*Application Brief*

# *Keyword Spotting Using AI at the Edge With Sitara Processors*

*Reese Grimsley and Tarkesh Pande* *Sitara MPU*

Time-series signals are a core sensor modality in embedded systems. Voice assistants use microphones to listen for speech and analyze the signal to recognize keywords, which can act as a trigger for short commands or internet queries. Speaker detection and beamforming enables conferencing systems to improve audio quality and reduce noise; time-series signatures like acoustic or accelerometer data can capture signs of machine wear and impending failure to motivate preventative maintenance.

Embedded applications that require human machine interaction (HMI) or industrial monitoring can benefit from local time-series processing and analysis. Machine learning and AI are effective for signal analysis and finding subtle patterns in data. This work presents two keyword spotting applications on two separate Sitara Arm microprocessors (AM62x and AM62Ax); this shows high performance and low latency.

## Time-Series Processing and Analysis for Keyword Spotting

There is a wide variety of algorithms and use-cases for time-series signals. Keyword spotting and command recognition are valuable features for HMI applications. Many of these applications use either:

- An integrated DSP for traditional signal processing (Fourier transform, resampling, digital filters),
- A small machine learning model (hidden Markov model, neural network) at the edge on a CPU/MCU, before sending signal data to a powerful cloud server that can use a much larger, more comprehensive and accurate model for speech recognition and analysis. As embedded processors improve, local compute capabilities mitigate the need for a secondary model on a cloud server, which comes at recurring monetary cost, higher latency, and reduced privacy.

Time-series analysis for keyword spotting tasks requires several preprocessing steps. This often includes techniques such as resampling, filtering, windowing, Fourier transforms [1], and Mel-frequency spectrograms and cepstrums aka MFCC [2]. In addition to traditional signal processing algorithms and statistic algorithms like independent component analysis (ICA), neural networks are increasingly being implemented for domain-specific tasks such as keyword spotting, speech transcription, speech generation, speaker recognition, and anomaly detection. Neural networks in these domains utilize a variety of techniques from CNNs, RNNs, and transformers.

## Microprocessors for Time-Series Analysis

The AMx series of industrial microprocessors (MPUs) are capable of time-series analysis using CPU resources alongside a wide variety of tasks. Processors like the AM62x and AM62A contain up to four Arm® Cortex® A53 CPUs at 1.4 GHz. For more intensive applications and algorithms, the enhanced memory capacity and speed from DDR, as opposed to limited on-chip SRAM, is crucial for audio analysis like speech transcription. Keyword spotting small dictionaries has low CPU utilization, allowing it to be run alongside other tasks.

TI's Linux Processor SDK and MCU+ SDK provide many software tools and drivers to accelerate evaluation and development. Linux is the most convenient and extensible OS for these SoCs. Debian (from SDK v9.0) eases development on select SoCs by simplifying the process of installing additional packages that are not part of the base SDK using the "apt" framework. Packages for time-series signal processing can be installed through apt on Debian, added to the Yocto build, cross-compiled from a host machine like Ubuntu, or built directly on the target; libraries specific to a programming language such as Python and Node.JS are also installable on the target through respective packaging frameworks.

The SDK includes several open source machine learning runtimes like onnxruntime and tensorflow-lite, which enables a wide variety of neural network types, including CNNs and RNNs, for analyzing audio and other types of time-series signals like accelerometer data.

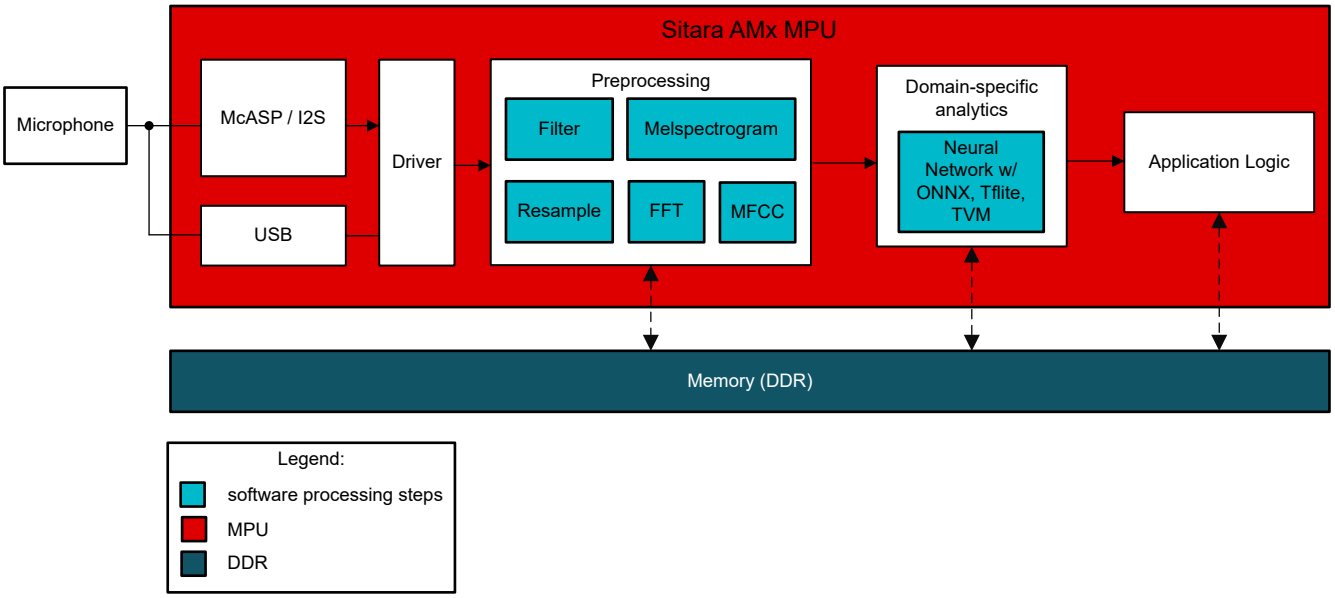


**Figure 1. Block Diagram of Audio Processing Flow for a Keyword Spotting Application Using Neural Networks on AMx MPUs**

### Keyword Spotting Application

Two demo applications are available on github [3] for CPU-based keyword spotting. Both of these are developed with Python3 on the Linux Processor SDK and leverage python libraries for sampling audio, preprocessing, and running a pretrained neural network. Speech data is brought into the SoC via a USB2.0 microphone. One demo uses matchboxnet [4] for command recognition among a 35-word vocabulary; the other demo uses a keyword spotting model from MLCommons [5] for speech recognition of a 12-word vocabulary. Each of these use similar preprocessing techniques in the Mel frequency domain. Table 1 shows performance for these two applications.

**Table 1. Processing Speed for 1 Second of Audio at 16 kHz, Including Preprocessing and Keyword Spotting Using a Convolutional Neural Network**

| Device | Runtime Specs | Preprocessing Time (MFCCs) | Network 1: Matchboxnet Inference Time (quad core) | Network 2: ML Commons Tiny KWS Inference Time (single core) |
|---|---|---|---|---|
| AM62A | DDR: 3733 MT/s, 32-bit LPDDR4<br>CPU: 1.25(1) GHz 4x A53s | 38 ms | 8 ms | 2.2 ms |
| AM62x | DDR: 1600 MT/s, 16-bit DDR4<br>CPU: 1.25 GHz 4x A53s | 40 ms | 20 ms | 2.8 ms |

(1) 1.25 GHz is the highest supported frequency when using a 0.75V core voltage. 0.75 V core voltage provides a differentiated low power capability, which is more standard than the 0.85 V required for full 1.4 GHz speed.

The preprocessing steps for these two applications involves resampling an audio stream from 48 kHz to 16 kHz and calculating the MFCC. The MFCC is calculated using a short-term Fourier transform, squaring the signal, filtering with a Mel filterbank, calculating the logarithm, and computing the discrete cosine transform. In python3, these steps were performed with a combination of numpy and librosa libraries.

Audio preprocessing benefits from multicore parallelization provided through the librosa library. However, small neural networks like these do not necessarily see improvement from parallel processing: matchboxnet saw better performance with 4 cores whereas tinyML had the best performance on single core. Note that the intent of these applications is proof of concept, and these implementations can be considered closer to the minimum achievable performance than the maximum.

In both of these models, the preprocessing time dominates because these are small neural networks that only need to determine which word was spoken among a small vocabulary. As neural networks increase in size and computational requirements, memory starts to become the bottleneck. This is why the AM62A has noticeably better performance on Matchboxnet than the AM62x due to >4x the DDR capability of the AM62x, whereas the difference is less apparent on the MLCommons network. Regardless, both tasks can easily run in real-time alongside other tasks because both applications require no more than 5% of the overall CPU resources.

## Conclusion

Time-series analytics like keyword spotting continues to be a useful component in a wide variety of applications, especially those with HMI elements. General CPU resources are sufficient in many cases for time-series signal processing, allowing such applications to run in parallel with other tasks, including other time-series analysis. For both the AM62x and AM62A processors, the high performance and low latency (<60 ms for 1s of data) is such that the time-series and audio analysis can be performed on CPU cores with plenty of resources remaining for other tasks.

## References

1. https://en.wikipedia.org/wiki/Fourier_transform
2. https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
3. https://github.com/TexasInstruments/edgeai-keyword-spotting
4. https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/commandrecognition_en_matchboxnet3x2x64_v2
5. https://github.com/mlcommons/tiny/tree/master/benchmark/training/keyword_spotting

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265