# AN14276

## i.MX的Voice Seeker和VoiceSpot

**第1.0版—2024年6月24日** **应用笔记**

**文档信息**

| 信息 | 内容 |
| --- | --- |
| 关键词 | AN14276、语音处理、VoiceSeeker、VoiceSpot |
| 摘要 | 恩智浦创建了一个语音处理环境，并不断改进。这个环境使客户能够以低延迟、低误报率、低能耗和最少的操作将其用于他们的应用程序。 |

# 1 介绍

恩智浦创建了一个语音处理环境，并不断改进。这个环境使客户能够以低延迟、低误报率、低能耗和最少的操作将其用于他们的应用程序。

VoiceSeeker和VoiceSpot是恩智浦提供的两个语音处理工具，它们可以与大多数i.MX 8M系列搭配使用，并且CPU使用率较低。

# 2 目的

本文介绍了VoiceSeeker，这是一个提供高分辨率波束成形和多通道声学回声消除（AEC）的库，不需要预定的固定麦克风几何形状，这意味着板上设计更具灵活性。

除了VoiceSeeker之外，本文还介绍了VoiceSpot，这是一个不需要重新编译即可与不同模型一起工作的语音检测引擎。VoiceSeeker和VoiceSpot完美契合，因为两者都是为协同工作而设计的。VoiceSeeker清除音频中的噪声，清除完噪声后，将缓冲区发送给VoiceSpot，后者只专注于检测唤醒词。如果检测到唤醒词，VoiceSpot会向VoiceSeeker返回反馈。

本文的主要目标如下所示：

- 帮助读者充分了解VoiceSeeker、VoiceSpot和AFE。
- 介绍如何配置VoiceSeeker和VoiceSpot及其依赖项。
- 让读者能够区分默认配置和外部配置之间的差异。
- 帮助读者了解校准进程。

# 3 概述

语音辅助的第一阶段是能够倾听语音。尽管这是语音辅助的第一阶段，但它非常重要，因为用于触发语音辅助的缓冲区与后续步骤中使用的缓冲区是相同的。这意味着缓冲区应该被清理干净，并且只包含人类的声音特征，以便准确检测唤醒词和解析命令。VoiceSeeker的主要目标是清理输入的麦克风数据，而VoiceSpot则用于检测唤醒词并触发语音辅助的下一阶段。

VoiceSeeker是一个处理麦克风输入数据的库，其中一个进程是去除麦克风捕获数据中不需要的噪声。VoiceSeeker的核心是一个静态库，具有处理音频的API。Imx-voiceui工程在这个静态库上增加了一个新层，使其更易于使用。这一层与静态库的结合产生一个共享对象。从现在开始，当谈论VoiceSeeker时，指的是这个共享对象。

VoiceSeeker库预装在大多数恩智浦Linux发行版中，因此您可以快速体验其部分功能。VoiceSeeker是一个库，这意味着它需要一个可以在运行时加载它的应用程序。这个应用程序称为AFE，它也预装在系统中。

AFE是一个允许在运行时选择所需语音处理引擎的程序，例如选择VoiceSeeker或Conversa。后续章节提供了AFE的概述，重点介绍加载VoiceSeeker库的用例。有关AFE的更多用例，请参见GitHub上的TODO.md文件。

`voice_ui_app`是恩智浦语音处理环境中的另一个程序。`voice_ui_app`等待VoiceSeeker的输出，并搜索是否有任何语音特征。语音特征可以是唤醒词或语音命令，对每个特征的处理方式都不同。

首先，它在VoiceSpot的帮助下查找唤醒词，如果VoiceSpot找到了唤醒词命令，它就能够触发第三方应用程序，使第三方开始监听捕获流。VoiceSpot还将缓冲区发送给VIT，以便查找语音命令。如果VoiceSpot无法找到唤醒词，它只会将缓冲区发送给VIT，而不会向第三方应用程序发送任何类型的信号。本文仅介绍了从使用AFE和VoiceSeeker进行麦克风捕获，到在`voice_ui_app`中使用VoiceSpot检测唤醒词的工作流程。有关VoiceSpot API的详细信息，请参阅其相关文档。

Voice Intelligent Technology（VIT）是恩智浦另一款处理语音命令的产品。它不在本文的范围内，但可以与VoiceSeeker和VoiceSpot集成（见第5.1.4节）。如需了解更多关于VIT的信息，请访问其网站或参阅《i.MX RT芯片VIT入门指南：Voice Intelligent Technology SDK演示》（文档IMXRTVITGSUG）。

# 4 材料

本节介绍了正确执行本文所示示例所需的硬件和软件要求。

## 4.1 软件要求

运行本文中所述的示例需要以下软件：

- i.MX Linux BSP v6.6.23或更高版本（推荐版本或其他版本可从https://www.nxp.com.cn/IMXLINUX下载）。
- Audacity或类似软件。

有关如何烧写电路板并准备运行BSP的更多信息，请参阅《i.MX Linux用户指南》（文档IMXLUG）。

## 4.2 硬件要求

运行本文所示示例需要以下硬件：

- i.MX 8MP EVK板。
- 8MIC-RPI-MX8电路板。
- 带AUX连接的扬声器。
- USB-A转USB-C数据线。
- USB-A转USB-B Micro数据线。
- 3.5mm插孔音频数据线。

如果您对硬件设置有任何疑问，请参阅《i.MX 8M Plus EVK快速入门指南》，该入门指南介绍了如何下载软件并启动电路板。

# 5 语音UI框架

本节介绍了语音UI框架。

## 5.1 集成到恩智浦Linux BSP

本文中提到的所有程序和二进制文件都随每个BSP版本一起提供。然而，其中一些库或程序仅供评估使用。例如，VoiceSeeker仅配备麦克风波束成形功能，允许使用VoiceSpot检测唤醒词。完整版的VoiceSeeker还具有AEC（声学回声消除）和DOA（到达方向）功能。

这些功能在开发语音应用程序时非常有用。如果您对测试完整版感兴趣，请发送电子邮件至voice@nxp.com，以获取更多信息。

### 5.1.1 架构

VoiceSeeker和VoiceSpot可以彼此独立工作，但建议搭配使用，以实现每个库的最佳性能。VoiceSeeker需要指示唤醒词是否被激活的反馈，以便根据此信息进行调整。这种反馈由VoiceSpot在检测到唤醒词的那一刻提供。如果VoiceSeeker单独工作，可能需要实现这样的机制，以在波束成形方面达到最佳性能。这样做的原因是每个库在单独的进程中运行，VoiceSeeker需要反馈才能正常工作。如果VoiceSeeker和VoiceSpot一起运行，那么这种机制已经在它们所依赖的库中实现了。

Linux版本中包含的二进制文件演示了一个用例，其中一个小型语音助手可以在本地处理一定数量的命令。这个小型助手不依赖互联网连接便可工作。信号的清理由VoiceSeeker完成。VoiceSpot负责搜索唤醒词，VIT对语音命令进行解码和处理。VIT库也在`voice_ui_app`中运行。

本文介绍的完整用例的架构如图1所示。

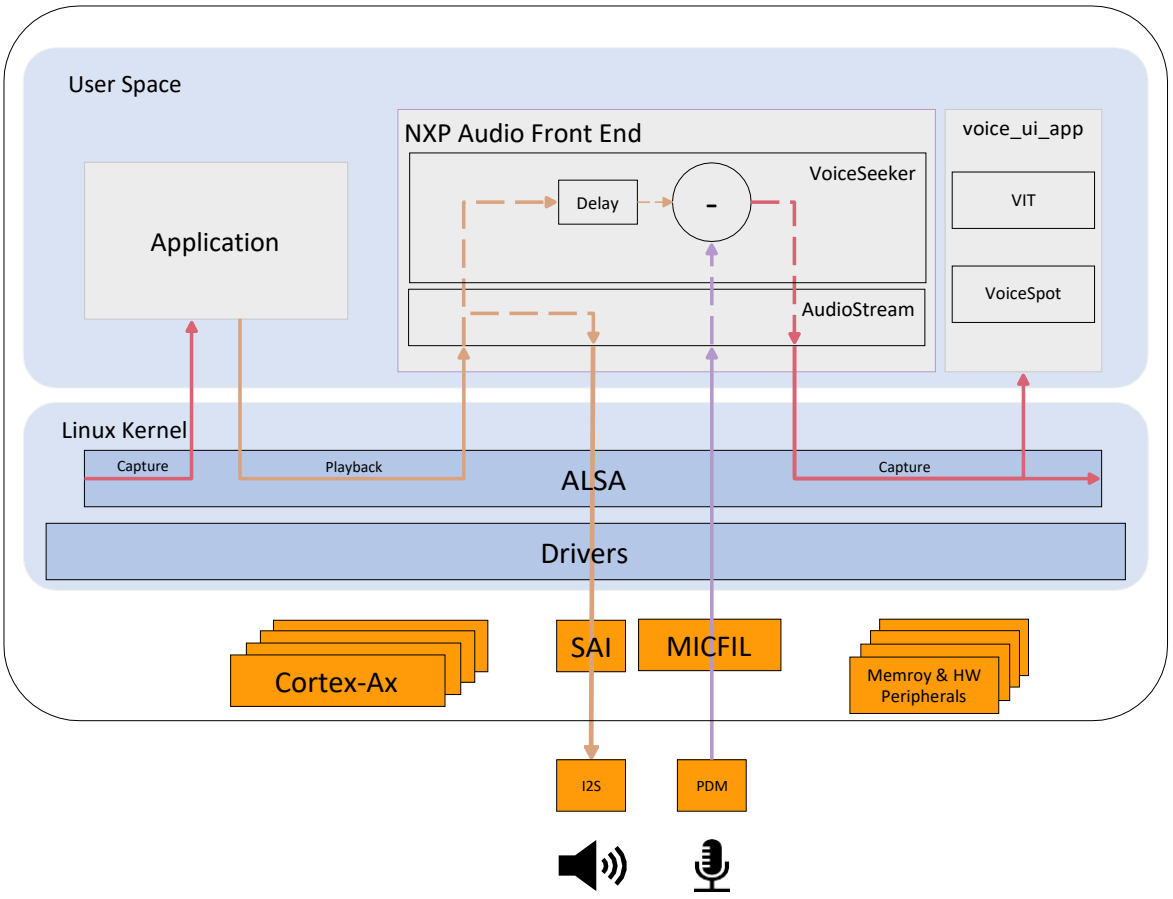


**图1. 用例结构框图**

另一个独立的应用程序必须使用默认的ALSA设备，负责启动播放流，以便AFE可以管理传入和传出的数据。假设asound.conf文件的配置与AFE兼容。因此，必须先启动voice_ui_app，然后才能使用VoiceSpot和VIT库进行其他操作。AFE必须在其之后启动。AFE动态加载VoiceSeeker。最后，可以启动播放流。另一方面，如果asound.conf文件没有兼容的配置（AFE将无法打开其设备），AFE会发出错误消息，并停止执行。本文稍后将介绍兼容的asound.conf文件。

当VoiceSeeker（在AFE上运行）完成其进程时，VoiceSpot会收到通知，在VoiceSeeker的输出中查找唤醒词，即仅包含人类语音特征的缓冲区。最后，如果找到唤醒词，VoiceSpot可以通知第三方应用程序处理语音命令，并向VoiceSeeker提供反馈以调整波束成形。由于VoiceSpot是此流程中的最后一个进程，它必须第一个进行初始化，甚至在AFE之前进行。此用例或任何相似用例的初始化过程如下：

1. 使用voice_ui_app或其他软件启动VoiceSpot共享库。
2. 使用VoiceSeeker启动AFE进行AEC和波束成形（仅在voice_ui_app完成初始化后进行）。
3. 运行上层应用程序（任何播放音频的应用程序，例如aplay、gst_launche、Alexa）。

#### 5.1.1.1 外部参考信号

当AFE之外没有额外的后处理时（AFE处理的数据与扬声器上播放的数据相同），或者当它并非实时系统（对时间要求严格）时，上述架构是有用的。在这些情况下，系统架构可能需要进行一些变更。AFE也可以处理这些情况。

初始化过程与上述用例相似，除非使用RTOS进行实时处理。在这种情况下，请确保RTOS已启动并运行，并在Linux端公开一个声卡。在Linux上，按照上述方式运行VoiceSpot和AFE。这是一个多核应用程序，需要一个实时操作系统来管理播放，需要另一个操作系统来处理语音命令。

本文介绍了这两种场景（环回模式和外部模式）。外部配置是指播放的音频并非在用户空间中生成。音频源来自声卡，而非用户空间的应用程序。

### 5.1.2 恩智浦AFE

恩智浦AFE是i.MX Linux BSP的音频流管理模块。它可以加载并执行特定的语音算法（作为命令行参数选择）。它管理ALSA音频卡与算法的输入和输出缓冲区之间的连接。

AFE是作为语音处理库解决方案而创建的，您必须控制系统的音频输入和输出。在VoiceSeeker的特定情况下，输出信号（通过扬声器听到的信号）必须存储为参考信号，以滤除噪声。在进入更深层次的进程之前，必须清理传入信号（由麦克风捕获的包含噪声、参考信号和人类语音的信号）。这两个信号由VoiceSeeker使用，并生成一个仅包含来自麦克风的人类语音的第三个缓冲区。

AFE将流控制从语音处理阶段中移除。这意味着您可以有一个仅操作缓冲区的处理阶段，并将音频流留给AFE处理。AFE在所需缓冲区准备好处理时调用这些进程。就VoiceSeeker而言，当麦克风和参考信号的缓冲区准备就绪时便调用。

最后，AFE处理语音进程的输出缓冲区，并将其发送到默认的ALSA设备，任何其他客户端都可以监听并等待清洁的语音信号。一些客户端示例包括VoiceSpot、VIT、MS Teams、Alexa等。

AFE仅控制流。它甚至不修正流之间的延迟（声学延迟）。请记住这一点，因为您可能需要一种机制来解决这种延迟（本文后面将介绍声学延迟的解决方案和解析）。

AFE本身不进行任何音频处理。它需要一个额外的库来处理信号。它可以加载任何使用其API的库，并位于`/usr/lib/nxp-afe`文件夹中。如果库遵循这两个规则，AFE就可以加载它并与之搭配工作。要告知AFE使用哪个引擎，提供其名称作为参数（仅名称，而不是完整路径或扩展名）即可。有关如何将AFE与不同库一起使用的更多信息，请参阅TODO.md文件。

#### 5.1.2.1 asound.conf

AFE依赖asound.conf文件。在这个文件中，定义了ALSA（高级Linux声音架构）设备，并且可以通过ALSA API（应用程序编程接口）打开这些设备。AFE使用这些API来控制数据流，供给语音引擎和ALSA默认捕获设备。

i.MX Linux BSP包含每个电路板的`asound`文件。这些文件位于`/unit_tests/nxp-afe/asound.conf_<platform>`，并且可以在硬件不同的情况下进行定制。每个电路板都有不同的`asound.conf`文件，因为每个电路板使用不同的播放编解码器。

在每个文件中，有六个主要设备。其中四个设备由AFE控制，其他两个设备是主要的默认播放和捕获设备，应用程序应该写入（`pwloop`）和读取（`crloop`）。必须通过扬声器听到的音频应写入`pwloop`。为了捕获语音特征，应用程序必须使用`crloop`。其他设备由AFE管理。其中两个是实际的物理源和接收音频设备。一个通过扬声器（`spk`）播放，另一个捕获麦克风（`mics`）的输入信号。最后两个是虚拟设备。一个充当接收器，另一个充当音频源。`prloop`是一个接收器，用于捕获应用程序试图播放的音频。它复制缓冲区并将其写入扬声器。同样，`cwloop`是一个虚拟设备，作为应用程序的音频源，试图捕获语音特征。AFE将这个缓冲区传递给VoiceSeeker，后者使用此缓冲区去除来自扬声器的数据（使用刚刚复制的扬声器数据）。当该进程完成时，它返回一个包含清洁音频的新缓冲区。AFE通过`cwloop`将数据传播到默认捕获设备（`crloop`）。

图2所示为通过AFE的数据流：

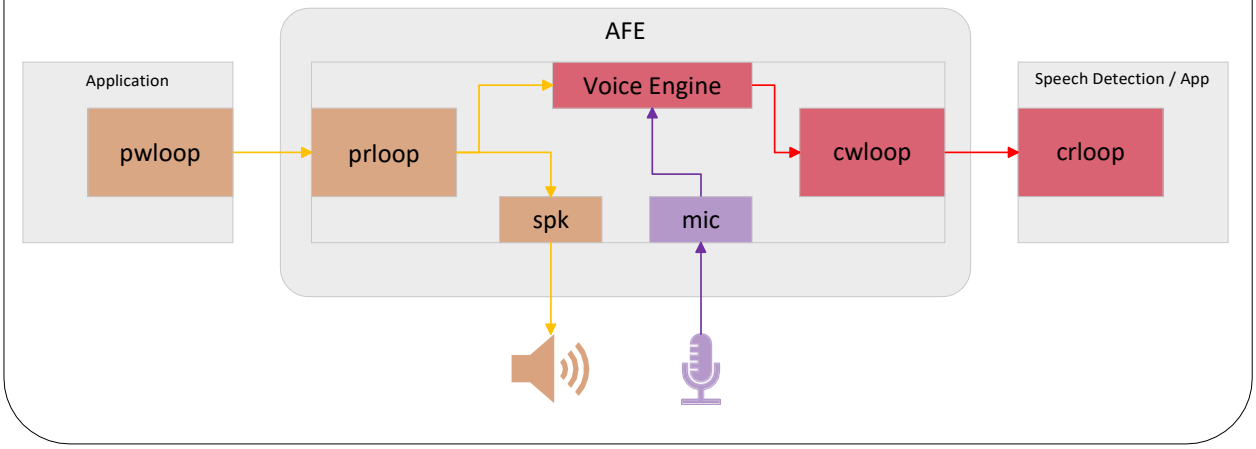


**图2. 通过AFE的数据流**

表1列出了这六个设备的属性。

表1. 设备属性

| 设备名称 | 管理者 | 访问 | 类型 | 流程 |
|---|---|---|---|---|
| pwloop | 应用程序 | 写入 | 环回 | 播放 |
| prloop | AFE | 读取 | 环回 | 播放 |
| spk | AFE | 写入 | 硬件 | 播放 |
| mic | AFE | 读取 | 硬件 | 捕获 |
| cwloop | AFE | 写入 | 环回 | 捕获 |
| crloop | 应用程序 | 读取 | 环回 | 捕获 |

#### 5.1.2.1.1 外部参考信号

在外部参考配置中，AFE禁用`spk`设备，因为它不负责播放音频，并且`prloop`未打开。相反，它会打开另一个设备，该设备必须具有与扬声器播放的音频相同的数据。捕获流程保持不变。

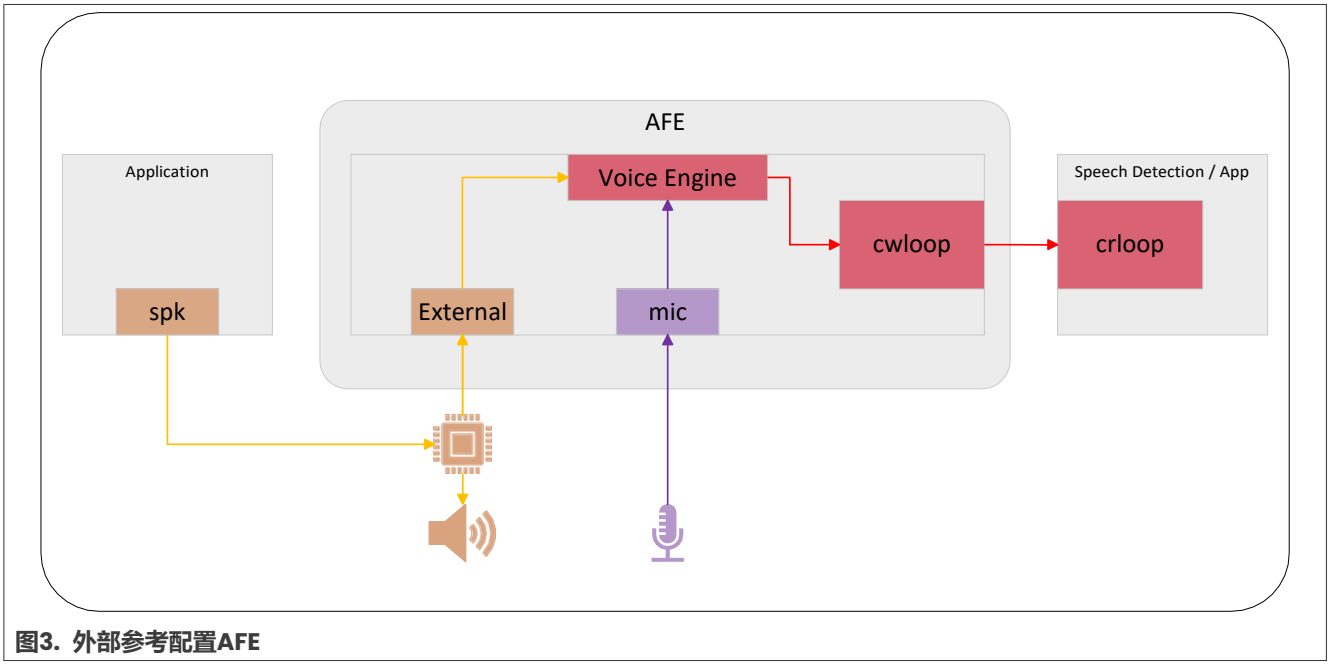


图3. 外部参考配置AFE

### 5.1.3 VoiceSeeker

VoiceSeeker是恩智浦的语音处理库，用于清理传入数据。该解决方案的全名为VoiceSeeker Light (VSL)。VoiceSeeker可以执行以下操作：

- 检测说话声音来自哪个麦克风，并聚焦该麦克风（波束成形）。
- 去除环境背景噪音，提供清洁的语音版本（AEC）。
- 适应任何麦克风阵列，无论其物理位置在哪里。
- 修正声学延迟（见第8.1节）。

VoiceSeeker是一个使用AFE API的库示例，它具有所有方法，并作为共享对象构建，允许AFE加载它（如有必要）。AFE可以使用`afe libvoiceseekerlight`命令加载VSL。当AFE加载VoiceSeeker库时，它会根据库的要求设置ALSA设备。VoiceSeeker的默认要求如下：

- 四个麦克风输入通道。
- 两个扬声器输出通道。
- 所有设备的采样率为16000kHz。
- 所有设备的格式为`S32_LE`。

当所有设备都打开并配置好后，AFE初始化相应的ALSA设备并开始流式传输。AFE控制这些流，当有足够的数据需要处理时，会调用VSL进程。在VSL进程中，延迟在处理数据之前被修正，音频进行了清洁处理。当VSL完成数据处理后，它将数据写入输出缓冲区，AFE负责将数据写入环回接口，其他应用程序在此监听。

这是对VSL在AFE中所做工作的简要说明。然而，进程中有一些重要的部分需要您注意并进行配置。从一开始，在VSL初始化时，会发生一些事情。首先是一些变量的初始化，这取决于要处理的通道数量。如果想使用比默认值更多的通道（四个麦克风和两个参考信号），则必须更改`heap_size`和`scratch_size`。另一个取决于参考通道数量的变量是`sizeBufDelay`，它也取决于系统的声学延迟。如果有更多的参考信号，或者麦克风离扬声器太远，请更改此变量。

继续初始化，可以使用`vsl_config`变量来配置VSL，该变量是一个包含表2中列出的元素的结构：

**表2. vsl_config变量的元素**

| 元素 | 说明 |
|---|---|
| num_mics | 麦克风数量 |
| num_spks | 扬声器（参考信号）数量 |
| framesize_out | 输出缓冲区大小 |
| buffer_length_sec | 预期唤醒词的时间 |
| aec_filter_length_ms | 窗口大小 |
| create_aec | 启用AEC |
| create_doa | 启用DOA（到达方向） |
| mic_xyz_mm | 麦克风之间的距离 |
| device_id | 此处列出的i.MX 8M或i.MX 9的平台ID |

前两个变量不言自明，它们只包含参考信号和麦克风的数量。第三个有点复杂。VoiceSeeker每次迭代处理32帧的块，当缓冲区长度达到请求的样本数时，它返回一个指向过滤缓冲区的指针。这个样本数使用`framesize_out`文件设置，默认为200个样本。由于VoiceSpot的限制，这是默认值和建议值。第四个元素与唤醒词大小有关。例如，如果您使用一个长单词作为唤醒词（如“Hey NXP”），请将此值增加到三秒。如果唤醒词较短（如“Alexa”），则此元素的值可以设置为1.5秒（这是默认值）。

`aec_filter_length_ms`表示窗口大小。值越大，每次迭代所需的时间就越长，且效果不佳。将其保持在150ms到300ms之间是一个不错的选择。

下一个变量启用AEC算法。免费版本不允许启用AEC。要获取完整版本，请通过Voice@nxp.com联系我们的团队。本文涵盖了免费和非免费版本的库。您还可以启用到达方向（DOA），但默认情况下，预安装版本上禁用此功能，本文也未涵盖此功能。

mic_xyz_mm是一个矩阵，其大小为通道数乘以三个维度（宽、深、高）。原点和麦克风之间的距离以毫米为单位测量。原点可以在任何地方，不一定在其中一个麦克风中。

VSL需要知道它在哪个平台上运行，并根据运行的平台进行一些优化，这是最后一个变量的用途。

在构造函数中，最后一步是创建并初始化用于修正声学延迟的机制。该机制在使用前需要校准，以实现可靠的性能。校准方法见第8.1节。

#### 5.1.3.1 Config.ini

VoiceSeeker读取正确配置的文件。此文件名为Config.ini，位于unit_tests/nxp-afe/文件夹中。在初始化过程中，VoiceSeeker和VoiceSpot读取并解析该文件以进行自我配置。

表3列出了文件的参数及其作用。

**表3. 文件参数**

| 字段 | 类型 | 说明 | 选项 |
|---|---|---|---|
| WWDectionDisable | Int | 禁用VoiceSpot、或VIT的唤醒词检测功能 | 0或1 |
| WakeWordEngine | String | 用于唤醒词的引擎 | VoiceSpot或VIT |
| DebugEnable | Int | 允许将缓冲区转储到文件中。库校准需要该字段。 | 0或1 |
| RefSignalDelay | Int | 样本中现有声学延迟的值。 | [0，sizeBuffDelay - 1] |
| mic0 | Float | 与第一个麦克风上的原点之间的距离（毫秒）。 | Float |
| mic1 | Float | 与第二个麦克风上的原点之间的距离（毫秒）。 | Float |
| mic2 | Float | 与第三个麦克风上的原点之间的距离（毫秒）。 | Float |
| mic3 | Float | 测量与第四个麦克风上的原点之间的距离（毫秒）。 | Float |
| VoiceSpotModel | File | 要在VoiceSpot中使用的模型文件。这是VoiceSpot的字段。VoiceSpot在Config.ini文件所在的路径中查找该文件。 | String |
| VoiceSpotParams | File | 模型参数文件。这是VoiceSpot的字段。VoiceSpot在Config.ini文件所在的路径中查找该文件。 | String |
| VITLanguage | String | VIT模型的语言 | 请访问VIT，了解可用的语言信息。 |

VoiceSeeker使用此文件进行自我配置。如果该文件不存在，VoiceSeeker将恢复默认值。默认值与该文件中列出的值相同。如果配置文件中不存在字段，VoiceSeeker将使用默认字段进行设置，因此请确保字段无拼写错误。

### 5.1.4 VoiceSpot

与VoiceSeeker一样，VoiceSpot是一个库，为了更方便地使用语音引擎，封装在（名为`SignalProcessor_VoiceSpot`）的类中。VoiceSpot是一个独立的程序，可以与VoiceSeeker进行通讯，以检索检测到语音信号的时间和地点的反馈。这就是为什么VoiceSeeker和VoiceSpot能够搭配工作的原因。VoiceSeeker在有大量可用数据时发出通知，VoiceSpot告知VoiceSeeker传入数据中是否存在语音特征，这样VoiceSeeker就可以专注于该通道，并在运行时重新调整其过滤器。

和VoiceSeeker一样，VoiceSpot也是一个对象。必须有一个主进程来控制数据的工作流程并创建库的实例。这个程序称为`voice_ui_app`，过去名为`voicespot`。该程序负责打开捕获ALSA设备。当数据准备就绪时，它会调用VoiceSpot或VIT进行语音解码。它通知（如果如此配置）第三个应用程序，以便可以根据刚刚到达的命令或唤醒词启动操作。此类应用可以是语音助手（或相似的应用）。

`Voice_ui_app`与VIT配合使用，用于唤醒词检测和命令检测，或仅用于命令检测。在第一种情况下，VoiceSpot被完全禁用，VIT在检测到唤醒词或命令时会通知VoiceSeeker。启用VoiceSpot后，`voice_ui_app`会调用VoiceSpot来搜索唤醒词。如果VoiceSpot能够检测到唤醒词，则会将缓冲区传递给VIT来处理传入的语音命令。要为唤醒词引擎设置VIT，请编辑配置文件。将`WakeWordEngine`变量设置为VIT而非VoiceSpot。

在运行`voice_ui_app`时，通过传递“`-notify`”标志来启用通知机制。如果标志被传递，两个引擎都会通知第三个应用程序。这意味着，当缓冲区中有唤醒词时，VoiceSpot会发出通知，VIT会通知检测到的语音命令的ID。当VoiceSpot被禁用时，VIT仍然会在检测到唤醒词时发出通知。

#### 5.1.4.1 模型

VoiceSpot使用一个机器学习（ML）模型来检测唤醒词，该模型经过最佳质量和环境的训练，即使在嘈杂的环境中也能达到很高的精度。因此，如果想把VoiceSpot与其他模型一起使用，请通过Voice@nxp.com联系我们，以更好地了解为自定义唤醒词训练模型所需的成本和时间。

在获得了模型及模型参数后，请确保这些文件位于`/unit_tests/nxp-afe`文件夹中，以便VoiceSpot能够找到它们。

## 6 安装指南

本节介绍了安装过程。

### 6.1 硬件检查清单

本节介绍硬件检查清单。

#### 6.1.1 对于i.MX 8M Plus

VoiceSeeker和VoiceSpot目前由恩智浦i.MX ArmV8-A处理器支持，如i.MX 8M Plus和i.MX 8M MINI，并支持ArmV8.2架构，如i.MX93。i.MX 8M Plus EVK套件（以及其他套件）包含以下用于评估VoiceSeeker和VoiceSpot的项目：

**表4. 评估VoiceSeeker和VoiceSpot的项目**

| 数量 | 名称 | 说明 |
|---|---|---|
| 1 | i.MX 8M Plus EVK板 | 恩智浦i.MX 8M Plus评估板。 |
| 1 | 电源 | USB-Type C 45-W供电电源，支持5 V/3 A、9 V/3 A、15 V/3 A、20 V/2.25 A。 |
| 1 | Type-C公头对type-A公头数据线 | 已组装好，符合USB 3.0规范。 |
| 1 | Type-A公头对micro-B公头数据线 | 已组装好，符合USB 2.0，用于UART调试。 |

还需要其它硬件：

**表5. 其他硬件**

| 数量 | 名称 | 说明 |
|---|---|---|
| 1 | 8MIC-RPI-MX8 | 恩智浦8麦克风板。可在https://www.nxp.com.cn/part/8MIC-RPI-MX8#/获得。 |
| 1 | 扬声器 | 一个或两个兼容3.5-mm音频插孔连接的扬声器。 |
| 1 | 3.5-mm数据线 | 连接EVK和扬声器的3.5-mm音频连接线。 |

## 6.2 设置视图

本节介绍了测试VoiceSeeker和VoiceSpot库的常见设置。每种设置可能有所不同，但连接必须相同。

图4. 常见设置

1. 对于8MIC-RPI-IMX8电路板：
   a. 将电路板安装到EVK的EXP_CN上。确保其引脚编号与EVK上的编号相匹配。
   b. 通过下拉电路板的静音开关来取消电路板的静音。如果未设置，当电路板上电时，则开关会显示静音状态。
   c. 上拉MIC_SEL，使ON位置处于活动状态。
2. 对于8M Plus：
   a. 将音频插孔数据线连接到EVK和扬声器。
   b. 连接电源。
   c. 将调试数据线连接到PC。

## 6.3 软件安装

本节介绍了VoiceSeeker和VoiceSpot所需的所有软件、程序和文件。

### 6.3.1 软件检查清单

推荐的BSP版本为MM_04.09.00_2405_lf6.6.23或更高版本。最新的镜像可以从https://www.nxp.com.cn/design/design-center/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX下载。

当镜像完全下载后，可以按照《i.MX Linux用户指南》（文档IMXLUG）中的步骤进行安装。

镜像应包含表6中列出的文件和驱动程序。

**表6. 文件和驱动程序**

| 文件 | 位置 | 说明 |
|---|---|---|
| libvoiceseekerlight.so | /usr/lib/nxp-afe/ | VoiceSeeker包装器。运行时由AFE加载。 |
| afe | /unit_tests/nxp-afe/ | AFE二进制文件。 |
| voice_ui_app | /unit_tests/nxp-afe/ | 主应用程序。 |
| asound.conf_imx8mp | /unit_tests/nxp-afe/ | ALSA配置文件。 |
| Confing.ini | /unit_tests/nxp-afe/ | VoiceSeeker和VoiceSpot的配置文件。 |
| HeyNXP_1_params.bin | /unit_tests/nxp-afe/ | 恩智浦唤醒词参数。 |
| HeyNXP_en-US_1.bin | /unit_tests/nxp-afe/ | 恩智浦唤醒词模型。 |
| snd-aloop | /lib/modules/<bs_version>/kernel/sound/drivers | 用于在ALSA内创建环回设备的驱动程序。 |

# 7 软件设置

本节介绍了软件设置。

## 7.1 为8MIC-RPI-MX8电路板选择设备树

将Linux BSP烧写到EVK的目标存储介质后，启动电路板并在U-Boot终端停止。在启动过程中，在U-Boot提示后按任意键即可完成此操作。当U-Boot控制台准备就绪时，输入以下命令：

```
u-boot=> editenv fdtfile
edit: imx8mp-evk-<revision>-8mic-revE.dtb
u-boot=> saveenv
u-boot=> boot
```

不同版本的特定设备树如表7所示：

**表7. 设备树**

| 设备树 | 支持的i.MX 8MP EVK版本 |
|---|---|
| Imx8mp-evk-revb4-8mic-revE.dtb | B3、B4 |
| Imx8mp-evk-revA3-8mic-revE.dtb | A3、B、B1 |
| Imx8mp-evk.dtb | A2或更早版本 |

### 7.2 安装缺失的驱动程序

默认情况下，镜像不会加载允许ALSA执行环回流的驱动程序。因此，需要使用以下命令手动安装：

```
$ modprobe snd-aloop
```

### 7.3 编辑VoiceSeeker的ALSA asound配置

以下命令会覆盖ALSA虚拟设备的配置，以便与VoiceSeeker正常工作，同时保存原始配置的备份。

```
$ cd /unit_tests/nxp-afe/
$ mv /etc/asound.conf{,.back}
$ cp asound.conf_imx8mp /etc/asound.conf
```

如果EVK版本为B3或B4，请复制`asound.conf_imx8mp_revb4`文件。

## 8 音频实验室指南

本节介绍了如何构建库，以及测试库的方法。

### 8.1 测量延迟

如“介绍”部分所述，AFE通过从捕获信号（播放信号和语音信号的混合）中去除播放信号来执行AEC（通过加载VoiceSeeker库）。但是，由于声音从扬声器传到麦克风并被录制下来需要时间，捕获流中混合了语音特征的播放信号相对于原始播放信号有一些延迟，因此精确测量这个延迟对于实现良好的性能非常重要。

在测量延迟之前，需要注意以下几点：

1. 只要设置不变，这是一次性校准。如果麦克风和扬声器之间的距离发生变化，则需要重新校准。
2. 每次测量的延迟可能会有所不同，但这些测量之间的差异通常在一到两个样本的范围内。
3. 还有其他方法可以测量延迟。例如，一种方法是使用L/R（左和右）扬声器和麦克风通道之间的交叉关联。可以使用MATLAB、Python或任何其他具有该功能的库的语言。
4. 本文采用了一种更直观的方法，使用音频调谐应用程序，该应用程序统计分析的音频流中两点之间的样本数。

要测量延迟，需要使用AFE进行样本录音。在样本录音过程中，AFE生成三个音频文件：播放 + 捕获流、仅播放流和仅捕获流（AEC的结果，不需要用于校准）。这些文件有助于测量延迟。

在按照`TODO.md`文件的建议运行程序之前，必须更改VoiceSeeker的配置。需要启用调试功能，并将延迟属性设置为0。这可以通过更改`Config.ini`文件来完成，如下所示：

```
$ vi Config.ini
```

设置DebugEnable和RefSignalDelay属性的值。校准完成后可以关闭DebugEnable变量，但为了本文的目的，保持开启状态。

```
DebugEnable = 1
RefSignalDelay = 0
```

保存文件并退出Vi。

使用VoiceSeeker库启动voice_ui_app和AFE，如TODO.md文件所述。确保AFE和voice_ui_app都在后台运行。如果其中一个未运行，可能会导致意外行为。

```
$ ./voice_ui_app &
```

要验证voice_ui_app是否正确初始化，请查看日志。如果voice_ui_app打印了可用的VIT命令，则表示voice_ui_app正确初始化VoiceSpot和VIT。

```
$ ./afe libvoiceseekerlight &
```

要生成已知信号，使用GST。使用gst-launch-1.0命令创建一个最小流程。该命令会生成一个周期性滴答，这使得测量延迟变得更加容易。播放音频几次滴答。

```
$ gst-launch-1.0 audiotestsrc wave=8 ! alsasink
```

扬声器中可以听到滴答声。

有关任何GST插件的更多信息，请运行gst-inspect-1.0命令。

```
$ gst-inspect-1.0 audiotestsrc #As an example
```

播放音频几次滴答后，按Ctrl + C停止流程，并终止AFE和voice_ui_app。

```
$ pkill afe
$ pkill voice_ui_app
```

上述三个音频文件应生成并位于/tmp/目录中：

```
$ ls -lh /tmp/*.wav
-rw-r--r-- 1 root root 8.4M Mar 5 07:31 mic_in_delay_S0_E1.wav
-rw-r--r-- 1 root root 2.1M Mar 5 07:31 mic_out_S0_E1.wav
-rw-r--r-- 1 root root 4.2M Mar 5 07:31 ref_in_delay_S0_E1.wav
```

- mic_in_delay_S0_E1.wav文件包含从扬声器播放并在8MIC电路板录制的混合信号（播放和语音）（延迟信号）。
- mic_out_S0_E1.wav文件包含从AEC算法获得的清洁语音（在本示例中，理想情况下为空，因为未录制语音）。
- ref_in_delay_S0_E1.wav文件包含未录制的原始播放流（滴答声）（无延迟信号）。

其中：

- _S0_ (Start minute)：开始录制的分钟数。
- _E1_ (End minute)：停止录制的分钟数。

默认情况下，库会记录一对分钟（0-1、2-3、4-5等）。要更改默认行为，请更改MINUTE_INTERVAL_WAV_FILE配置宏（位于<root_dir>/voiceseeker/src/SignalProcessor_VoiceSeekerLight.h）。例如，要在独立文件中记录所有分钟，将宏设置为 1。

只需要mic_in_delay_S0_E1.wav和ref_in_delay_S0_E1.wav文件便可测量延迟。将这两个文件复制到主机（使用scp或其他方法），用Audacity打开其中一个文件，并将另一个波形文件拖放到同一个Audacity窗口中。

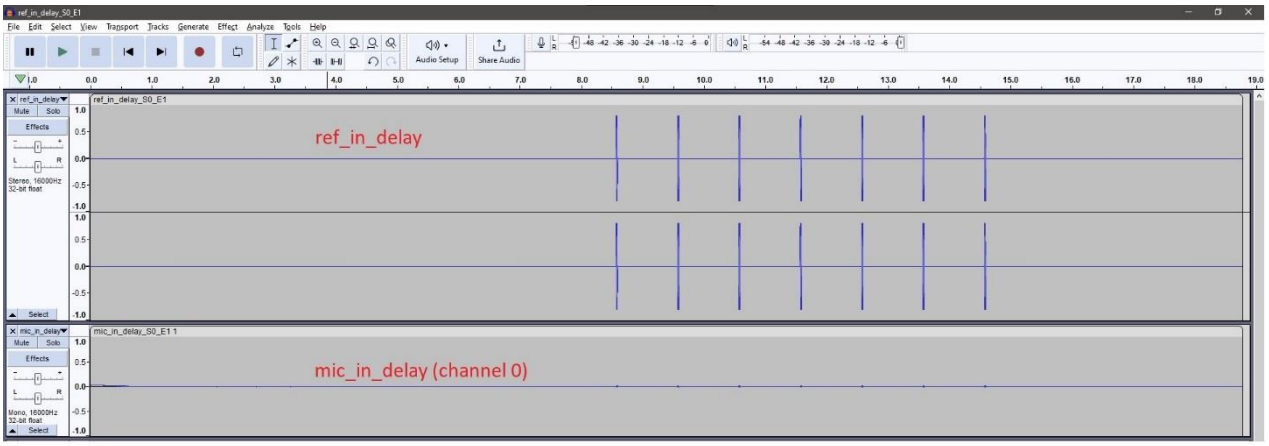


**图5. Audacity窗口**

为了更好地查看数据，请执行以下步骤，以增加麦克风输入的增益：

1. 选择四个麦克风通道。
2. 转到“Effect -> Volume and Compression -> Amplify”（“效果->体积和压缩->放大”）。
3. 出现一个弹出窗口。勾选“允许剪切”框并设置放大倍数，以便清楚地看到滴答波。

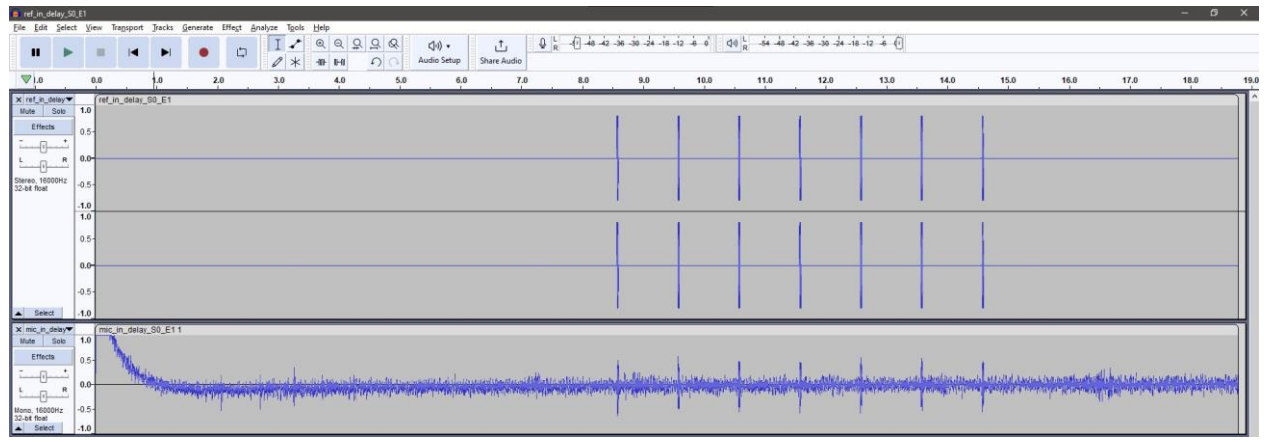

**图6. 滴答波**

为了获得良好的精度，请放大参考信号中第一个滴答的开始部分，并选择参考点与其在麦克风通道中的对应位置之间的所有样本。

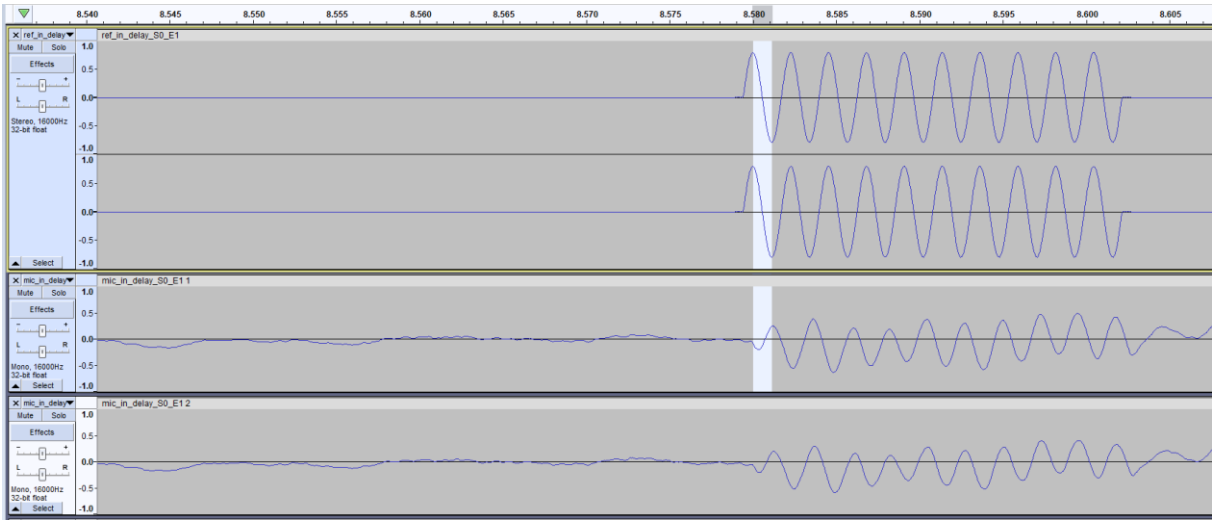

图7. 麦克风通道

延迟可以用时间或样本为单位来测量。VoiceSeeker 使用样本单位的延迟。音频编辑器通常允许以样本单位进行时间帧测量。在Audacity中，转到左下角并将选项栏更改为“开始和长度选择”。然后您将看到信号延迟了多少样本。

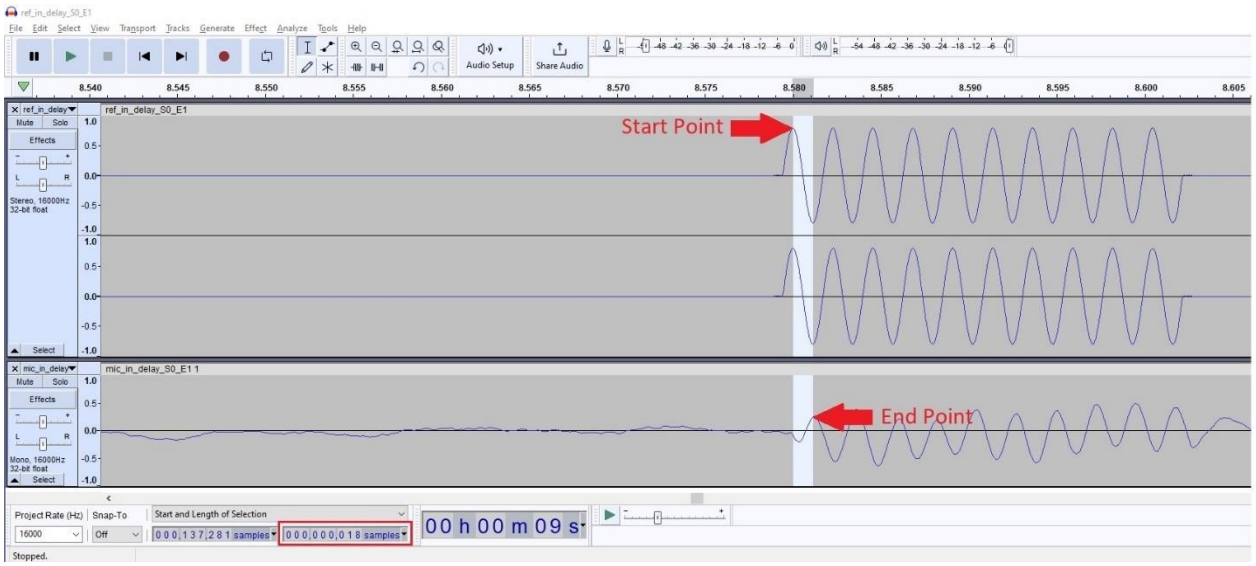


图8. 信号延迟

使用这个新值设置`RefSignalDelay`。

## 8.2 波束成形

本节介绍了VoiceSeeker与VoiceSpot结合使用时的波束成形功能的性能。

波束成形是指在检测到某些属性时突出信号的一部分的能力。VoiceSeeker在检测到唤醒词时使用波束成形功能。

通过播放纯音，触发唤醒词，并从VIT发出语音命令，可以清晰地查看此功能。

请确保先删除临时文件，以使用最新的WAV文件。

```
$ rm -v /tmp/*.wav
```

使用voice_ui_app启动VoiceSpot和VIT，并使用AFE来运行VoiceSeeker。

```
$ ./voice_ui_app &
$ ./afe libvoiceseekerlight &
```

使用GST播放纯音。

```
$ gst-launch-1.0 audiotestsrc wave=0 ! alsasink
```

说出唤醒词和语音命令，例如“Hey NXP! Mute” （“嘿，恩智浦！静音”）。等待几秒钟，然后重复。唤醒词和语音命令可以更改。

按“Ctrl + C”停止GST并停止进程，与之前一样。

```
$ pkill voice_ui_app
$ pkill afe
```

这次需要的波形文件为mic_in_delay_S0_E1.wav和mic_out_S0_E1.wav。在第一个文件中，语音听起来像是在背景中，而在第二个文件中，语音听起来像是移到了前面。

将这些文件复制到主机，打开mic_out_S0_E1.wav文件，然后拖放另一个文件。

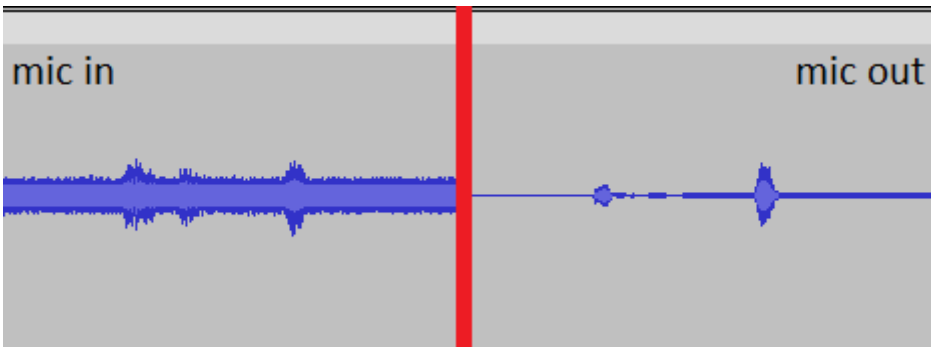


**图9. mic_in_delay_S0_E1.wav和mic_out_S0_E1.wav**

选择所有通道并对其应用一些增益。在播放音频之前，可以看到波束成形如何作用于麦克风输出信号，在检测到语音时将其隔离。

另一种比较音频的方法是，按每个通道左侧的“solo” （“独奏”）按钮，在独奏模式下播放每个音频。从麦克风播放任何通道时，语音听起来像是在背景中。确实如此，因为扬声器离麦克风更近。然而，当播放麦克风输出数据时，语音似乎更接近麦克风而不是扬声器。

## 8.3 外部参考信号

本节介绍了如何使用当前设置测试外部信号功能。

其目的是创建一个虚拟设备，该设备负责将信号写入扬声器并向AFE提供参考信号，如图10所示：

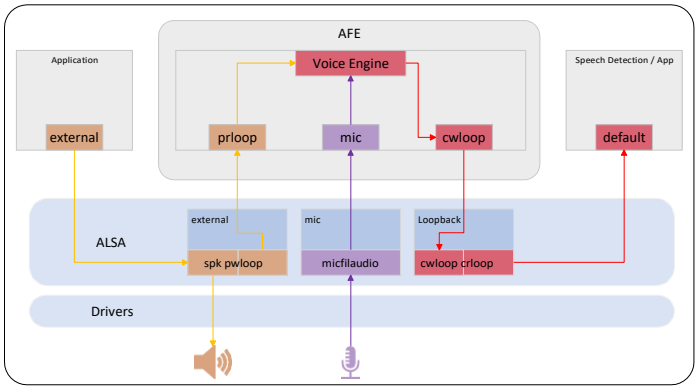


**图10. 虚拟设备**

### 8.3.1 asound.conf

要测试此功能，请将以下设备添加到/etc/asound.conf文件中。其目的是在ALSA级别复制播放流，ALSA负责写入扬声器并向AFE提供流数据。该应用程序必须写入默认设备之外的其它设备，但如果外部设备在默认流程中替换了plug:mix设备，则对应用程序来说是透明的。

使用选择的编辑器打开asound.conf文件，并在该文件中添加以下行：

```
pcm.external {
        type plug
        slave.pcm "stereo2quad"
}

pcm.stereo2quad {
        type route
        slave.pcm "loopNspk"
        ttable.0.0 1
        ttable.1.1 1
        ttable.0.2 1
        ttable.1.3 1
}

pcm.loopNspk {
        type multi
        slaves.a.pcm "spk"
        slaves.a.channels 2
        slaves.b.pcm "pwloop"
        slaves.b.channels 2
        bindings.0 { slave a; channel 0; }
        bindings.1 { slave a; channel 1; }
        bindings.2 { slave b; channel 0; }
        bindings.3 { slave b; channel 1; }
}
```

保存文件并退出。

上面的代码创建了三个设备。第一个设备（external）是链中的第一个元素，应用程序必须与之通讯。它负责将格式/速率转换为支持的输出。第二个设备（stereo2quad）复制通道。第三个设备（loopNspk）是主要部分。该设备负责写入两个不同的声卡。一个用于扬声器，另一个用于打开写入AFE输入设备的设备。

使用这些设备和使用默认设备的区别在于，ALSA管理播放线程。当运行以下命令时，它会导致扬声器在AFE执行之前就开始播放。

```
$ gst-launch-1.0 audiotestsrc wave=0 ! alsasink device=external
```

### 8.3.2 运行流程

本节介绍如何运行流程。

#### 8.3.2.1 voice_UI

`voice_ui_app`的运行方式与之前相同，不需要任何更改。

```
$ ./voice_ui_app &
```

#### 8.3.2.2 AFE

在外部模式下运行AFE非常简单，如前所述。在此配置中，最重要的是知道应该打开哪个设备，AFE会处理其余事项。在本示例中，`prloop`是必须打开才能获取参考信号的默认设备。然而，通过与以前相同的方式运行该程序可能会产生错误，因为扬声器设备已被另一个应用程序占用。AFE必须禁用此设备，以避免运行时出现任何错误。所有这些都通过以下命令完成：

```
$ ./afe libvoiceseekerlight prloop &
```

当AFE已经打开了包含参考数据的设备时，执行播放应用程序：

```
$ gst-launch-1.0 audiotestsrc wave=0 ! alsasink device=external
```

几秒钟后停止该应用程序。将`ref_in_delay_S0_E1.wav`文件复制到主机，并使用Audacity打开它以验证文件不为空。此外，执行`hexdump`就足够了。

## 8.4 启用AEC

本节介绍了如何在启用AEC的情况下构建VoiceSeeker。构建VoiceSpot而不超时需要相似的过程。

当VoiceSeeker AEC配置正确且系统中没有振动或AFE未检测到信号中有任何额外的后处理时，VoiceSeeker AEC会提供25dB到30dB的参考信号衰减。

### 8.4.1 获取库

要启用声学回声消除（AEC），请通过voice@nxp.com联系恩智浦语音团队以升级VoiceSeeker库。

语音团队会为所请求的平台提供一个文件夹，类似于imx-voiceui存储库中的`voiceseeker/platforms`文件夹中的文件夹。获取了文件夹后，将其安装到源代码中并重新编译二进制文件。

### 8.4.2 安装库

从GitHub克隆存储库：

```
$ git clone https://github.com/nxp-imx/imx-voiceui.git
```

转到最新的分支：

```
$ cd imx-voiceui
$ git switch MM_04.09.00_2405_L6.6.y
```

用包含VoiceSeeker完整版本的新文件夹替换`voiceseeker/platforms/iMX8M_CortexA53`文件夹：

```
$ cp -r ../iMX8M_CortexA53/* ./voiceseeker/platforms/iMX8M_CortexA53
```

### 8.4.3 编译库

readme.md文件提供了在启用AEC的情况下编译库的说明。然而，从终端设置AEC变量更容易，并且在构建镜像时需要按的键更少。

```
$ export AEC=1
$ make
```

完成上一步后，应在编译开始前打印以下消息：

```
Building with AEC
```

构建库会生成`release`文件夹，该文件夹中包含`voice_ui_app`二进制文件和`libvoiceseekerlight.so.2.0`，以及其他不需要部署到电路板上的文件，因为这些文件未更改。

### 8.4.4 电路板部署

在本示例中，唯一真正需要的二进制文件是VoiceSeeker，但最好覆盖这两个二进制文件。

```
$ scp release/libvoiceseekerlight.so.2.0 root@<BOARD_IP>:/usr/lib/nxp-afe/
$ scp release/voice_ui_app root@<BOARD_IP>:/unit_tests/nxp-afe/
```

在将二进制文件安装到目标电路板上后，运行这些文件以验证其是否正常工作。

当VoiceSeeker初始化时，它会转储一个配置状态，告知它是如何配置的。在该部分中，有一个名为`create_aec`的属性，当AEC启用时，该属性的值为“1”，未启用时，值为“0”。

## 8.5 低功耗语音演示

由于这个主题太复杂，需要一个专门的文档来涵盖所有内容。有关如何启用该演示，请参阅《低功耗语音UI演示》（文档AN13957）。

# 9 附录

## 9.1 使用恩智浦SWPDM和AFE

本节介绍了如何将恩智浦SWPDM ALSA插件集成到语音流程中。

### 9.1.1 恩智浦SWPDM

恩智浦SWPDM是面向i.MX 8MM和i.MX 8MN处理器的解决方案，这些处理器不具备语音处理任务所需的32位宽分辨率。通过将此ALSA插件添加到流程中，它提高了唤醒词和语音命令的准确性，而不会显著增加CPU负载。

SWPDM插件使用CIC滤波算法将来自麦克风的PDM数据转换为任何音频后处理活动所需的PCM格式。

### 9.1.2 将SWPDM添加到VoiceSeeker流程

本节介绍了将SWPDM集成到VoiceSeeker流程所需的步骤。尽管8M Plus不需要该插件，但它与该插件兼容，并且步骤对于所有电路板（8M Plus、8MM或8MN）都是相同的。

#### 9.1.2.1 使用SWPDM

要使用SWPDM插件，请参阅技术论坛帖子，其中介绍了此主题以及将插件与AFE和`voice_ui_app`相集成的步骤。

## 9.2 麦克风排列

本节介绍了如何编辑麦克风的预期几何形状，以匹配8MIC-RPI-MX8电路板上的麦克风几何形状。这种方法也可以用于任何其他麦克风几何形状。

### 9.2.1 麦克风布线

8MiC-RPI-MX8电路板有8个麦克风，编号从0到7，各自的位置如图11所示。


**图11. 麦克风**

还需要麦克风相对位置来增强AEC。这些位置如图12所示。在此示例中，距离是从麦克风中心开始测量的，单位为毫米。

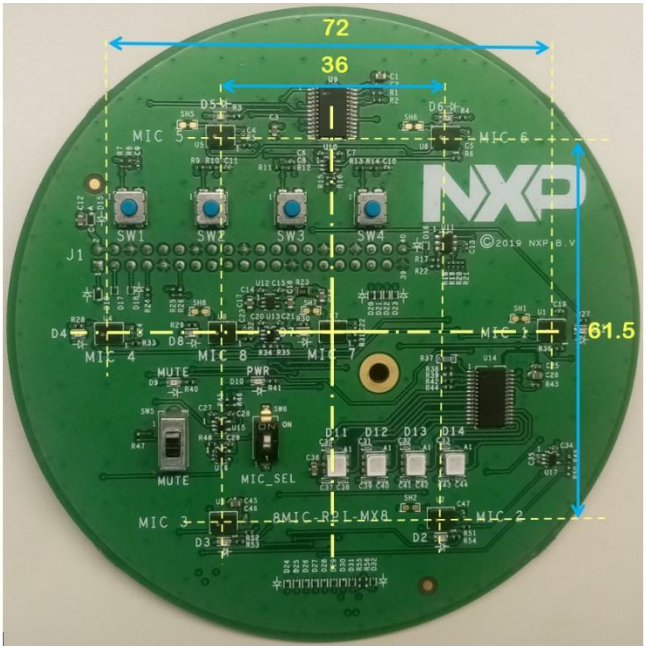


**图12. 麦克风的相对位置**

通过使用`ttable`更改`/etc/asound.conf`文件，可以将每个麦克风连接到录制音频流的任何可用通道：

```
pcm.mic
{ type
route
slave.pcm “hw:micfilauio,0”
slave.channels 8
```

```
ttable.1.1 1
ttable.2.2 1
ttable.3.3 1
ttable.4.4 1
ttable.5.5 1
ttable.6.6 1
ttable.7.7 1 }
```

ttable属性可以将物理麦克风重新连接到任何音频流通道，如下所示：

上述代码片段的图例为ttable.A.B 1，其中：

- A表示音频流中使用的通道索引。
- B表示电路板上的物理麦克风索引。
- 1表示麦克风的数量。

例如，ttable.3.6 1线路可以将来自麦克风6的音频信号连接到通道3。

### 9.2.2 麦克风位置配置

AFE使用config.ini文件设置麦克风的相对坐标。

config.ini文件也必须进行编辑。以下代码片段显示了典型的麦克风坐标设置：

```
mic0 = 0.00, 0.00, 0.00
mic1 = 36.00, 0.00, 0.00
mic2 = -18.00, 30.75, 0.00
mic3= -18.00, -30.75, 0.00
mic4 = 18.00, -30.75, 0.00
mic5 = -36.00, 0.00, 0.00
mic6 = 18.00, 30.75, 0.00
mic7 = -18.00, 0.00, 0.00
```

micx变量中的数字x是指音频流中的虚拟麦克风通道，而不是物理麦克风索引。如果要设置物理麦克风3的坐标，请将其连接到asound.conf文件中一个通道（以通道5为例）：

```
ttable.5.3 1
```

在config.ini文件中将坐标设置为同一通道（通道5）：

```
mic5 = -35.0, 18.0, 0.0
```

坐标的写法如下：

```
micX = x-coordinate, y-coordinate, z-coordinate
```

坐标以毫米为单位测量，并且相对于一个任意点。这个点的位置可以在任意位置，例如在麦克风上、电路板上的某个点，甚至可以在电路板外的某个位置。

为了简化，可以选择一个麦克风作为原点，将其坐标设置为 0, 0, 0，并写下其他麦克风相对于原点的坐标。

### 9.2.3 麦克风排列示例

此示例显示了使用一组自定义麦克风（紫色圈出的那些）的配置。此示例的目的是将麦克风6、5、3和1分别连接到通道0、1、2和3。

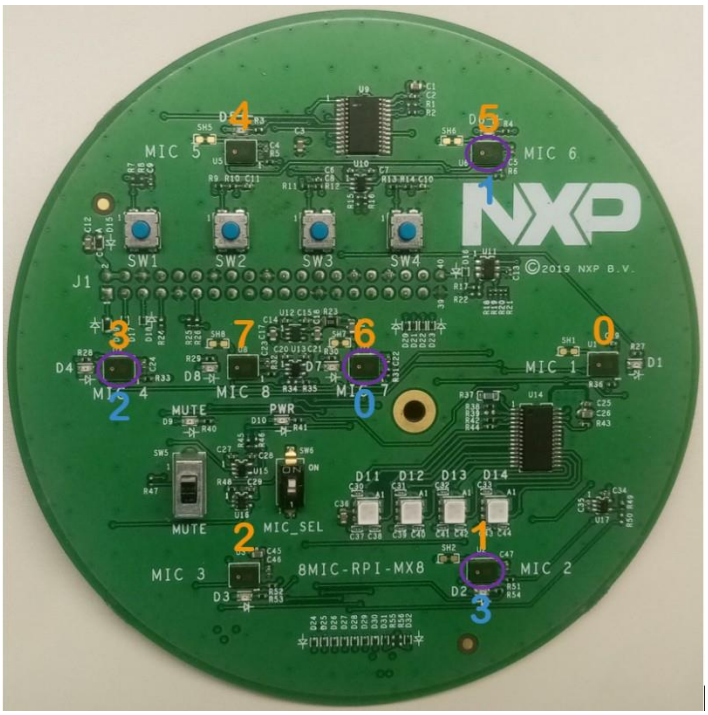


**图13. 麦克风编号以黄色显示，目标通道以蓝色显示**

要将麦克风连接到目标通道，请在`asound.conf`文件中使用以下`ttable`配置：

```
pcm.mic
{ type
route
 slave.pcm "hw:micfilaudio,0"
 slave.channels 8
 ttable.0.6 1
 ttable.1.5 1
 ttable.2.3 1
 ttable.3.1 1
}
```

要设置麦克风对应的坐标，请在`config.ini`文件中使用以下配置：

```
mic0 = 0.0, 0.0, 0.0
mic1 = 18.0, 30.0, 0.0
mic2 = -35.0, 0.0, 0.0
mic3 = 18.0, -30.0, 0.0
```

在本示例中，通道0上的物理麦克风6被选作原点，其他3个通道的位置以该麦克风为原点进行测量。图14显示了四个选定的麦克风及其相对坐标。

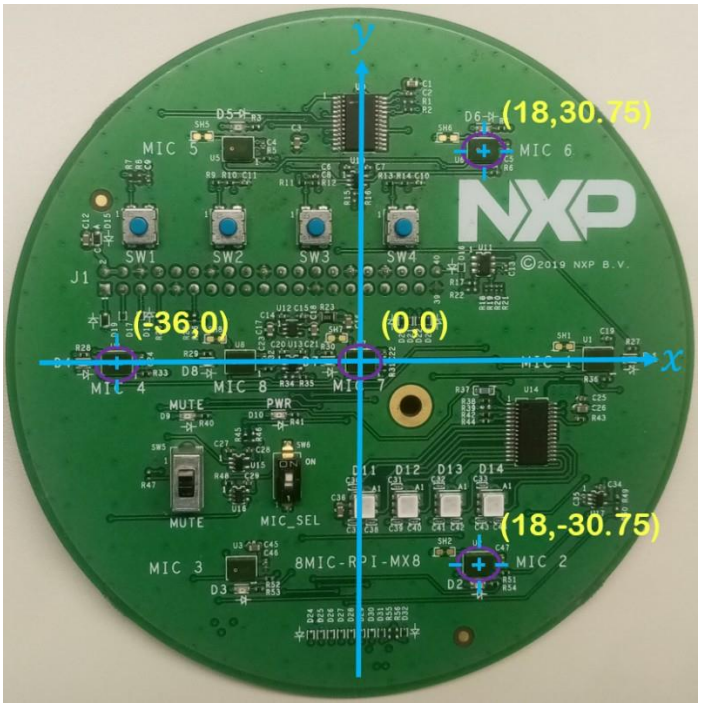


图14. 四个选定的麦克风及其相对坐标

## 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8汇总了本文的修订情况。

表8. 修订历史

| 文档ID | 发布日期 | 说明 |
| --- | --- | --- |
| AN14276 v.1.0 | 2024年6月24日 | 首次公开发布 |

# 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.

**i.MX** — is a trademark of NXP B.V.

**MATLAB** — is a registered trademark of The MathWorks, Inc.

**Microsoft, Azure, and ThreadX** — are trademarks of the Microsoft group of companies.

# 目录