展示了如何通过一个简单的GUI摄像头预览示例，实现一个基于框架的LVGL GUI。

本应用笔记介绍了如何通过SLN-TLHMI-IOT板的一个简单的LVGL GUI示例，使用支持中英文的DSMT或VIT模型，实现基于框架的语音识别功能。

在本应用笔记中，该示例展示了一个带摄像头预览和一些按钮的LVGL GUI界面，用户可以通过语音命令或触摸这些按钮来激活操作界面。

本应用笔记概述了以下内容：

- 基于框架启用语音识别功能。
- 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：

- 更深入地了解该框架和智能HMI解决方案软件。
- 通过LVGL GUI应用开发基于框架的语音识别功能。

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，可初始化和启动注册设备，然后等待并检查消息以将数据传输给设备和其他管理器。

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的其他管理器。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

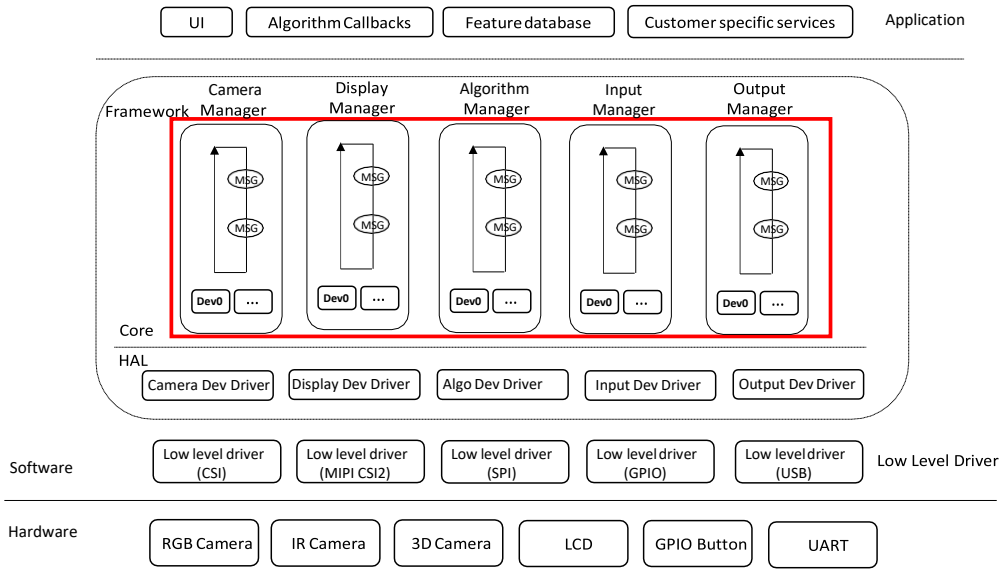

图1. 基于框架的智能HMI软件架构

该框架的架构设计围绕三个主要目标：

- 易于使用
- 灵活性/可移植性
- 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，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地添加到应用程序中。

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。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

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` — 基于框架实现的LVGL GUI的示例代码。请访问：https://mcuxpresso.nxp.com/appcodehub
  – RT1170 SDK V2.13.0 — SDK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。请访问：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 基于框架的语音识别架构

当前应用笔记中基于框架的语音架构如图2所示。

框架HAL的流程如下：

- 输入PDM MIC HAL将麦克风录制的原始语音流发送到音频处理AFE HAL。
- 音频处理AFE HAL会触发Voiceseeker算法对语音数据流进行预处理，然后将干净的数据流发送到语音算法DMST ASR HAL或语音算法VIT ASR HAL中。
- 语音算法DMST或VIT ASR HAL会触发相应的语音识别算法和模型来识别语音，并将结果发送到输出用户界面HAL。
- 输出用户界面HAL根据结果进行操作。

***注：**本应用笔记中不支持任何扬声器。这意味着应用笔记中不需要处理回声流。*

© 2024 NXP B.V. 版权所有。

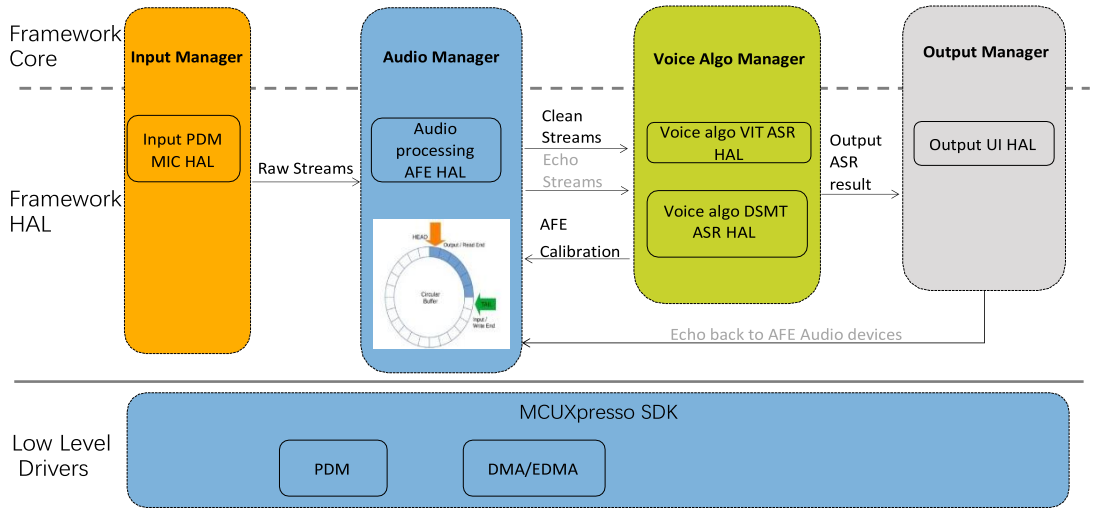

图2. 语音架构

4 基于框架的语音识别示例设计

基于框架的LVGL GUI语音识别示例（以下简称示例）是以示例代码`lvgl_gui_camera_preview_cm7`（请访问https://mcuxpresso.nxp.com/appcodehub）为基础实现的。此示例启用了中英文版的DMST和VIT ASR模型来实现语音识别功能。

为了演示此功能，GUI应用程序的设计如下：

- 添加一个开机后显示的待机画面。一个下拉列表组件提供中英文语言选择。屏幕上的文本以所选语言显示，并且只有当用户以所选语言说出唤醒词时，设备才会响应语音识别。识别唤醒词或触摸屏幕后，即可进入主界面。请参见图3。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

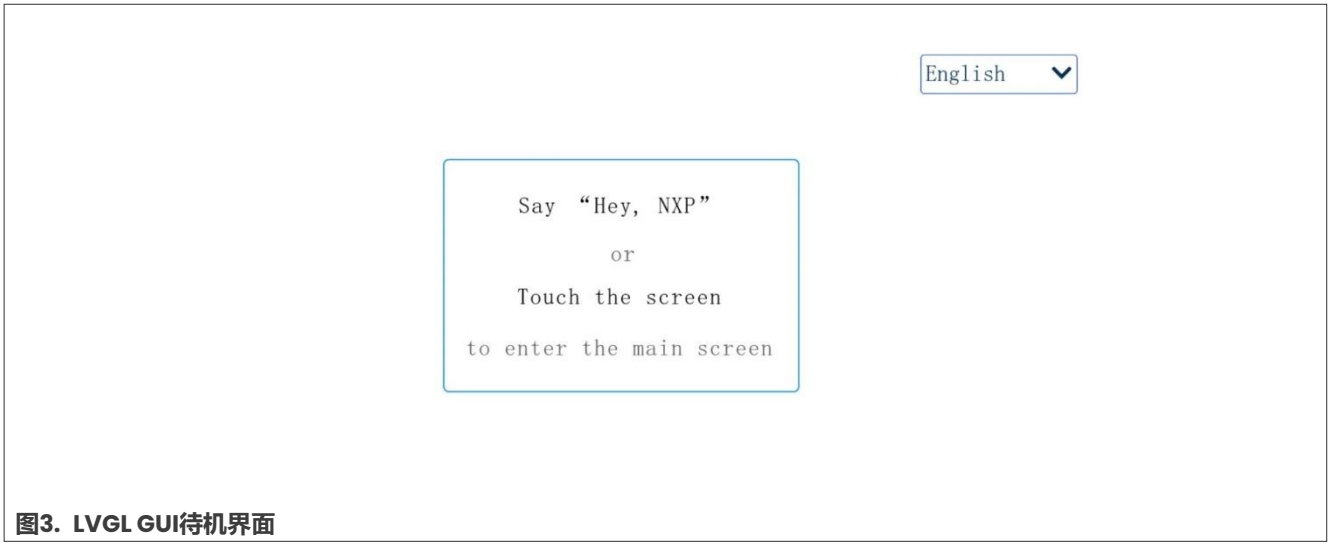

图3. LVGL GUI待机界面

- 同样，在主界面上添加了用于语言选择的下拉列表组件，以实现与待机界面相同的功能。支持的语音命令的提示也会显示在界面上。参见图4。语音命令与触摸具有相同的功能：
  - 说出**registration**或触摸**Registration**按钮，状态标签上会显示**Registration...**提示。
  - 说出**recognition**或触摸**Recognition**按钮，状态标签上会显示**Recognition...**提示。
  - 说出**preview**或触摸按钮以外的其他区域，状态标签上会显示**previewing...**提示。
- 主界面有超时返回功能。也就是说，当在主界面的停留时间达到60秒后，系统会自动返回待机界面。

图4. LVGL GUI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_voice_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。
  - 将框架文件夹替换为GitHub上的复制的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。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

– 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。

5 添加硬件支持

语音识别示例中涉及的硬件为麦克风。

5.1 添加麦克风的驱动程序

添加PDM麦克风接口和DMA支持驱动程序。

1. 将*SDK_2_13_0_MIMXRT1170-EVK\devices\MIMXRT1176\drivers*中的*fsl_dmamux.c*和*fsl_dmamux.h*、*fsl_edma.c*和*fsl_edma.h*、*fsl_pdm.c*和*fsl_pdm.h*、*fsl_pdm_edma.c*和*fsl_pdm_edma.h复制到此示例的drivers文件夹。*

5.2 添加麦克风的板级支持

1. 将*[smart HMI]\coffee_machine\cm7\board\pin_mux.c*中的`BOARD_InitMicPins()`函数复制到此示例的*pin_mux.c*，并在*pin_mux.c*中的`BOARD_InitBootPins()`中调用宏定义`ENABLE_INPUT_DEV_PdmMic`，以进行麦克风引脚设置。
2. 添加代码行 `#include "board_define.h"`，因为上述宏定义`ENABLE_INPUT_DEV_PdmMic`在*pin_mux.c*文件中定义。
3. 将`BOARD_InitEDMA()`函数从*[smart HMI]\coffee_machine\cm4\board\board.c*复制到开发板EDMA初始化示例的*\board.c*中。
4. 在*board.c*中添加以下代码行：

```
#include "fsl_edma.h"
#include "fsl_dmamux.h"
```

5. 在*board.h*中声明`BOARD_InitEDMA()`函数。

6 基于框架实现语音识别功能

6.1 添加语音算法库和引擎

智能HMI解决方案支持远场语音识别的功能，此功能是由基于音素的自动语音识别（ASR）引擎以及静态库提供的数字信号处理（DSP）和音频前端（AFE）来实现的。本示例同样会使用这些技术。下面是添加库和引擎的具体步骤。

1. 将包含DMST和VIT、DSP和AFE的ASR库的文件夹*local_voice*从smart *HMI\coffee_machine\cm7\libs*复制到此示例软件的*libs*文件夹。
   注：*不同语言的VIT语音模型资源也包含在克隆文件夹local_voice的头文件中。它们用于咖啡机应用程序，并需要根据本示例的要求进行修改（具体修改方法将在第6.2.2节中详细介绍）。*
   按照以下步骤对新文件夹进行配置：
   - 在**Project** > **Properties** > **C/C++ Build** > **settings** > **MCU C++ Linker** > **Libraries**中添加相应的库及其搜索路径。

```
VoiceSeekerLight
VIT_CM7_v04_07_07
sln_asr
arm_cortexM7lfdp_math
"${workspace_loc:/${ProjName}/libs/local_voice}"
```


本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

```
"${workspace_loc:/${ProjName}/libs/local_voice/vit/RT1170_CortexM7/Lib}"
```

注： *VoiceSeekerLight库用于AFE。VIT_CM7_v04_07_07库用于VIT模型。sln_asr库用于DSMT模型。arm_cortexM7lfdp_math库用于DSP。*
在**Project > Properties > C/C++ Build > settings > Tool Settings > MCU C compiler > Includes**和**MCU C++ compiler > Includes**中添加库的头文件的搜索路径。

```
"${workspace_loc:/${ProjName}/libs/local_voice/vit/RT1170_CortexM7/Lib}"
"${workspace_loc:/${ProjName}/libs/local_voice/vit/RT1170_CortexM7/Lib/Inc}"
```

2. 将*libs\local_voice*中包含上述算法库的头文件的音频文件夹、驱动AFE算法`VoiceSeeker`的API以及`VoiceSeeker`的公共用程序从*[smart HMI]\coffee_machine\cm7*复制到此示例。并对新文件夹进行如下配置：
   - 取消勾选“**Exclude resource from build**”，右键单击该文件夹并在弹出菜单中选择**Properties**，以将该文件夹构建到工程中。
   - 按照步骤1，在工程设置中添加此文件夹的搜索路径。

```
"${workspace_loc:/${ProjName}/audio}"
"${workspace_loc:/${ProjName}/audio/RDSP_Includes}"
"${workspace_loc:/${ProjName}/audio/rdsp_utilities_public/include}"
 "${workspace_loc:/${ProjName}/audio/rdsp_utilities_public/
rdsp_memory_utils_public}"
```

3. 将含不同语言DSMT模型资源的二进制文件的*local_voice*文件夹，以及使用DSMT和VIT模型相关定义将语音转换为操作命令的引擎从*[smart HMI]\coffee_machine\cm7*复制到此示例。并对新文件夹进行如下配置：
   - 取消勾选“**Exclude resource from build**”，按照上述步骤将此文件夹构建到此工程中。
   - 按照上述步骤在工程设置中添加此文件夹的搜索路径。

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

注： *DSMT模型资源原本用于咖啡机应用程序，需要根据本示例的需求进行修改（将在第6.2.2节中介绍具体内容）。*

6.2 实现语音模型

本示例支持DSMT和VIT两种语音模型。两者均支持英语和中文。但需注意，一个应用程序无法同时支持这两种模型。因此，需要在启用某一模型后重建工程，以便切换到该模型。系统启动时默认语言为英语。以下是实现这两种模型的步骤。

6.2.1 设置模型

模型资源设计包含英文唤醒词“**hey NXP**”和三条语音命令：`registration`、`recognition`和`preview`，以及相应的中文唤醒词“你好 恩智浦”和语音命令“用户注册、人脸识别、预览”。

设置DMST和VIT模型。

- 如需使用第三方工具生成DSMT模型，请参阅《智能HMI软件开发用户指南》（文件MCU-SMHMI-SDUG）。
- 如需使用恩智浦在线工具生成VIT模型，请参阅《智能HMI软件开发用户指南》（文件MCU-SMHMI-SDUG）。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

6.2.2 将DMST和VIT模型集成到此示例中

如前所述，*local_voice*文件夹中包含多语言DSMT模型资源的二进制文件，以及根据应用程序将语音转换为操作命令的引擎。因此，根据当前示例，对该文件夹中的文件进行修改。

1. 使用工具生成的资源来更新示例中的DSMT和VIT模型资源：
   - 对于DSMT，将生成的中文模型二进制文件克隆并重命名为*oob_demo_cn_pack_WithMapID.bin*，然后替换示例中*local_voice**oob_demo_cn*下的文件。生成的英文模型二进制文件也采取同样的操作。
     考虑到兼容性和简化操作，清除本示例不支持的咖啡机应用程序中原有的法语文件*oob_demo_fr_pack_WithMapID.bin*和德语文件*oob_demo_de_pack_WithMapID.bin*中的内容。
   - 对于VIT，使用VIT在线工具生成的文件替换示例*libs**local_voice**vit**RT1170_Cortex M7**Lib*下的*VIT_Model_cn.h*和*VIT_Model_en.h*文件。
     考虑到兼容性和简化操作，在咖啡机应用程序中，声明一个空的德语数组，并在*VIT_Model_de.h*中对旧声明进行注释。
2. 根据此示例，修改已定义的各种操作类型以及相关函数，以将语音命令转换为与咖啡机应用程序相关的*IndexCommands.h*文件中的两种模型操作：
   - 重命名枚举`_coffee_machine_action`，并重新定义语音命令的操作类型，如下所示：

```
enum _voice_rec_demo_action
{
    kVoiceRecDemoActionReg = 0,
    kVoiceRecDemoActionRec = 1,
    kVoiceRecDemoActionPre = 2,
    kVoiceRecDemoActionInvalid
};
```

   - 删除函数`get_cmd_number()`和`get_action_index_from_keyword()`中与示例不支持的`ASR_CMD_USER_REGISTER`实例相关的所有代码行。
   - 在本示例中，将字符串`COFFEE_MACHINE`（区分大小写）替换为`VOICE_REC_DEMO`，将`coffee_machine`（区分大小写）替换为`voice_rec_demo`。
   - 在`get_action_index_from_keyword()`函数中将`kCoffeeMachineActionInvalid`更改为`kVoiceRecDemoActionInvalid`。
3. 按照示例修改*local_voice_model_vit.h*中获取VIT模型的函数：
   - 在本示例中，将字符串`COFFEE_MACHINE`（区分大小写）替换为`VOICE_REC_DEMO`。
   - 删除与示例不支持的`ASR_CMD_USER_REGISTER`实例相关的所有代码行。
4. 使用*IndexCommands.h*中定义的操作类型，重新定义*IndexCommands_vit.h*中针对各种语言的特定操作：
   - 将字符串`coffee_machine`（区分大小写）替换为`voice_rec_demo`，将`COFFEE_MACHINE`（区分大小写）替换为`VOICE_REC_DEMO`以及将`CoffeeMachine`替换为`VoiceRecDemo`。
   - 细化英语数组*action_voice_rec_demo_en*、中文数组*action_voice_rec_demo_cn*、*德语数组 action_voice_rec_demo_de*（不支持）以及法语数组*action_voice_rec_demo_fr*（支持）。例如：

```
unsigned int action_voice_rec_demo_en[] =
  { kVoiceRecDemoActionInvalid,  // unknown
  kVoiceRecDemoActionReg,    // "registration"
  kVoiceRecDemoActionRec,    // "recognition"
  kVoiceRecDemoActionPre,    // "preview"
};
unsigned int action_voice_rec_demo_de[] = {
```


本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

```
  kVoiceRecDemoActionInvalid,     // unknown
};
```

注：*对用于DSMT模型的IndexCommands_dsmt.h文件也进行相同的修改。*

5. 修改DSMT模型所用的中文*IndexToCommand_cn.h*、英文*IndexToCommand_en.h*、德语*IndexToCommand_de.h*和法语**IndexToCommand_fr.h**文件：
   - 根据设计好的命令，重新定义中英文唤醒词和语音命令（可参考DMST工具生成的ww.txt和cmd_voice_rec_example.txt文件中的命令），例如英文唤醒词的定义。

```
char *ww_en[] = {"Hey NXP"};
```

删除此示例中所有不支持的德语和法语命令的定义。例如，法语唤醒词的定义如下。

```
char *ww_fr[] = {};
```

   - 将所有文件中的字符串`coffee_machine`替换为`voice_rec_demo`。
6. 考虑到兼容性，在*hal_voice_algo_asr_local.h*文件的`enum _asr_inference`中添加以下命令类型的定义，同时保留之前的定义。

```
ASR_CMD_VOICE_REC_DEMO = (1U << 1U)
```

6.3 更新并启用语音相关的HAL

为了驱动语音识别，需要使用以下HAL驱动程序：在*hal_input_pdm_mic.c*中实现的PDM麦克风HAL、在*hal_audio_processing_afe.c*中实现的音频处理AFE HAL、在*hal_voice_algo_dsmt_asr.c*中实现的DSMT语音算法HAL、以及在*hal_voice_algo_vit_asr.c*中实现的VIT语音算法HAL。

这些HAL中实现的功能对于不同的应用程序是通用的。虽然需要进行少量修改，但主要是对示例进行一些配置：

1. PDM麦克风HAL驱动程序的*hal_input_pdm_mic.c*文件无需修改，但需要在*board_define.h*中添加定义来启用它。

```
#define ENABLE_INPUT_DEV_PdmMic
```

2. 更新音频处理AFE HAL驱动程序：
   - 在*hal_audio_processing_afe.c*文件中声明全局变量`g_MQSPlaying`，该变量在音频播放相关的HAL中声明，而本示例中不支持该变量，并将该变量默认设置为false。参见下文。

```
#ifdef ENABLE_OUTPUT_DEV_MqsAudio || ENABLE_OUTPUT_DEV_MqsStreamerAudio
extern volatile bool g_MQSPlaying;
#else
volatile bool g_MQSPlaying = false;
#endif
```

   - 在*board_define.h*文件中添加*hal_audio_processing_afe.c*文件中使用的缓冲区的内存分区定义。

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


本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

- 在*board_define.h*中添加定义以启用音频处理HAL驱动程序。

```
#define ENABLE_AUDIO_PROCESSING_DEV_Afe
```

3. 更新DSMT和VIT语音算法HAL驱动程序。
   - 添加定义，以使用当前支持的以下应用程序。

```
#if ENABLE_VOICE_REC_DEMO
#define CURRENT_DEMO ASR_CMD_VOICE_REC_DEMO
```

   - 在*hal_voice_algo_dsmt_asr.c*文件的voice_algo_dev_input_notify()函数中，添加当前应用程序的设置。

```
#if ENABLE_VOICE_REC_DEMO
s_AsrEngine.voiceConfig.demo = ASR_CMD_VOICE_REC_DEMO;
```

   - 在hal_voice_algo_asr_local.h文件中重新定义DSMT模型当前使用的语言，因为仅支持两种语言，而示例中的DSMT模型设置了四种语言。

```
#ifdef ENABLE_DSMT_ASR
#define DEFAULT_ACTIVE_LANGUAGE (ASR_ENGLISH | ASR_CHINESE)
```

   - 为了配置DSMT和VIT ASR及AFE HAL，请在board_define.h文件中添加相关定义。
     – 添加定义以启用两种算法HAL（目前默认启用DSMT ASR）：

```
#define ENABLE_DSMT_ASR
//#define ENABLE_VIT_AS
```

     – 添加定义以在上述启用的控制下配置DSMT、VIT和AFE的某些功能。例如，最大唤醒字长度。

```
#elif defined(ENABLE_DSMT_ASR)
/* "Hey NXP" and its corresponding translations in other languages may
 take up to 3s to be spoken. */
#define WAKE_WORD_MAX_LENGTH_MS 3000
```

     – 配置用于DSMT和VIT ASR算法HAL的SRAM内存分区。

```
#define AT_CACHEABLE_SECTION_ALIGN_OCRAM(var, alignbytes) \
__attribute__((section(".bss.$SRAM_OC1,\"aw\",%nobits @"))) var
 __attribute__((aligned(alignbytes)))
```

   - 更新**Project > Properties > C/C++ Build > MCU Settings**中的内存设置，将上述SRAM_OC1的大小扩大到0x10000，并删除**SRAM_OC2**的设置，因为其空间已并入**SRAM_OC1**。
   - 在工程设置中添加上述语音HAL的搜索路径。

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

6.4 添加并更新输出用户界面HAL

输出用户界面HAL取决于LVGL GUI应用程序。它负责将事件通知给语音算法HAL，并根据语音算法HAL的推理结果进行响应。使用GUI应用程序时，这些事件通常由GUI触发，其结果会显示在GUI上。

要启用HAL，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：

1. 人脸算法触发和结果处理，具有以下功能，包括进度条、人脸指示器（预览模式）。
2. 音频播放，支持多种语言。
3. 咖啡机相关应用程序，包括GUI。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

4. 语音算法触发和结果处理，支持多种语言。
5. 带有会话定时器的待机和唤醒机制。
6. 其回调函数是GUI应用程序调用的API，用于与输出用户界面HAL的通信。
7. 从LVGL GUI应用程序调用API与LVGL GUI应用程序通信。

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1、2和3相关的代码。
- 保留并更新上述4、5、6和7。

主要步骤如下：

1. 克隆*hal_output_ui_coffee_machine.c*文件，将文件名更改为*hal_output_ui_ voice_rec.c*（以下更新均针对此文件）。
2. 将文件中的所有`CoffeeMachine`字符串替换为`VoiceRecDemo`。
3. 删除与视觉（人脸识别）、含`prompt`字符串的音频和含图标及进度的人脸指示符的相关代码。例如：

```
#include "hal_vision_algo.h", #include "hal_event_descriptor_face_rec.h"
```

4. 删除包含与咖啡机应用程序相关的GUI的代码。例如，与咖啡类型注册有关的变量：`s_IsWaitingAnotherSelection`和`s_IsWaitingRegisterSelection`。
5. 将`s_OutputDev_UiVoiceRecDemo`结构中体的`.attr.pSurfac`成员设置从`&s_UiSurface`更改为`NULL`，因为它用于人脸识别。
6. 更新示例的枚举类型中的语音命令类型。

```
enum
{
    VOICE_CMD_REG = 0,
    VOICE_CMD_REC = 1,
    VOICE_CMD_PRE = 2,
    VOICE_CMD_INVALID
};
```

7. 由于实例仅使用了单核– CM7，因此在`_SetVoiceModel()`和`_StopVoiceCmd()`函数中，将输出事件结构体中的**eventInfo**成员设置从`kEventInfo_Remote`更改为`kEventInfo_Local`。
8. 更新`_InferComplete_Voice()`函数，以便对语音识别的推理结果采取相应的操作：
   - 保留用于主界面和待机界面的界面ID `kScreen_Home`和`kScreen_Standby`，同时删除其他界面ID。
   - 添加上述步骤中定义的`VOICE_CMD_REG`、`VOICE_CMD_REC`和`VOICE_CMD_PRE`三种语音命令类型，作为语音推理结果的三种情况，同时删除`kScreen_Home`下的旧命令。
   - 添加API `gui_show_voice_rec_action()`的调用，以便在GUI界面上显示语音推理结果。LVGL GUI应用程序提供的API在*custom.c*中实现（将在第7节介绍）。
9. 将`WakeUp()`函数中的宏定义`ASR_CMD_COFFEE_MACHINE`更改为`ASR_CMD_VOICE_REC_DEMO`。
10. 更新用户界面回调函数，以处理GUI触发的事件。
    - 更新用于初始化的函数`UI_EnterScreen_Callback()`，包括会话定时器设置、进入界面时的语音模型设置等，如下所示。
      – 保留主界面和待机界面的界面ID `kScreen_Home`和`kScreen_Standby`，同时删除其他界面ID。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

- 删除不需要的`UI_Finished_Callback()`函数。
- 新增`UI_GetLanguage_Callback()`函数，将当前识别的语言索引转换为GUI应用程序所需的语言索引。

11. 继续使用GUI应用程序的API：`get_current_screen()`、`gui_set_home()`和`gui_set_standby()`（这些需要在*custom.c*中实现，并会在第7节中详细介绍），同时删除其他API。
12. 添加代码行`#include "custom.h"`，以使用与GUI相关的API。

```
#define ENABLE_OUTPUT_DEV_UiVoiceRecDemo
```

13. 在*board_define.h*中添加以下定义来启用人机界面输出HAL。

注：*由于HAL与应用程序密切相关，可根据应用程序设计对输出用户界面HAL进行一些优化。*

6.5 更新显示HAL

在用于显示HAL的*hal_display_lvgl_camerapreview.c*文件中，摄像头预览功能可随时启用。然而，示例中的主界面需要摄像头预览，而待机界面则不需要。因此还需在`HAL_DisplayDev_LVGLCameraPreview_Blit()`函数中禁用待机界面上的摄像头预览。

7 实现LVGL GUI应用程序

基于框架的LVGL GUI应用程序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。

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

以下是实现的介绍，仅供参考：

1. 自定义代码在GUI Guider提供的*custom.c*实现，而*custom.h*中作为GUI Guider工程与嵌入式系统工程之间的接口。
   - 在*custom.c*文件中，实现函数`get_current_screen()`、`gui_set_standby()`和`gui_set_home()`以及相关子函数，如`gui_setup_screen()`，以设置主界面和待机界面（可参考咖啡机应用程序中的函数）。
   - 在*custom.c*文件中，实现`gui_show_voice_rec_action()`函数，将结果显示在GUI界面上，作为对按钮事件和语音命令的响应。
   - 在*custom.c*文件中，实现GUI应用程序的多语言支持：
     – 添加函数`gui_standby_set_language_UI()`和`gui_home_set_language_UI()`，以便在待机和主界面上显示所选语言的所有文本。据此，定义所有中英文的文本字符串。例如，主界面上的标题定义如下。

```
static const char *s_HomeTitleStr[kLanguage_Ids][1] = {
     {"Camera preview",},
     {"相机预览",},
};
```

     – 添加函数`gui_standby_language_changed_cb()`和`gui_home_language_changed_cb()`，以响应通过单击待机界面和主界面上的下拉组件触发的语言选择事件。它们在event.c文件中被调用以进行事件处理。
   - 用户界面回调函数和输出用户界面HAL中实现的`WakeUp()`函数取决于嵌入式平台。同时，还需要在GUI Guider工程中调用它们，以便在模拟器上运行，而模拟器与嵌入式平台无关。因此，为了确保与嵌入式平台和GUI Guider模拟器的兼容性，需要进行以下更新。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

– 在*custom.c*中，通过宏定义`#ifdef LV_USE_GUIDER_SIMULATOR`的控制，实现具有相同原型的另一组函数（可能更简单），以实现兼容性。例如，`UI_EnterScreen_Callback()`函数的实现方式如下。

```
#if LV_USE_GUIDER_SIMULATOR
uint8_t UI_EnterScreen_Callback(screen_t screenId)
{
  return 1;
}
```

– 上述宏定义`LV_USE_GUIDER_SIMULATOR`最初是在GUI Guider的lvgl-simulator文件夹中*lv_conf.h*文件中定义并设置为1的。将该文件复制到示例软件的source文件夹下，并将定义设置为0，以在嵌入式平台上禁用它。

- 更新*custom.h*文件，以声明全局类型和函数：
  – 定义与语音识别操作索引、语言ID和界面ID相关的枚举类型。
  – 将输出用户界面HAL的枚举类型`_wake_up_source`移至*custom.h*，因为*event.c*文件也使用了枚举中定义的按钮类型。
  – 声明所有全局函数，如`UI_xxx_Callback()`和`gui_xxx()`。

2. 在GUI Guider上开发GUI：
   - 克隆基础软件包`lvgl_gui_camera_preview_cm7`的`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`voice_rec`。
   - 将上述更新的custom.c和custom.h文件复制到新的GUI Guider工程软件中。
   - 打开GUI Guider上的新`voice_rec`工程。按照第4节开头介绍的设计进行更新。并添加以下各种事件处理：
     – 使用API实现GUI Guider上的各种事件处理的自定义C代码，例如`WakeUp()`，点击待机界面时可进入主界面。
3. 将GUI Guider生成的代码更新到MCUXpresso工程。
   - 将生成的文件夹中（images文件夹除外）的*.c*和*.h*文件替换为GUI Guider工程软件生成的文件夹中的相应内容。
4. 添加GUI图像支持
   示例中的GUI图像与克隆的GUI应用程序相同。然而，图像的名称和描述符会随着示例中使用的新版GUI Guider而改变。因此，需要在克隆的版本的基础上进行一些更新：
   - 将基础软件包*lvgl_gui_camera_preview_cm7*中的图像资源文件夹*resource_lvgl_gui_camera_preview*和工具文件夹*resource_build*克隆到示例软件中。然后删除生成的文件*resource_information_table.txt*和*camera_preview_resource.bin*。
   - 对于新示例，将名称中的所有*camera_preview*字符串和其余文件的内容更改为*voice_rec*。
   - 将图像文件*_NxpVoiceRec_alpha_185x55.c*和*_NxpFaceRec_alpha_185x55.c*从GUI Guider工程软件中的*generated\images*路径复制到*resource_lvgl_gui_voice_rec\images*路径。删除旧文件*_NxpVoiceRec_185x55.c*和*_NxpFaceRec_185x55.c*。
   - 相应地，将*voice_rec_resource.txt*文件中的文件名更改为*_NxpVoiceRec_alpha_185x55.c*和*_NxpFaceRec_alpha_185x55.c*。
   - 在windows中双击执行脚本文件*voice_rec_resource_build.bat*，以构建图像资源。生成二进制文件*voice_rec_resource.bin*和信息文本文件*resource_information_table.txt*。
   - 图像资源的大小和地址信息无需更新到示例软件中，因为图像数据未发生改变。因此仅需更新图像描述符数组。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

因此，仅使用*_NxpVoiceRec_alpha_185x55.c*和*_NxpFaceRec_alpha_185x55.c*文件中的图像描述符数组*_NxpFaceRec_alpha_185x55*和*_NxpVoiceRec_alpha_185x55*，就可以更新*setup_images.c*文件中的相同数组。

8 添加应用级支持

应用程序级别的代码和配置都位于*source*文件夹中。主要更新通常在主文件**lvgl_gui_voice_rec_cm7.cpp**中进行，其中包含了硬件电路板的初始化和框架设置。

8.1 更新电路板初始化

要更新电路板初始化，请遵循以下步骤：

1. 在`APP_BoardInit()`中调用`BOARD_ConfigMPU()`函数用于内存MPU的配置。

8.2 更新框架设置

要基于框架设置已启用的语音算法HAL和用户界面输出HAL及其管理器，请遵循以下开发转换步骤：

1. 包含与启用的HAL管理器相关的头文件，如下所示：

```
#include "fwk_input_manager.h"
#include "fwk_audio_processing.h"
#include "fwk_voice_algo_manager.h"
#include "fwk_output_manager.h"
```

2. 声明已启用的HAL设备：

```
HAL_INPUT_DEV_DECLARE(PdmMic);
HAL_AUDIO_PROCESSING_DEV_DECLARE(Afe);
HAL_VOICEALGO_DEV_DECLARE(Asr);
HAL_VOICEALGO_DEV_DECLARE(Asr_VIT);
HAL_OUTPUT_DEV_DECLARE(UiVoiceRecDemo);
```

3. 在`APP_RegisterHalDevices()`函数中注册已启用的HAL设备：

```
  HAL_INPUT_DEV_REGISTER(PdmMic, ret);
  HAL_AUDIO_PROCESSING_DEV_REGISTER(Afe, ret);
#ifdef ENABLE_DSMT_ASR
  HAL_VOICEALGO_DEV_REGISTER(Asr, ret);
#elif defined(ENABLE_VIT_ASR)
  HAL_VOICEALGO_DEV_REGISTER(Asr_VIT, ret)
#endif /* ENABLE_DSMT_ASR */
  HAL_OUTPUT_DEV_REGISTER(UiVoiceRecDemo, ret);
```

4. 在`APP_InitFramework()`函数中初始化相关管理器：

```
FWK_MANAGER_INIT(InputManager, ret);
FWK_MANAGER_INIT(AudioProcessing, ret);
FWK_MANAGER_INIT(VoiceAlgoManager, ret);
FWK_MANAGER_INIT(OutputManager, ret);
```

5. 在**APP_StartFramework()**函数中启动相关管理器：

```
FWK_MANAGER_START(InputManager, INPUT_MANAGER_TASK_PRIORITY, ret);
FWK_MANAGER_START(AudioProcessing, AUDIO_PROCESSING_TASK_PRIORITY, ret);
FWK_MANAGER_START(VoiceAlgoManager, VOICE_ALGO_MANAGER_TASK_PRIORITY, ret);
FWK_MANAGER_START(OutputManager, OUTPUT_MANAGER_TASK_PRIORITY, ret);
```


本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

6. 定义上述步骤中任务管理器的优先级。

```
#define INPUT_MANAGER_TASK_PRIORITY          2
#define AUDIO_PROCESSING_TASK_PRIORITY       2
#define VOICE_ALGO_MANAGER_TASK_PRIORITY     3
#define OUTPUT_MANAGER_TASK_PRIORITY         1
```

***注：**有关上述所有修改的更多详细信息，请参阅随附的示例软件：https://mcuxpresso.nxp.com/appcodehub。*

9 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中打开示例工程。构建*.axf*文件并将其烧录到`0x30000000`地址，同时将资源二进制文件*voice_rec_resource.bin*烧录到`0x30800000`地址。

LVGL GUI语音识别示例正常工作流程如下：

- **Standby screen（待机界面）**
  开机后，待机界面如图3所示。根据所选语言的提示，说出唤醒词或触摸屏幕进入主界面，可通过界面上的下拉菜单对语言进行更改。
- **Home screen（主界面）**
  进入主界面后，摄像头捕获的视频流会显示在GUI界面上的摄像头预览区域。作为初始状态，预览状态显示在状态标签上，并提示支持的语音命令。这些命令将以当前语言显示，可通过下拉小部件进行更改。详见图4。
  – **Preview（预览）：**每次单击按钮和图像以外的区域，或根据当前选择的语言说出语音命令“`preview`或预览”时，状态标签上会显示相应的文本“**Preview...**或预览...”。
  – **Registration（注册）：**每次单击**Registration**按钮，或根据当前选择的语言说出语音命令“`registration`或用户注册”时，状态标签上会显示相应的文本“**Registration...**或注册...”。
  – **Recognition（识别）：**每次单击**Recognition**按钮，或根据当前选择的语言说出语音命令“`Recognition`或人脸识别”时，状态标签上会显示相应的文本“**Recognition...**或识别...”。
  当主界面停留时间达到60秒时，系统将自动返回到待机界面。

10 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。

本文件中提供的所有信息均受法律免责声明的约束。

© 2024 NXP B.V. 版权所有。

本软件由版权所有者和参与者“按原样”提供，不承担任何明示或暗示的担保责任，包括但不限于对适销性和特定用途适用性的暗示保证。在任何情况下，无论因何种原因或根据何种法律条例，版权所有者或参与者均不对因使用本软件而导致的任何直接、间接、偶然、特殊、惩戒性或后果性损害（包括但不限于采购替代商品或服务；使用损失、数据损失或利润损失或业务中断）承担责任，无论是因合同、严格责任还是侵权行为（包括疏忽或其他原因）造成的，即使事先被告知有此类损害的可能性也不例外。

11 修订历史

表1总结了本文档的修订情况。

表1. 修订历史

文档ID	发布日期	说明
AN14250 v.1	2024年3月26日	首次公开发布

© 2024 NXP B.V. 版权所有。

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.

© 2024 NXP B.V. 版权所有。

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

i.MX — is a trademark of NXP B.V.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

目录

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2024 NXP B.V. All rights reserved.

For more information, please visit: https://www.nxp.com.cn

Date of release: 26 March 2024
Document identifier: AN14250