基于 Zephyr 平台的 M55M1 NPU 关键词检测

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	该示例代码在 Zephyr 平台上使用 M55M1 NPU 演示机器学习的关键词检测。
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.003 Zephyr v4.3.0
开发平台	NuMaker-X-M55M1D

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本示例展示了一个关键词检测（KWS）推理应用，它能在录音中识别是否存在指定关键词，运行于新唐（Nuvoton）的 Ethos-U NPU 平台。其源码来源于 Arm 的 “ML Embedded Evaluation Kit/Keyword Spotting”示例，关于关键词检测应用的更多信息可参考该示例。

本示例主要包括：

- 通过数字麦克风（DMIC）录制音频片段。
- 使用 TensorFlow Lite Micro 框架和 Ethos-U NPU 运行关键词检测推理。

  这将涉及三个步骤：

  1. 将录制的音频片段预处理为推理所需的指定输入张量格式。
  2. 运行关键词检测推理。
  3. 对输出张量格式进行后处理，使其便于外部使用。

本示例运行来自 Arm 模型库（Arm Model Zoo）的 DS-CNN Medium INT8 模型。随后使用 Vela 编译器对该模型进行优化，使其可在新唐（Nuvoton）的 Ethos-U NPU 平台上运行并获得加速。

Vela 以 *.tflite* 文件为输入，并生成另一个 *.tflite* 文件作为输出，其中由 Ethos-U 支持的算子替换为 Ethos-U 的自定义算子。理想情况下，整个模型网络将被替换为单个 Ethos-U 自定义算子。

1.1 硬件功能

NuMicro® M55M1 系列是新一代 AI 微控制器，提供高达 110 GOPS 的 AI 计算性能，专为端点 AI 应用（如数据识别和智能音频）而设计。M55M1 是一款 32 位微控制器，基于 220 MHz 的 Arm® Cortex®-M55 内核，并集成 Arm® Ethos™-U55 神经网络处理单元（NPU）作为协处理器，用于加速神经网络运算。除了出色的 AI 性能之外，M55M1 还提供丰富的通信和外设接口，包括用于音频采集的数字麦克风（DMIC）。Ethos-U55 NPU 与 DMIC 的结合，使其非常适合用于展示关键词唤醒（Keyword Spotting）推理应用。

1.2 使用 Vela 优化模型

本节说明如何使用 Vela 将 DS-CNN Medium INT8 模型优化为面向 M55M1 Ethos-U NPU。

- 下载名为 *ds_cnn_m_quantized.tflite* 的 DS-CNN Medium INT8 模型。
- 使用 Vela 编译器优化模型 *ds_cnn_m_quantized.tflite*，用户将获得名为 *ds_cnn_m_quantized_vela.tflite* 的优化模型。

```
$ vela ds_cnn_m_quantized.tflite \
--output-dir . \
--accelerator-config ethos-u55-256 \
--optimise Performance
```

- 为便于区分和清晰命名，将其重命名为 *ds_cnn_m_quantized_ethos-u55-256_opt-speed.tflite*。

```
$ mv ds_cnn_m_quantized_vela.tflite \
ds_cnn_m_quantized_ethos-u55-256_opt-speed.tflite
```

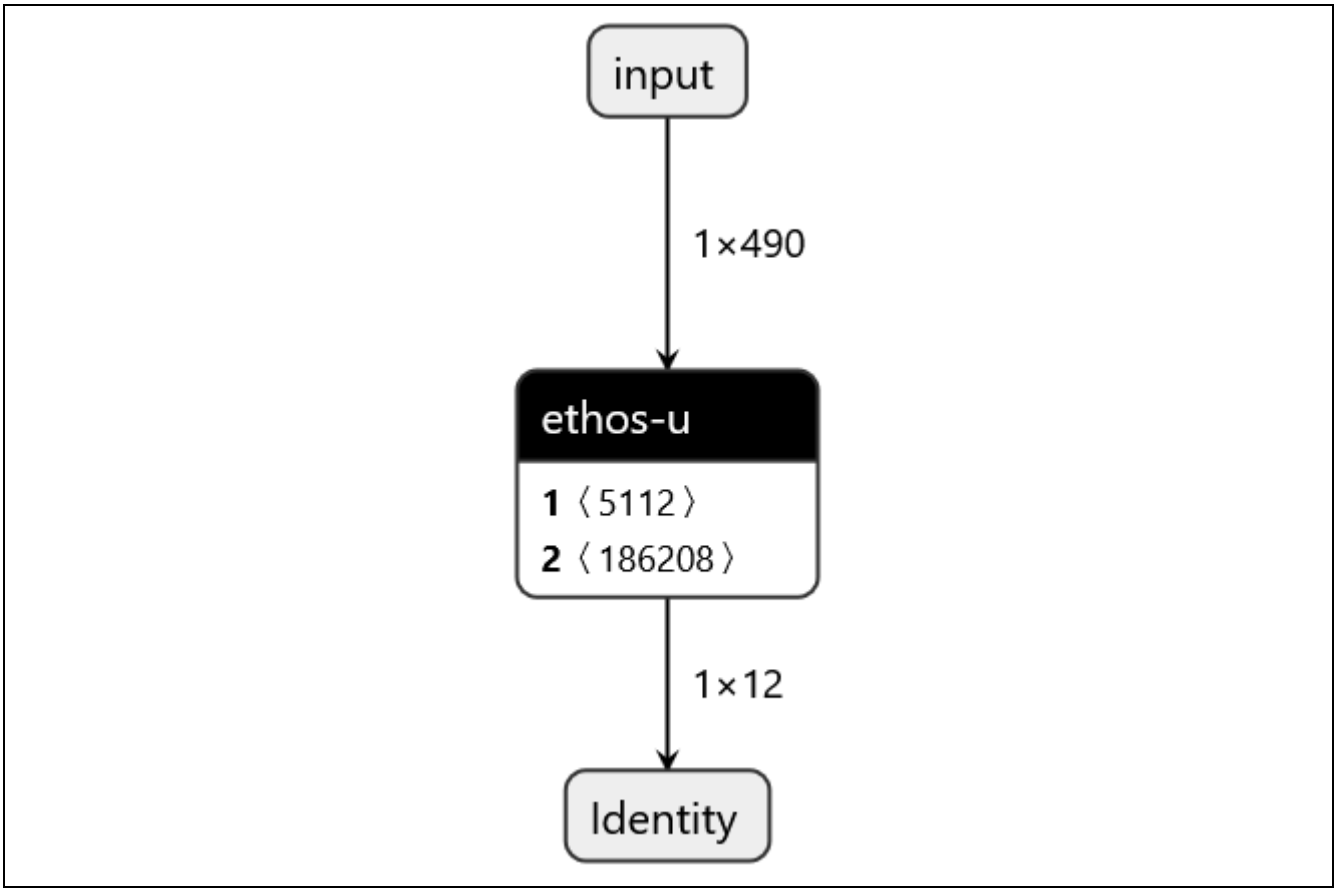

图 1-1 优化模型

1.3 代码、模型与 Ethos-U 之间的绑定

示例代码、Vela 编译后的模型与 Ethos-U NPU 之间存在绑定关系。这意味着，除非用户对它们有充分了解，否则不能随意更改其中任何一个。

由此可见，Vela 编译模型与 Ethos-U NPU 存在匹配约束：M55M1 的 NPU 为 Ethos-U55，且每周期 256 MAC，因此在使用 Vela 时必须将 **--config** 设置为 **ethos-u55-256**，才能与硬件配置一致并确保模型可在目标平台上正确加速运行。

示例代码必须负责推理所需的预处理和后处理。预处理和后处理的逻辑必须遵循模型网络的输入要求以及输出要求。

1.4 演示结果

要演示关键词检测推理，请按照“范例程序执行”一章中的说明进行操作，并将主机终端程序配置为 **115200/8-N-1**（115200 波特率、8 位数据位、无校验、1 个停止位）。执行 **west flash** 之后，可以看到类似如下的消息输出：

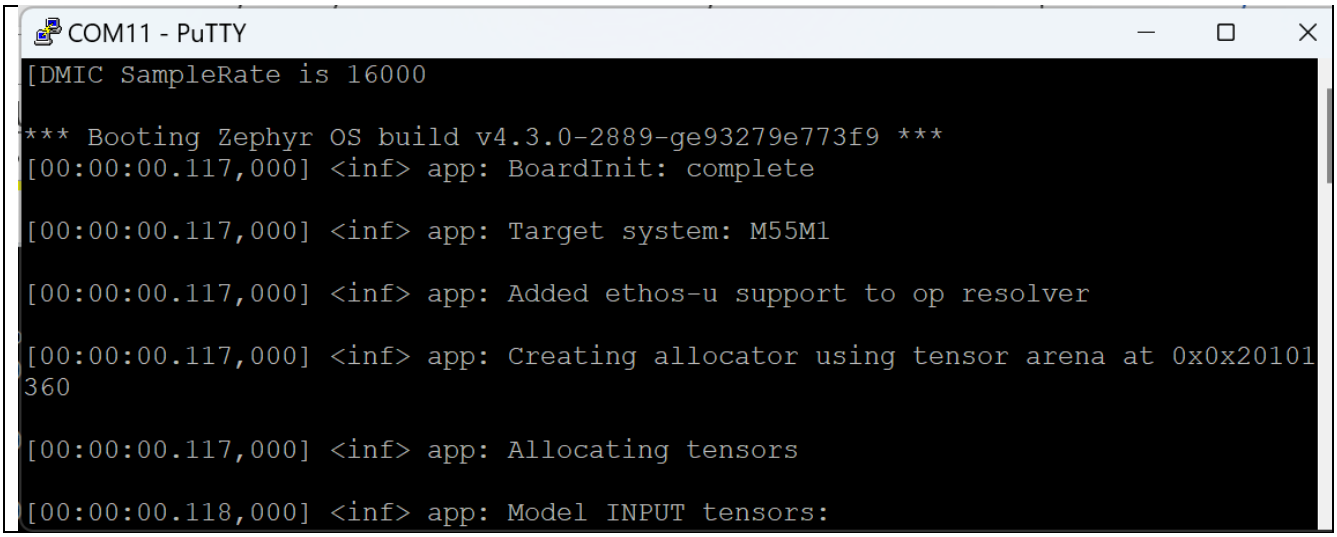

图 1-2 Terminal: Initialize Keyword Spotting Inference

在消息输出的末尾，系统会提示用户输入以继续：

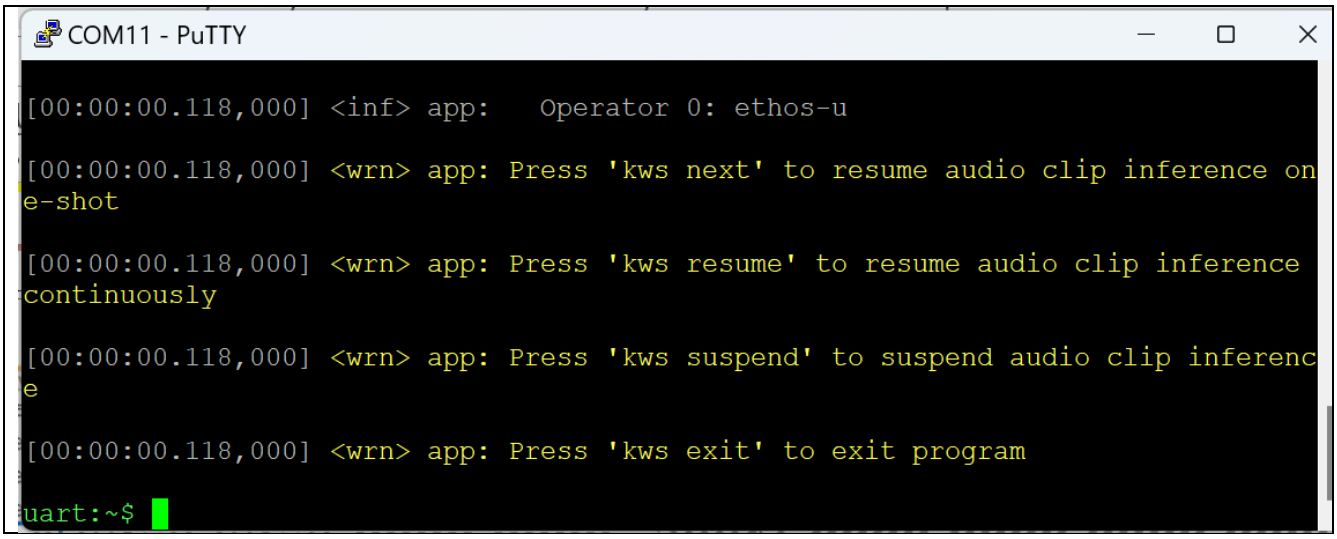

图 1-3 Terminal: Keyword Spotting Shell Commands

输入：**kws resume**

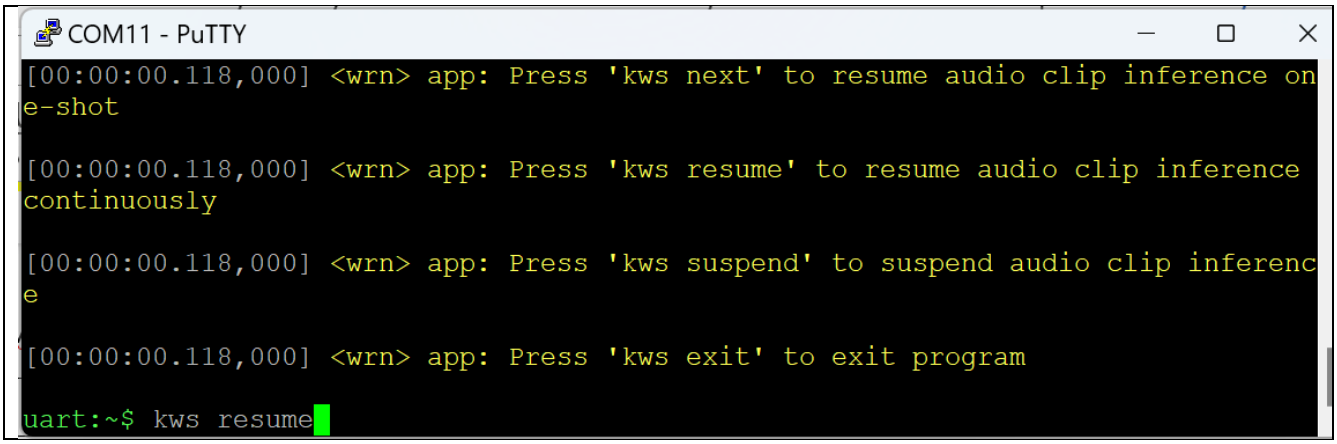

图 1-4 Terminal: Start Keyword Spotting Inference

用户会看到未检测到的消息输出：

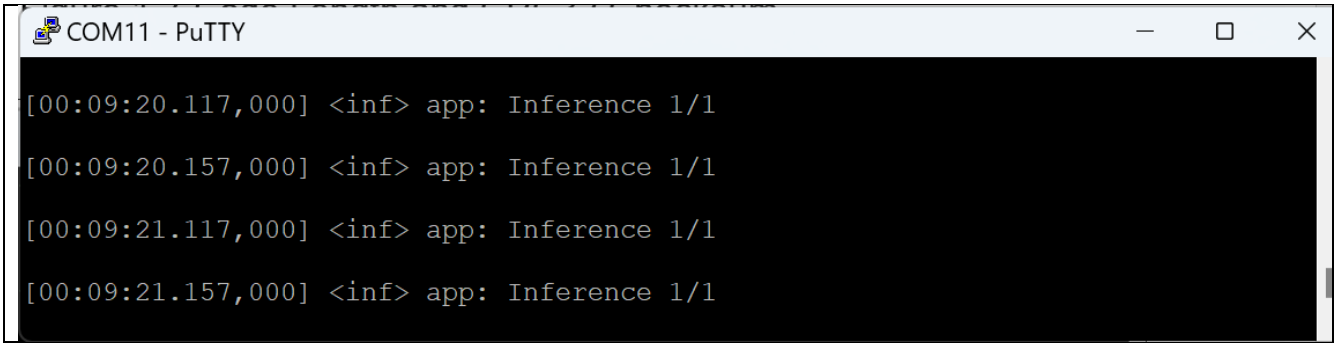

图 1-5 Terminal: Detect No Keyword

靠近目标板，说出已训练的关键词，例如 “Go”，然后用户会看到检测到的消息输出：

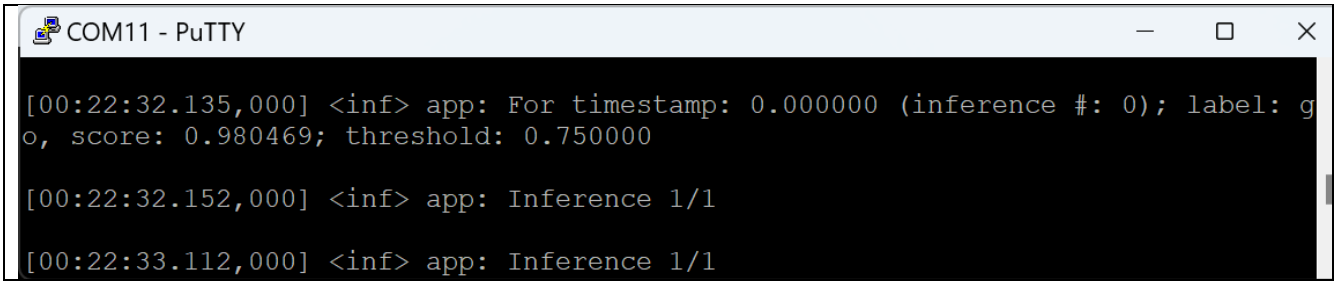

图 1-6 Terminal: Detect Keyword “Go”

该关键词检测模型训练所用的关键词列在文件 *ModelLabels.cpp* 中。

2. 代码介绍

在 Zephyr 上，用户无需显式编写诸如系统时钟和 UART 等常见的初始化代码，因为 Zephyr 的初始化流程会在 **main** 函数之前完成这些任务。本示例中的 Ethos-U 也在该阶段完成初始化。不过，用户仍需手动初始化 Zephyr 未内建支持的其他外设，这些操作在 **BoardInit** 中完成。

```
int main()
{
    /* Board specific initialization */
    BoardInit();
```

在 **BoardInit** 中，为本示例代码初始化 数字麦克风（ ）。

```
    // DMIC_APLL1_FREQ_196608KHZ for DMIC 8000/16000/48000 Hz sample-rate with 64/128/256
down-sample
    CLK_EnableAPLL(CLK_APLLCTL_APLLSRC_HIRC, DMIC_APLL1_FREQ_196608KHZ, CLK_APLL1_SELECT);
    // Select DMIC CLK source from APLL1_DIV2.
    CLK_SetModuleClock(DMIC0_MODULE, CLK_DMICSEL_DMIC0SEL_APLL1_DIV2, MODULE_NoMsk);
    // Enable DMIC clock.
    CLK_EnableModuleClock(DMIC0_MODULE);
    // DPWM IPReset.
    SYS_ResetModule(SYS_DMIC0RST);

    // LPPDMA Initial.
    CLK_EnableModuleClock(LPPDMA0_MODULE);
    CLK_EnableModuleClock(LPSRAM0_MODULE);
    SYS_ResetModule(SYS_LPPDMA0RST);

    /* Set multi-function pins for DMIC */
    SET_DMIC0_CLK_PB4();
    SET_DMIC0_DAT_PB5();
```

在 **main** 中，完成 **BoardInit** 之后，执行 TensorFlow Lite Micro 的初始化流程，包括准备关键词检测模型。

```
    /* Model object creation and initialization. */
    arm::app::KwsModel model;

    if (!model.Init(arm::app::tensorArena,
                    sizeof(arm::app::tensorArena),
                    arm::app::kws::GetModelPointer(),
                    arm::app::kws::GetModelLen()))
    {
        printf_err("Failed to initialise model\n");
        return EXIT_FAILURE;
    }

    constexpr int minTensorDims = static_cast<int>(
                                        (arm::app::KwsModel::ms_inputRowsIdx >
arm::app::KwsModel::ms_inputColsIdx)
```

```
                                    ? arm::app::KwsModel::ms_inputRowsIdx
                                    : arm::app::KwsModel::ms_inputColsIdx);

    const auto mfccFrameLength = 640;
    const auto mfccFrameStride = 320;
    const auto scoreThreshold  = 0.75;

    /* Get Input and Output tensors for pre/post processing. */
    TfLiteTensor *inputTensor  = model.GetInputTensor(0);
    TfLiteTensor *outputTensor = model.GetOutputTensor(0);

    if (!inputTensor->dims)
    {
        printf_err("Invalid input tensor dims\n");
        return EXIT_FAILURE;
    }
    else if (inputTensor->dims->size < minTensorDims)
    {
        printf_err("Input tensor dimension should be >= %d\n", minTensorDims);
        return EXIT_FAILURE;
    }

    /* Get input shape for feature extraction. */
    __maybe_unused TfLiteIntArray *inputShape     = model.GetInputShape(0);
#if defined (MODEL_DS_CNN)
    const uint32_t numMfccFeatures = 10;
    const uint32_t numMfccFrames = 49;
#else
#error "No model specified"
#endif

    /* We expect to be sampling 1 second worth of data at a time.
     * NOTE: This is only used for time stamp calculation. */
    const float secondsPerSample = 1.0 /
arm::app::audio::MicroNetKwsMFCC::ms_defaultSamplingFreq;

    /* Classifier object for results */
    arm::app::KwsClassifier classifier;

    /* Object to hold label strings. */
    std::vector<std::string> labels;

    /* Populate the labels here. */
    GetLabelsVector(labels);
```

内存保护单元配置（MPU）配置已在 **main** 之前的 Zephyr 初始化流程中完成。

```
    /* Set up MPU configuration
     *
     * On zephyr, configure mpu region for tensor arena in zephyr way
     * rather than direct control
     */
```

开始 DMIC 录音。

```
    ret = DMICRecord_Init(AUDIO_SAMPLE_RATE, AUDIO_CHANNEL, audioSlidingSamples,
AUDIO_SAMPLE_BLOCK);
    if (ret)
    {
        printf_err("Unable init DMIC record error(%d)\n", ret);
        delete []audioSlidingBuf;
        return EXIT_FAILURE;
    }
    DMICRecord_StartRec();
```

在 Zephyr 上，使用自定义 shell 命令控制关键词检测的录音。

```
    warn("Press 'kws next' to resume audio clip inference one-shot\n");
    warn("Press 'kws resume' to resume audio clip inference continuously\n");
    warn("Press 'kws suspend' to suspend audio clip inference\n");
    warn("Press 'kws exit' to exit program\n");
```

接着进入 super while 循环。

```
    while (1)
    {
        /* … */
    }
```

运行消息循环。这将获取如上所述的诸如“kws next”等 Shell 自定义命令，并对其进行处理。

```
        k_mutex_lock(&arm::app::kws::mutex_record_ctrl, K_FOREVER);
        if (arm::app::kws::record_ctrl_end)
        {
            k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
            warn("Bye!\n");
            break;
        }
        else if (!arm::app::kws::record_ctrl_cont && !arm::app::kws::record_ctrl_oneshot)
        {
            k_condvar_wait(&arm::app::kws::condvar_record_ctrl,
&arm::app::kws::mutex_record_ctrl, K_FOREVER);
            k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
            continue;
        }
        else if (arm::app::kws::record_ctrl_cont)
        {
            __ASSERT_NO_MSG(!arm::app::kws::record_ctrl_oneshot);
        }
        else if (arm::app::kws::record_ctrl_oneshot)
        {
            __ASSERT_NO_MSG(!arm::app::kws::record_ctrl_cont);
            arm::app::kws::record_ctrl_oneshot = false;
        }
        k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
```

运行 DMIC 录音。

```
        while (DMICRecord_AvailSamples() < audioSlidingSamples)
        {
```

```
            /*
             * On zephyr, other threads will starve with busy-wait here,
             * for example, logging thread won't work with lower priority.
             * Change to non-busy-wait.
             */
            DMICRecord_SleepUntilUpdate(200);
        }

        DMICRecord_ReadSamples(audioSlidingBuf, audioSlidingSamples);

        DMICRecord_UpdateReadSampleIndex(audioStrideSamples);
```

以及预处理、推理和后处理。

```
            if (!preProcess.DoPreProcess(inferenceWindow, audioDataSlider.Index()))
            {
                printf_err("Pre-processing failed.");
                break;
            }

            if (!model.RunInference())
            {
                printf_err("Inference failed.");
                break;
            }

            if (!postProcess.DoPostProcess())
            {
                printf_err("Post-processing failed.");
                break;
            }
```

最后，显示推理结果。

```
                for (uint32_t j = 0; j < result.m_resultVec.size(); ++j)
                {
                    info("For timestamp: %f (inference #: %" PRIu32
                         "); label: %s, score: %f; threshold: %f\n",
                         (double)result.m_timeStamp,
                         result.m_inferenceNumber,
                         result.m_resultVec[j].m_label.c_str(),
                         result.m_resultVec[j].m_normalisedVal,
                         (double)result.m_threshold);
                }
```

3. 软件与硬件需求

3.1 软件需求

- 主机操作系统：Windows 10 64 位或更高版本。

  大多数使用新唐（Nuvoton）Cortex-M 系列 MCU 的用户在 Windows 上进行开发，因此本文档以该环境为主。
  本文档中的命令行已在 Windows 的 Git Bash 环境下验证。对于其他 shell 环境，请注意不同 shell 在行续接、引号以及转义字符方面的差异。对于 Bash，行续接符为“\”。

- Python（3.10 或之后版本）

  请单独安装此工具，或按照《Developing with Zephyr Getting-Started Guide》进行安装。

- Git

  请单独安装此工具，或按照《Developing with Zephyr Getting-Started Guide》进行安装。

  本文档以 Git Bash 作为首选的命令行环境。

- CMake（3.20.5 或之后版本）和 Ninja

  请分别安装这些工具，或按照《Developing with Zephyr Getting-Started Guide》进行安装。

- Zephyr 开发环境

  在完成以上工具的安装后，请继续按照以下说明创建适配示例代码的 Zephyr 开发环境。
  创建一个名为 ***zephyrproject*** 的目录，作为 west 工作区：

```
$ cd ~
$ mkdir zephyrproject
```

  创建新的 Python 虚拟环境：

```
$ python -m venv zephyrproject/.venv
```

  激活 Python 虚拟环境：

```
$ source zephyrproject/.venv/Scripts/activate
```

安装 pip-system-certs 以解决可能发生的证书验证失败问题：

```
$ pip install --upgrade pip-system-certs
```

安装 west：

```
$ pip install --upgrade west
```

初始化名为 *zephyrproject* 的 west 工作区。该步骤会下载 *Zephyr* 源代码：

```
$ west init zephyrproject
```

将示例代码 ZIP 文件解压缩为 *EC_M55M1_Zephyr_ML_KWS*，放到 *zephyrproject* 目录中，其中已去除版本号部分，以便所示说明不受版本影响：

```
$ unzip -d zephyrproject EC_M55M1_Zephyr_ML_KWS_V1.00.zip
$ mv zephyrproject/EC_M55M1_Zephyr_ML_KWS_V1.00 zephyrproject/EC_M55M1_Zephyr_ML_KWS
```

基于示例代码的 west manifest 重新初始化名为 *zephyrproject* 的 west 工作区。这样后续的 west update 指令就不会拉入大量不必要的模块：

```
$ cd zephyrproject
$ rm -rdf .west
$ west init -l EC_M55M1_Zephyr_ML_KWS
$ west update
```

安装 Python 依赖项，其中包含 pyOCD 工具：

```
$ west packages pip --install
```

安装 Zephyr SDK 工具链 arm-zephyr-eabi：

```
$ west sdk install \
--install-base toolchains \
--toolchains arm-zephyr-eabi \
--no-hosttools
```

- 交叉编译器（GCC）

  请按照上述步骤安装此工具。

  请使用 Zephyr SDK 工具链，而非 Arm GNU Toolchain，以避免因工具链差异导致的构建失败。

- pyOCD

  请按照上述步骤安装此工具。

  Zephyr 会调用 pyOCD 来下载已构建的二进制文件。

PyOCD 目前尚未内置支持 M55M1 系列 MCU，请通过 Nuvoton 的 CMSIS Pack 添加支持：

1. 下载 Nuvoton.NuMicroM55_DFP.x.y.z.pack，其中 **x.y.z** 为最新版本。
2. 在名为 *zephyrproject* 的 west 工作区目录下，创建一个名为 *pyocd.yaml* 的 pyOCD YAML 文件，内容如下示例：

```
pack:
  - C:\Users\<USER>\Downloads\Nuvoton.NuMicroM55_DFP.<x.y.z>.pack
```

- Vela 编译器（PyPI 包）（可选）

通过 pip 安装该工具：

```
$ pip install --upgrade ethos-u-vela
```

在所有工具安装完成后，退出 Python 虚拟环境：

```
$ deactivate
```

3.2 硬件需求

- 电路组件
  - NuMaker-X-M55M1D 开发板
- 线路示意图
  - MEMS 数字麦克风

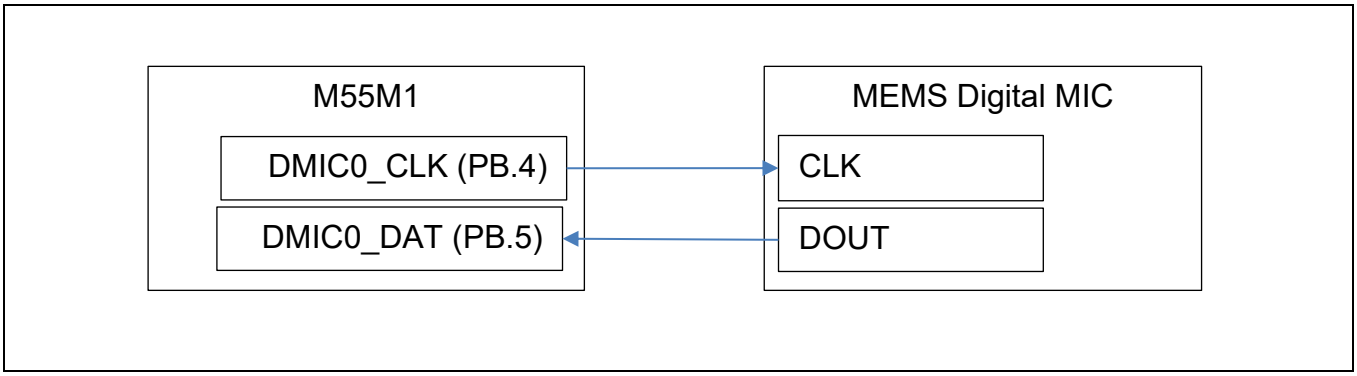

图 3-1 MEMS 数字麦克风

4. 目录信息

目录结构如下图所示。

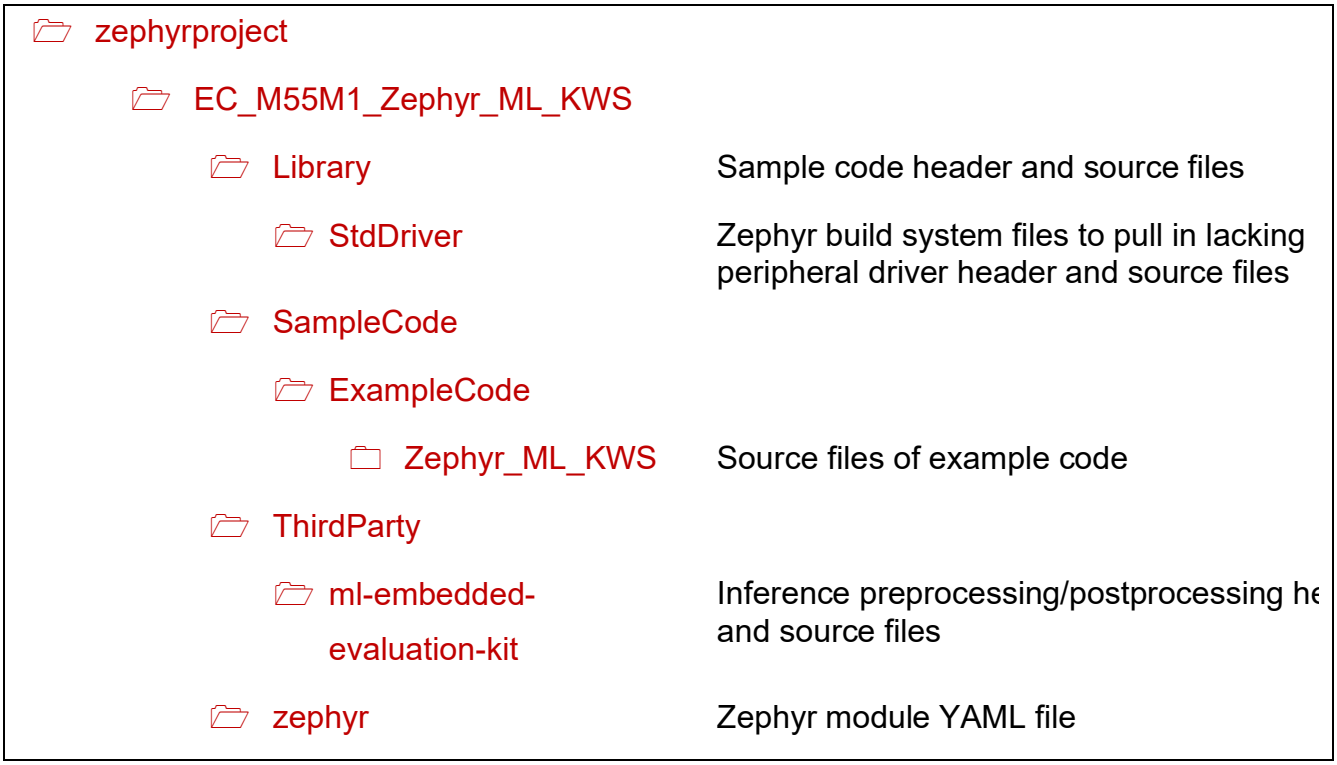

图 4-1 目录信息

5. 范例程序执行

1. 每次进行 Zephyr 开发时，请先进入 Python 虚拟环境：

```
$ cd ~
$ source zephyrproject/.venv/Scripts/activate
```

2. 进入名为 *zephyrproject* 的 west 工作区目录：

```
$ cd zephyrproject
```

3. 构建示例代码：

```
$ west -v build \
-b numaker_m55m1 \
EC_M55M1_Zephyr_ML_KWS/SampleCode/ExampleCode/Zephyr_ML_KWS
```

4. 下载已构建的二进制文件：

```
$ west flash
```

5. 在结束 Zephyr 开发时，退出 Python 虚拟

```
$ deactivate
```

6. 修订纪录

Date	Revision	Description
2026.01.02	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.