M55M1 NPU Keyword Spotting Inference

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M55M1 NPU to demonstrate machine learning keyword spotting on the Zephyr platform.
BSP Version	M55M1_Series_BSP_CMSIS_V3.01.003
Hardware	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. Overview

This example demonstrates one keyword spotting (KWS) inference application that recognizes the presence of a keyword in a recording on Nuvoton's Ethos-U NPU platform. Its source comes from Arm's ML embedded evaluation kit/Keyword Spotting example, which has more information on the keyword spotting application.

This example mainly consists of:

- Recording audio clip through digital microphone (DMIC)
- Running keyword spotting inference using the TensorFlow Lite Micro framework and the Ethos-U NPU. This will involve three steps:
  1. Preprocess recorded audio clip to specified input tensor format for inference.
  2. Run keyword spotting inference.
  3. Postprocess output tensor format usable by external applications.

This example runs a model DS-CNN Medium INT8 that has been downloaded from the Arm Model Zoo. Then this model is optimized using the Vela compiler to be runnable and accelerated on the Ethos-U NPU platform.

Vela takes a *.tflite* file as input and produces another *.tflite* file as output, where the operators supported by Ethos-U have been replaced by an Ethos-U custom operator. In an ideal case the complete model network would be replaced by a single Ethos-U custom operator.

1.1 Hardware Features

The NuMicro® M55M1 series is a new generation AI microcontroller (MCU), which delivers up to 110 GOPS of AI computing power, specifically designed for Endpoint AI such as data recognition and intelligent audio. The M55M1 series 32-bit MCU is based on a 220 MHz Arm® Cortex®-M55 core, featuring an Arm Ethos™-U55 Neural Processing Unit (NPU) as a coprocessor to accelerate neural network operations. In addition to its outstanding AI performance, the M55M1 series also offers a rich set of communication and peripheral interfaces, including digital microphone (DMIC) for audio recording. The combination of Ethos-U55 NPU and DMIC makes it suitable to show the keyword spotting inference application.

1.2 Optimizing Model Using Vela

This section instructs how to optimize the DS-CNN Medium INT8 model using Vela to the target M55M1 Ethos-U NPU.

- Download the DS-CNN Medium INT8 model named *ds_cnn_m_quantized.tflite*.
- Optimize the model *ds_cnn_m_quantized.tflite* using Vela compiler, and users can get optimized model named *ds_cnn_m_quantized_vela.tflite*.

```
$ vela ds_cnn_m_quantized.tflite \
--output-dir . \
```

```
--accelerator-config ethos-u55-256 \
--optimise Performance
```

- Rename the file to ds_cnn_m_quantized_ethos-u55-256_opt-speed.tflite for clarity and distinction.

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

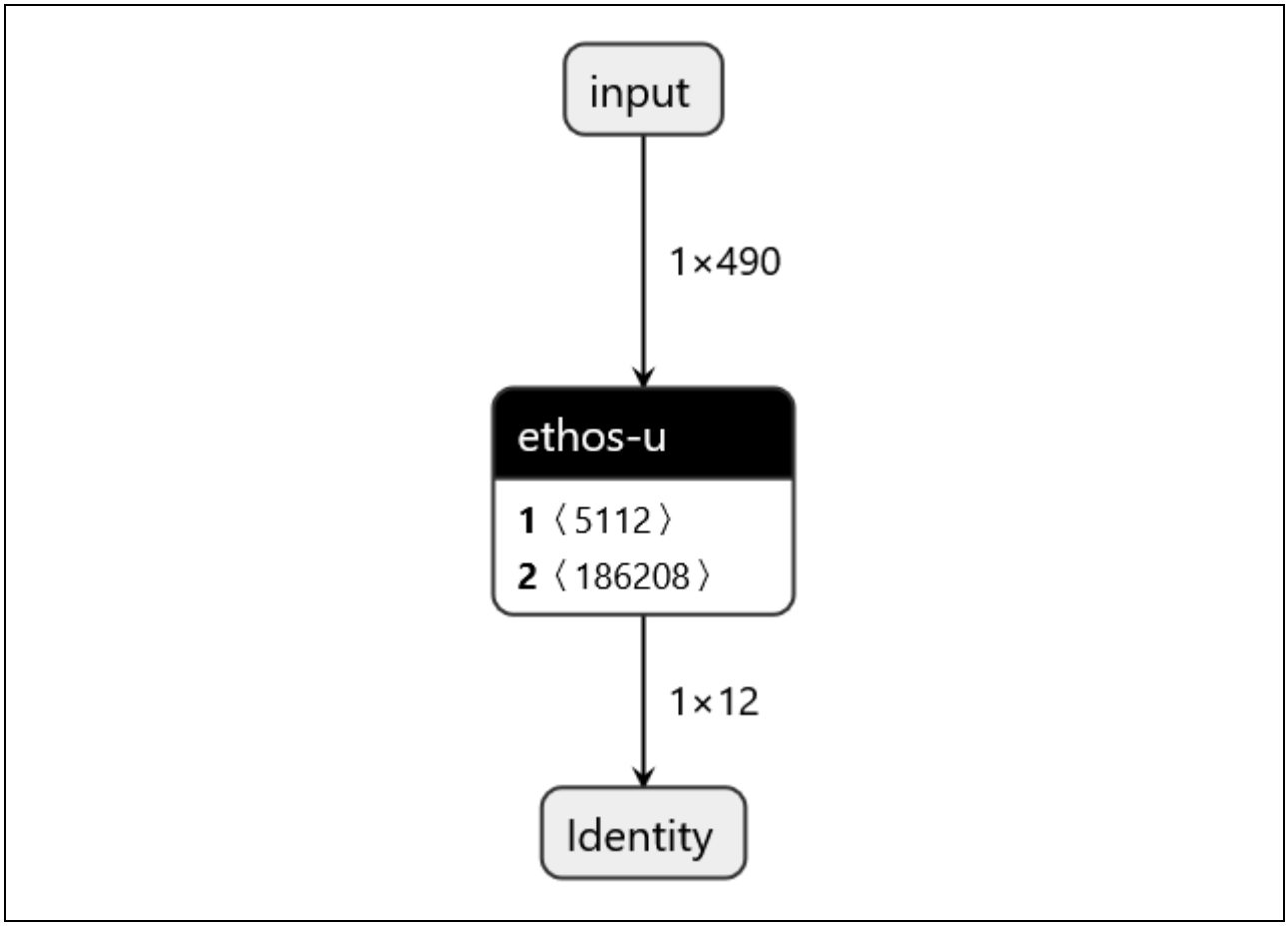

Figure 1-1 Optimized Model

1.3 Binding among Code, Model, and Ethos-U

There is a binding relationship among the example code, Vela-compiled model, and Ethos-U NPU. This means users cannot change any one of them arbitrarily, except the users have full knowledge over them.

From above, users should have noticed the binding between Vela-compiled model and Ethos-U NPU: M55M1 Ethos-U NPU is Ethos-U55, 256 macs per cycle, and the **--config** argument must be **ethos-u55-256** to match it.

The example code must be responsible for preprocessing and postprocessing for the inference. The preprocessing and the post processing logic must follow the model network input requirements and the output requirements.

1.4 Demo Result

To demonstrate the keyword spotting inference, follow the instructions in the Example Code Execution chapter and configure host terminal program with **115200/8-N-1**(115200 baud rate, 8 data bits, no parity, 1 stop bit). After **west flash** is executed, you would see message output like:

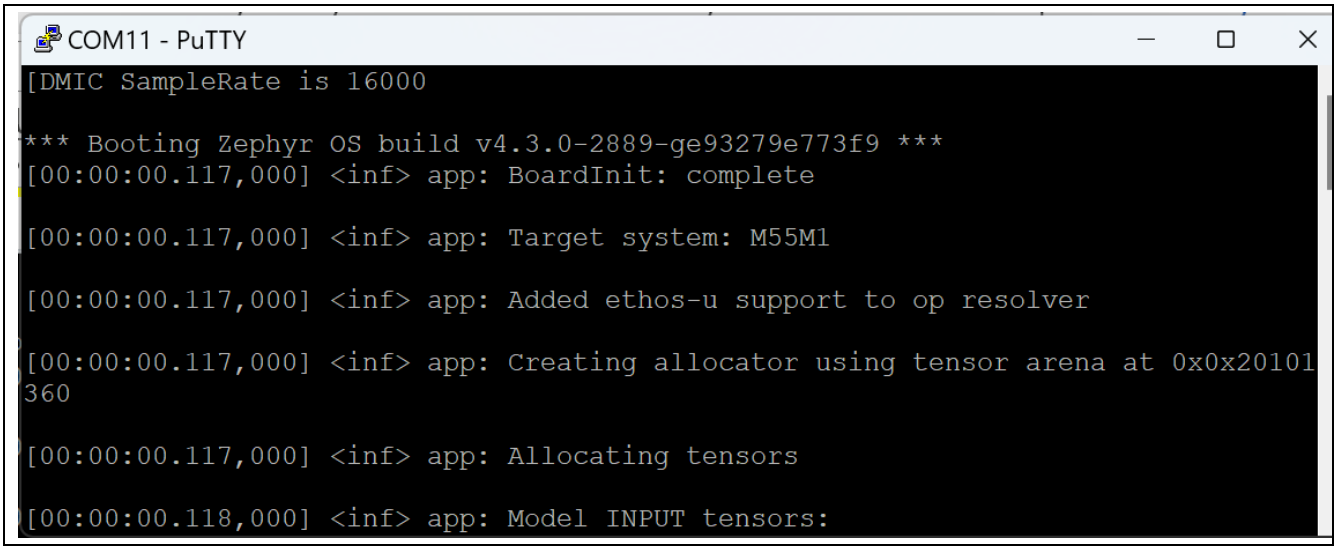

Figure 1-2 Terminal: Initialize Keyword Spotting Inference

In the end of the message output, you will be prompted to input to continue:

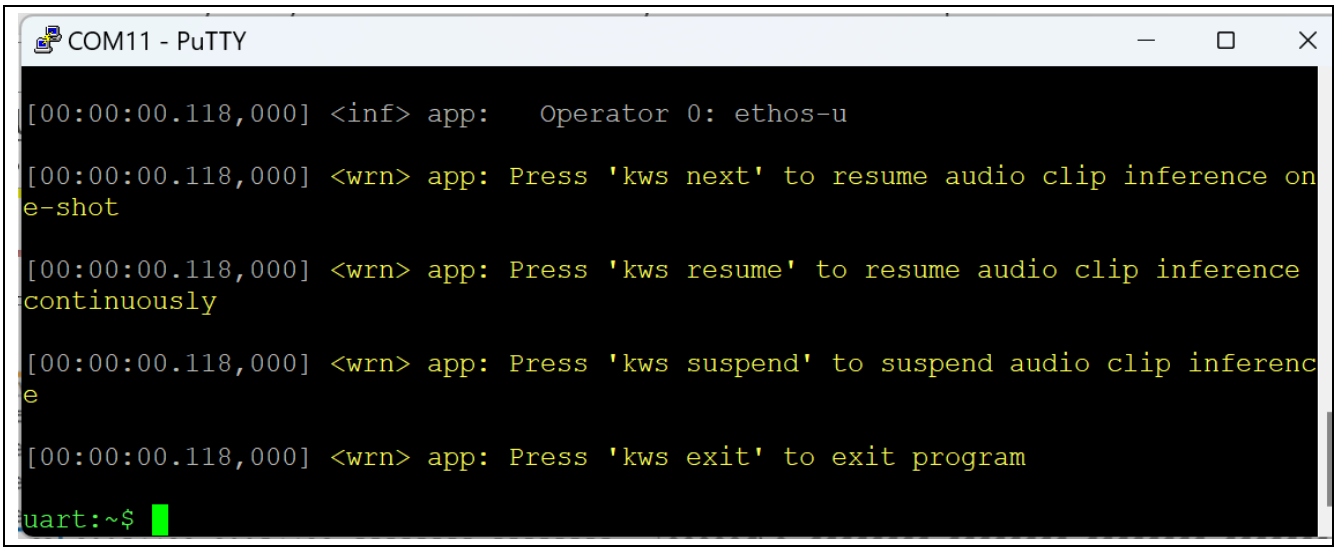

Figure 1-3 Terminal: Keyword Spotting Shell Commands

Type: kws resume

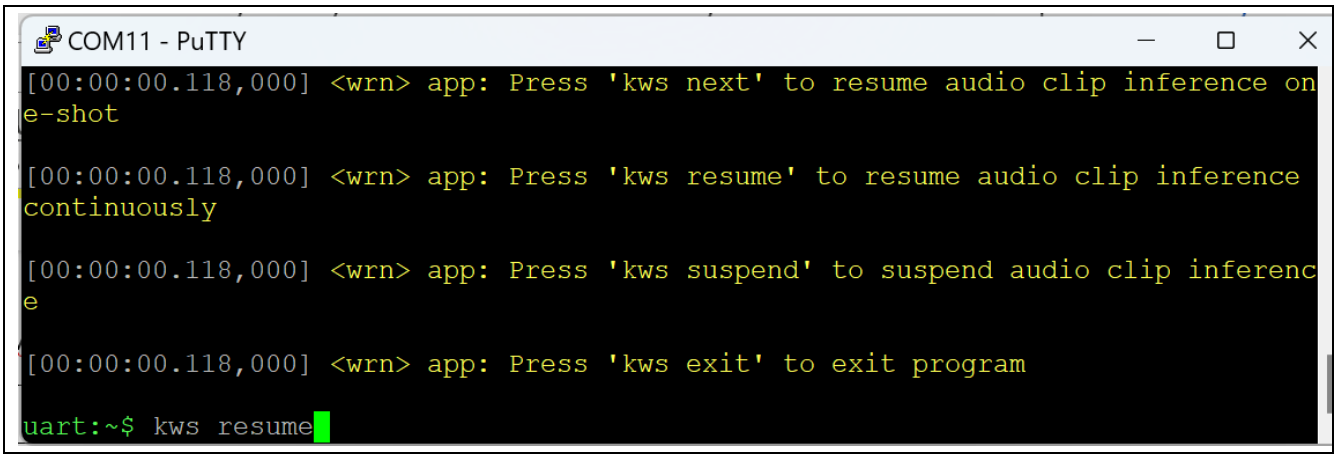

Figure 1-4 Terminal: Start Keyword Spotting Inference

You would see undetected message output:

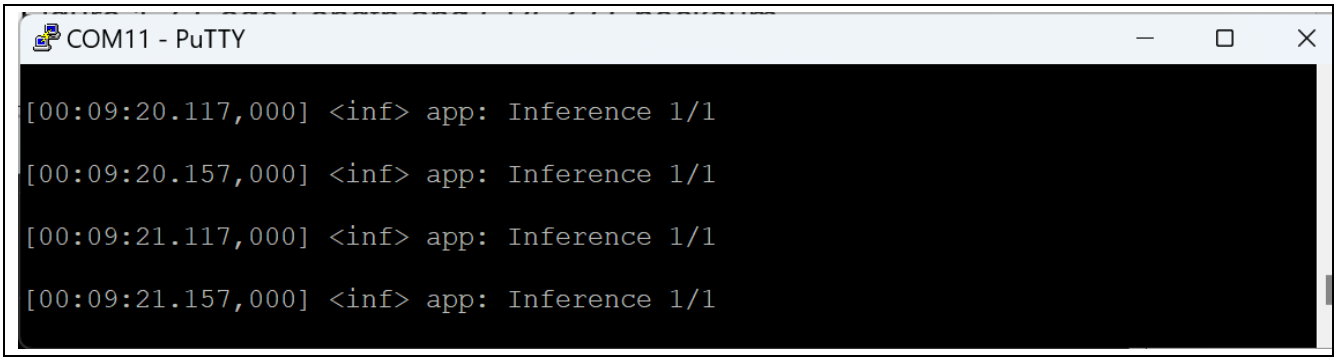

Figure 1-5 Terminal: Detect No Keyword

Approach the target board, speak trained keyword like “Go”, and you would see detected message output:

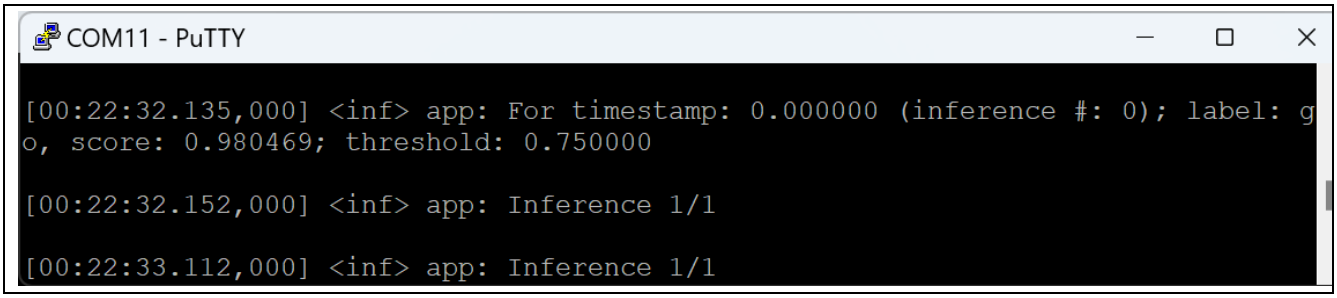

Figure 1-6 Terminal: Detect Keyword “Go”

The keywords over which the keyword spotting model is trained are listed in the file *Model/Labels.cpp*.

2. Code Description

On Zephyr, users do not need to explicitly write common initialization code such as system clock and UART because these tasks have been handled by Zephyr initialization sequence before the **main** function is executed. The Ethos-U in this example code is also initialized in this process. However, for other peripherals that are not natively supported by Zephyr, users still need to perform manual initialization in **BoardInit**.

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

In **BoardInit**, DMIC is initialized for this example code.

```
    // 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();
```

In **main**, after **BoardInit**, perform TensorFlow Lite Micro initialization sequence, including preparing keyword spotting model.

```
    /* 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);
```

MPU configuration has been done in the Zephyr initialization sequence before **main**.

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

Start DMIC recording.

```
    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();
```

On Zephyr, use the shell custom command for keyword spotting record control.

```
    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");
```

Then in super while loop.

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

Run message pump.

```
        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);
```

Run DMIC recording.

```
        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);
```

Then, preprocessing, inference, and postprocessing.

```
            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;
            }
```

Finally, show inference results.

```
                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. Software and Hardware Requirements

3.1 Software Requirements

- Host operating system: Windows 10 64-bit or later.

  Most users of Nuvoton's Cortex-M series MCU develop on Windows, so this document favors this environment.

  The command lines in this document are verified on Windows Git Bash environment. For other shell environments, check on how different shells use line continuation, quotation marks, and escapes characters. For Bash, line continuation mark is "\".

- Python (3.10 or later)

  Install the tool separately or by following the Developing with Zephyr Getting-Started Guide.

- Git

  Install the tool separately or by following the Developing with Zephyr Getting-Started Guide.

  This document favors Git Bash as CLI environment.

- CMake (3.20.5 or later) & Ninja

  Install these tools separately or by following the Developing with Zephyr Getting-Started Guide.

- Zephyr development environment

  After the above tools are installed, continue to follow the instructions below to create the Zephyr development environment that is adapted to the example code.

  Create a directory named *zephyrproject* as west workspace:

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

  Create a new Python virtual environment:

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

  Activate the Python virtual environment:

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

  Install pip-system-certs to resolve possible certificate verification failures:

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

  Install west:

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

Initialize the west workspace named *zephyrproject*. This will contain the downloading Zephyr source code:

```
$ west init zephyrproject
```

Unzip the example code ZIP file as EC_M55M1_Zephyr_ML_KWS to the *zephyrproject* directory, in which the version part is removed so that the shown instructions can be version invariant:

```
$ 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
```

Reinitialize the west workspace named *zephyrproject* based on the west manifest of the example code. This will make the instruction **west update** not pull in a lot of unnecessary modules:

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

Install Python dependencies, containing the pyOCD tool:

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

Install the Zephyr SDK toolchain, **arm-zephyr-eabi**:

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

- Cross GCC compiler

  Install the tool according to the description above.

  Use Zephyr SDK toolchain instead of Arm GNU Toolchain to avoid build failure caused by toolchain difference.

- pyOCD

  Install the tool according to the description above.

  Zephyr invokes pyOCD to download built binary.

  PyOCD doesn't have built-in support for M55M1 series MCU. Add the support via Nuvoton's CMSIS Pack:

  1. Download *Nuvoton.NuMicroM55_DFP.x.y.z.pack* where **x.y.z** is the most recent version.
  2. Under west workspace directory named *zephyrproject,* create one pyOCD YAML file named *pyocd.yaml* which has the content as follows:

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

- Vela compiler (PyPI package) (optional)

Install the tool by pip:

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

After all the tools are installed, exit the Python virtual environment:

```
$ deactivate
```

3.2 Hardware Requirements

- Circuit components

  - NuMaker-X-M55M1D board

- Pin Connect

  - MEMS digital microphone

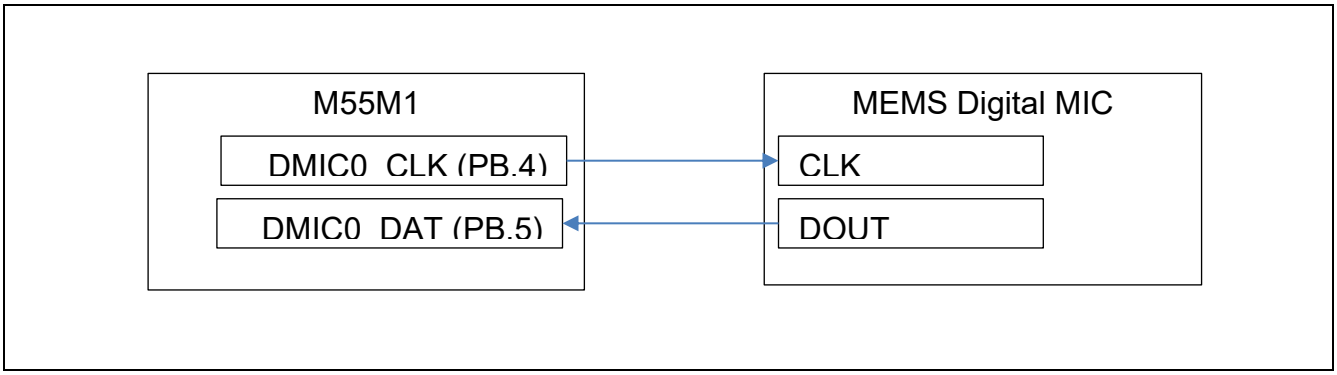

Figure 3-1 MEMS Digital MIC

4. Directory Information

The directory structure is shown below.

zephyrproject	
EC_M55M1_Zephyr_ML_KWS	
Library	Sample code header and source files
StdDriver	Zephyr build system files to pull in lacking peripheral driver header and source files
SampleCode	
ExampleCode	
Zephyr_ML_KWS	Source files of example code
ThirdParty	
ml-embedded-evaluation-kit	Inference preprocessing/postprocessing header and source files
zephyr	Zephyr module YAML file

Figure 4-1 Directory Structure

5. Example Code Execution

1. For each Zephyr development, enter the Python virtual environment first:

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

2. Enter the west workspace directory named *zephyrproject*:

```
$ cd zephyrproject
```

3. Build the example code:

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

4. Download the built binary:

```
$ west flash
```

5. To come to the end of the Zephyr development, exit the Python virtual environment:

```
deactivate
```

6. Revision History

Date	Revision	Description
2026.01.02	1.00	Initial version.

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.