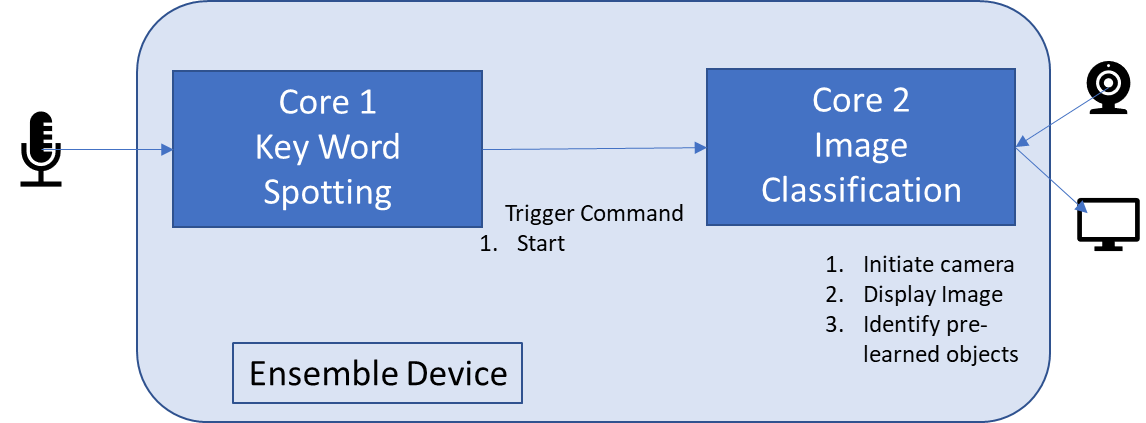
# Introduction

This is a brief set of instructions to help set up the KWS and Image Classification demo on dual M55 cores with NPU. A detailed set of app notes is being written but this is an interim quick guide for customers who need to quickly get started.

The demo consists of two applications. A Key Word Spotting (KWS) application which runs on the M55 HE core and an image classification application running on the M55 HP core. When the demo is started, the KWS application running on the M55 HE continuously listens to the audio input from the built-in microphones on the Alif base board. When the word “Go” is recognized, a command is sent to the M55 HP core to initiate the image classification application. The image classification application uses the camera to detect objects and displays the image output of the camera to a display along with the results of the image classification in real time. The M55 HE continues to listen for keywords and upon detecting the word “Stop”, a command is sent to the M55 HP core to stop the image classification.



# Required tools and hardware

* Alif CPU board
* Alif base board
* Camera
* Display
* Arm DS
* Ulink pro D

# Connecting FTDI Serial Adapters

* The bare metal getting started user guide explains how to connect an FTDI adapter to the SEUART as part of the board setup.
* For this example, you will need an adapter connected to both UART2 and UART4 for console output from each of the two M55 cores. Following the precautions in the Development Kit Quick Start Guide for setting the 1.8V jumper on the adapters, connect the UART2 and UART4 adapters as shown below.
* Connect the FTDI UART cable between the host PC and the CPU Board UART2 connection using the following UART2 pins:
* J413 pin 13 (UART2\_RX): Connects to TXDATA (blue wire or as noted above) on the FTDI cable
* J413 pin 14 (UART2\_TX): Connects to RXDATA (black wire or as noted above) on the FTDI cable
* A picture containing text, electronics

  Description automatically generatedJ418 (any pin) (GND): Connects to GND (green wire or as noted above) on the FTDI cable
* Connect the FTDI UART cable between the host PC and the CPU Board UART4 connection using the following UART4 pins:
* J412 pin 11 (UART4\_RX): Connects to TXDATA (blue wire or as noted above) on the FTDI cable
* J412 pin 12 (UART4\_TX): Connects to RXDATA (black wire or as noted above) on the FTDI cable
* J417 (any pin) (GND): Connects to GND (green wire or as noted above) on the FTDI cable

A close-up of a circuit board

Description automatically generated with medium confidence

# Building the applications

Separate applications are built and flashed to the M55 HE and M55 HP cores.

## M55 HE core KWS application

Private Repository on AlifSemi GitHub:

<https://github.com/alifsemi/ml-kws-ic-example-ensemble-share>

git clone git@github.com:alifsemi/[ml-kws-ic-example-ensemble-share](https://github.com/alifsemi/ml-kws-ic-example-ensemble-share).git

git submodule init  
git submodule update  
python set\_up\_default\_resources.py --additional-ethos-u-config-name ethos-u55-256 ( this will take some time, minutes )  
mkdir build  
cd build  
cmake -DTARGET\_PLATFORM=ensemble \  
-DTARGET\_SUBSYSTEM=RTSS-HP \  
-DCMAKE\_TOOLCHAIN\_FILE=scripts/cmake/toolchains/bare-metal-armclang.cmake \  
-DUSE\_CASE\_BUILD=kws \  
-Dkws\_MODEL\_TFLITE\_PATH=resources\_downloaded/kws/kws\_micronet\_m\_vela\_H256.tflite \  
-Dkws\_FILE\_PATH=resources/kws/samples/down.wav \  
-DCMAKE\_BUILD\_TYPE=Debug -DLOG\_LEVEL=LOG\_LEVEL\_DEBUG ..

make ethos-u-kws -j20

## M55 HP core image classification application

<https://github.com/alifsemi/ensembleML/tree/image_demo> (public repository)

python set\_up\_default\_resources.py

download\_dependencies.py

Build command

cmake -DTARGET\_PLATFORM=ensemble \

-DTARGET\_SUBSYSTEM=RTSS-HP \

-DCMAKE\_TOOLCHAIN\_FILE=scripts/cmake/toolchains/bare-metal-armclang.cmake \

-DUSE\_CASE\_BUILD=img\_class \

- Dimg\_class\_MODEL\_TFLITE\_PATH=resources/img\_class/model/mobilenet\_softmax\_v2\_1.0\_224\_uint8\_vela\_H256.tflite \

-Dimg\_class\_FILE\_PATH=resources/img\_class/samples/cat.bmp \

-DCMAKE\_BUILD\_TYPE=Debug -DLOG\_LEVEL=LOG\_LEVEL\_DEBUG ..

make -j20

# Running the demo using Arm DS debugger

The demo applications can be run using Arm Development Studio debugger.

## Running the M55 HE KWS Application

Open Arm DS

Set up debug connection to M55 HE using .axf file

Outputs to the UART2

## Running the M55 HP Image Classification Application

Open Arm DS

Set up debug connection to M55 HP using .axf file

Outputs to the UART4

# Running the demo using binaries

Convert built binary to bin format and copy to your current SETOOLS app-release folder. Do this for both applications.

* cd build
* cd bin
* fromelf -v --bin --output ethos-u-kws.bin ethos-u-kws.axf

Copy the binaries to the app release directory build/images

Note, if you see a warning, when writing the binaries to MRAM, about the binaries not being aligned then use one of the following commands to pad out the binaries.

* truncate --size /16 build/images/image\_name.bin
* python3 pad\_16bytes.py -b build/images/image\_name.bin

The next step is to generate the binary ATOC image for the two programs.  Create a new JSON file called <app-release>/build/config/kws\_img\_class\_demo.json with the following content.

{

"HP\_Image": {

"binary": "ethos-u-img\_class.bin",

"version" : "1.0.0",

"mramAddress": "0x80001000",

"cpu\_id": "M55\_HP",

"flags": ["boot"],

"signed": false

},

"HE\_Voice": {

"binary": "ethos-u-kws.bin",

"version" : "1.0.0",

"loadAddress": "0x62000000",

"cpu\_id": "M55\_HE",

"flags": ["boot", "load"],

"signed": false

}

}

Next, run app-gen-toc.py to generate the package image, which will be written to the file AppTocPackage.bin in the build directory. We will use the “-f” option to specify the input filename (kws\_img\_class\_demo.json) for the configuration file we just created. Execute this command:

* python app-gen-toc.py --filename build/config/kws\_img\_class\_demo.json

Write the applications using the SETOOLS command

* python app-write-mram.py

Outputs to the UART2 (kws) and UART4 (image classification)