Language: c

Model deployment tutorial of Himax-AIoT-NB-G3

Table of Contents
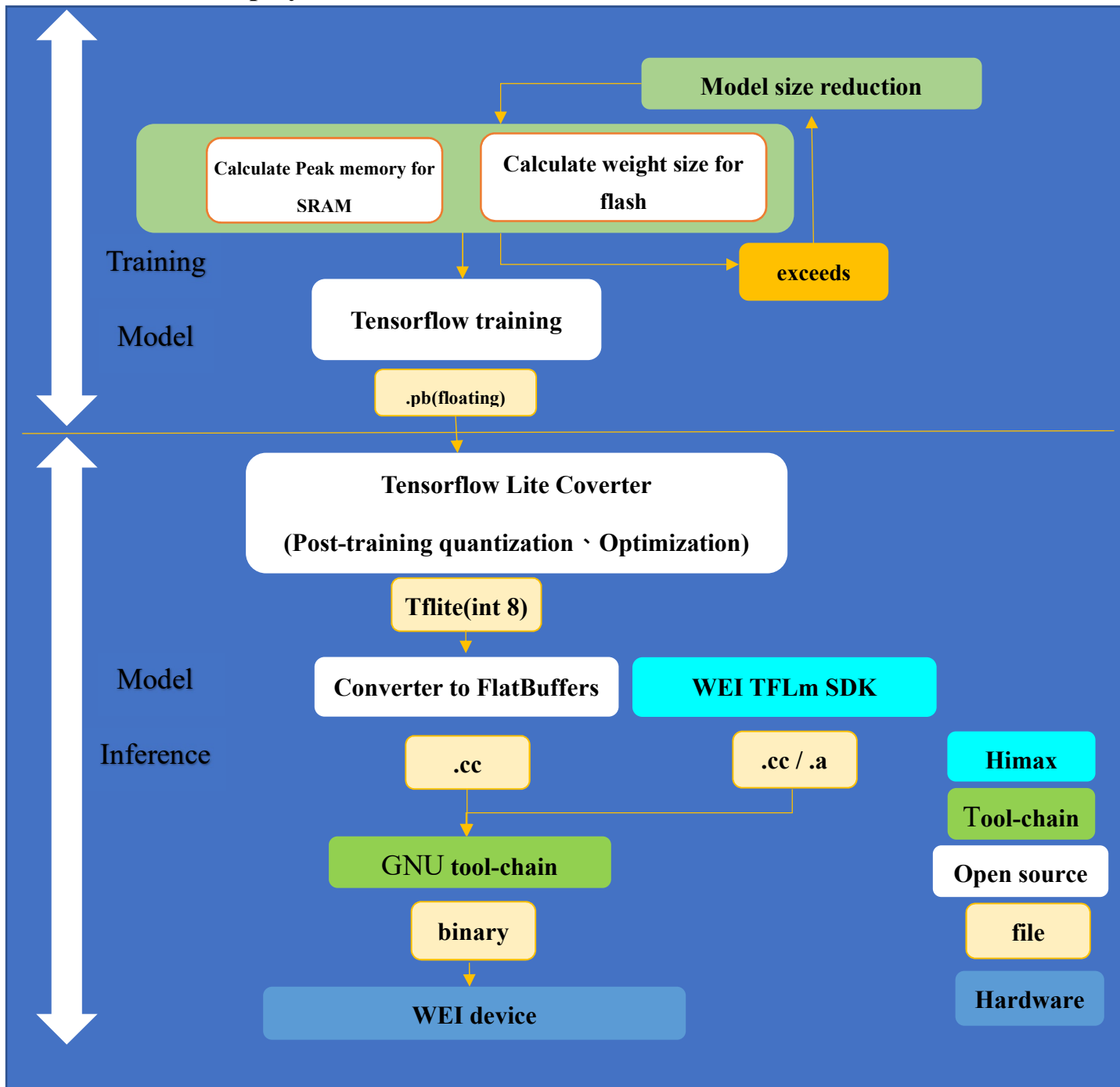
I. Model deployment tutorial of Himax WE-I Plus

❖ Himax WE-I Plus often have limited memory or computational power. Various optimizations can be applied to models so that they can be run within these constraints. In addition, some optimizations allow the use of specialized hardware for accelerated inference.

- ■ FLASH $\leq$ 600KB (Given size example)
- ■ SRAM $\leq$ 900KB (Given size example)

❖ It's recommended that you consider model optimization during your application development process. This document outlines some best practices for optimizing TensorFlow models for deployment to Himax WE-I Plus device.

❖ TensorFlow Lite for Microcontrollers(TFLµ) is a port of TensorFlow Lite(TFlite) aimed at microcontrollers and other devices with only kilobytes of memory.

II. Why models should be optimized
Size reduction

❖ Some forms of optimization can be used to reduce the size of a model. Smaller models have the following benefits:

❖ **Smaller storage size:**
- ▪ Smaller models occupy less storage space on FLASH and SRAM.

❖ **Smaller download size:**
- ▪ If your model storage in Flash, smaller models require less time and bandwidth to download to SRAM.

❖ **Less memory usage:**
- ▪ Smaller models use less SRAM when they are run, which frees up memory for other parts of your application to use and can translate to better performance and stability.

III. Model deployment workflow of Himax WE-I Plus

Training Model

Model size reduction

Calculate Peak memory for SRAM

Calculate weight size for flash

exceeds

Tensorflow training

.pb(floating)

Model Inference

Tensorflow Lite Coverter

(Post-training quantization、Optimization)

Tflite(int 8)

Converter to FlatBuffers

WEI TFLm SDK

.cc

.cc / .a

GNU tool-chain

binary

WEI device

Himax

Tool-chain

Open source

file

Hardware

IV. Training Model: Calculate Peak memory for SRAM

- Beforetraining model and converting to TensorFlow Lite for microcontrollers(TFLµ)format, we need to calculate the model's peak memory.We will use Object detection MobileNetv2-SSD as an example to training and inference.

- https://www.tensorflow.org/lite/examples/object_detection/overview
  (Pretrain model size: 27MB)

MolibeNetV2-Single Shot Multibox Detector (SSD)
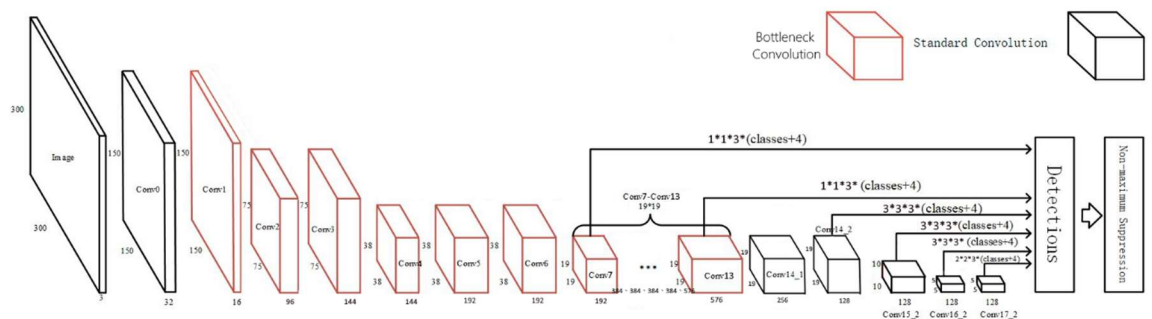
- The purpose of the calculate peak memory is to set the tensor arena size for TFLµ.
- How to calculate the peak memory?
  - ❖ Peak memory is to calculate each convolution layer and select the maximum memory consumption layer.
  - ❖ Each layer of convolution consumes memory include:
    - ❖ Input tensor size
      - ▪ Image width x image height x input channel
    - ❖ Output tensor size
      - ▪ Image width x image height x output channel

● **For example: MobileNetV2 –SSD:**

  ❖ Take 2nd bottleneck layer as example, which is composed with three stacked operations. The weighting is shown as below.

2nd bottlenecks layer	Input channel	Krenel size	Stride	Output channel
Operation 1	16	1*1	1	96
Operation 2	1	3*3	2	96
Operation 3	96	1*1	1	24

  ➢ According to the input and output tensor formula. Each stacked operation memory usage shown as below

2nd bottlenecks layer	Input tensor	Output tensor	Memory
Operation 1	150*150*16 =360000 Bytes	150*150*96 =2160000 Bytes	360000+2160000 =2520000 Bytes
Operation 2	150*150*96 =2160000 Bytes	75*75*96 =540000 Bytes	2160000+540000 =2700000 Bytes
Operation 3	75*75*96 =540000 Bytes	75*75*24 =135000 Bytes	540000+135000 =675000 Bytes

  ➢ The operation 2 also has the largest memory usage compares to other layer operation, which can be represent as peak memory.

● **Using tflite-tools Double confirm**

  ❖ Even though the peak memory can be calculated manually. It's recommended that using tflite-tools for cross-checking. Double confirm the peak memory calculation.

  ❖ You can reference the URL below to download and learn how to operate the tflite-tools.

    ◆ https://pythonawesome.com/tflite-model-analyzer-memory-optimizer/

  ❖ Here are the command line if you already have the tflite-tools.

    ◆ Command

```
• pipenv shell
• python tflite_tools.py -i MobileNetv2-SSD.tflite
```

Training Model: Calculate Weights size for Flash

- **How to calculate the Weights size?**
  - ❖ For example: MobileNetV2-SSD
  - ▪ Weights size is the sum of the kernel weighting size of stacked operation in each conv layer of model.
    - ➢ Input channel x kernel width x kernel size x output channel
  - ▪ Below chart shows each kernel weighting size of stacked operation.

Input channel	Kernel width	Kernel height	Output channel	Weighting	weights size
3 Bytes	3 Bytes	3 Bytes	32 Bytes	3*3*3*32=864 Bytes	864+…+34560=2777616 Bytes
…	…	…	…	…	
256 Bytes	3 Bytes	3 Bytes	15 Bytes	256*3*3*15= 34560 Bytes	

- The calculation values of weights size can be also obtained by adding model.summary() in python code, which will show the "Total params" of your model.

```
reshape_7 (Reshape)        (None, 1083, 15)    0    conv2d_23[0][0]
reshape_8 (Reshape)        (None, 300, 15)     0    conv2d_24[0][0]
reshape_9 (Reshape)        (None, 75, 15)      0    conv2d_25[0][0]
reshape_10 (Reshape)       (None, 27, 15)      0    conv2d_26[0][0]
reshape_11 (Reshape)       (None, 12, 15)      0    conv2d_27[0][0]
concatenate_1 (Concatenate) (None, 5829, 15)   0    reshape_6[0][0]
                                                     reshape_7[0][0]
                                                     reshape_8[0][0]
                                                     reshape_9[0][0]
                                                     reshape_10[0][0]
                                                     reshape_11[0][0]

Total params: 3,096,270
Trainable params: 3,080,078
Non-trainable params: 16,192
```

> **You may see the total params value is larger than manual calculation.**
> **The weight size value calculate by both methods will eventually smaller than array size of .cc file.**

- Both calculation methods are estimated size only. You need to make sure the size is less than FLASH limited memory.
- The actual weights size will be determined by the array size of .cc file after converting the trained model.

- **The target of reducing the size is to make the model use less memory, which frees up memory for other parts of your application to use and can translate to better performance and stability. There are two main ways:**
 - ❖ Resize input resolution
  - ❖ Change parameter(For the type of model you use)
- **Example: MobilenetV2 SSD**
  - ❖ Resize Input: 300*300*3 ->192*192*1->192*192*1
  - ❖ Change alpha: 1   ->   0.5   ->   0.25
  - ❖ Peak memory: 2700kb -> 553kb ->   276kb
  - ❖ Weight size: 2777kb -> 751kb ->   216kb

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Imagenet Accuracy vs Million Parameters
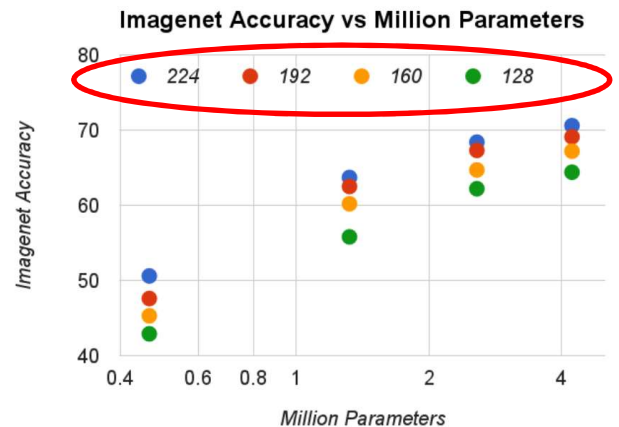
- **Trade-offs**
  - ❖ Optimizations can potentially result in changes in model accuracy, which must be considered during the application development process.
  - ❖ The accuracy changes depend on the individual model being optimized and are difficult to predict ahead of time.
  - ❖ Generally, models that are optimized for size or latency will lose a small amount of accuracy. Depending on your application, this may or may not impact your users' experience.
  - ❖ In rare cases, certain models may gain some accuracy as a result of the optimization process.

V. Model Inference: Tensorflow Lite Converter

- The purpose of Tensorflow Lite Coverter is to convert the model to standard TFLite format with TensorFlow Lite converter.

  ❖ Using post-training quantization (PTQ) statically quantizes only the weights from floating point to integer, which has 8-bits of precision.

  ❖ Reference code is shown as below.

```python
import tensorflow as tf
def representative_dataset_gen_MNIST():
    array = test_x.astype(np.float32)
    yield ([array])
# Get pd file
converter = tf.lite.TFLiteConverter.from_saved_model(saved_pd_model_dir)
# Set the optimization flag.
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset_gen_MNIST
# Enforce integer only quantization
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
# Set inference I/O data type
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8
model_tflite = converter.convert()

# Save the .tflite to disk
open('saved_model_dir/MobileNetv2-SSD.tflite', "wb").write(model_tflite)
```

calibrate or estimate the range, i.e, (min, max) of all floating-point tensors in the model

ensure compatibility with WE-I PLUS

- The trained model will store as FlatBuffers format with .tflite filename extension after TensorFlow Lite Convertor.

- Due to WE-I PLUS don't have native filesystem support. Convert FlatBuffers format into C byte array format with .cc filename extension for compile with WE-I PLUS is needed.

- Different operation system has different convert method.

  ❖ Linux

    ▪ Input following command.

    ```
    $ xxd -i MobileNetv2-SSD.tflite MobileNetv2-SSD.cc
    ```

  ❖ Windows

    ▪ Download package from NLUUG.

    ▪ Place .flite file at same directory as xxd.exe.

    ▪ cd to xxd.exe location.

▪ Input following command.

```
xxd.exe -i MobileNetv2-SSD.tflite > MobileNetv2-SSD.cc
```

● After converting to .cc file, the array length 395736Byte from .cc file is the actual weights size of the model, which is also the size that store in FLASH or SRAM. Previous calculate method are estimated value only.

```
32976      0x02, 0x00, 0x00, 0x00, 0xf0, 0xff, 0xff, 0xff, 0x04,
32977      0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0c,
32978      0x0f, 0x00, 0x00, 0x00, 0x08, 0x00, 0x04, 0x00, 0x0c,
32979      0x03, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00,
32980    };
32981    unsigned int __255MobileNetv2_SSD_tflite_len = 395736;
32982
```

● Following is the weights size value difference between different calculation method and array length from .cc file of the model MobilenetV2 SSD.
  ❖ Manual calculation => 264184Byte
  ❖ model.summary() API => 273030Byte
  ❖ FlatBuffers(.cc file) => 395736Byte

VI. Model Inference: Converter to FlatBuffers

■ The trained model is converted into the TensorFlow Lite format and the result is a FlatBuffers format that is a model file with .tflite filename extension

■ The target is convert the TFLite format to a C Byte Array for TFLµ.
  ❖ The following command shows how to convert the FlatBuffers into a C array of a tflite model by linux builtin tool 'xxd' .and output .cc file

```
$ xxd -i MobileNetv2-SSD.tflite MobileNetv2-SSD.cc
```

  ▪ In windows system, you can use the API provided by NLUUG.

● Due to the large difference between the .cc file and the estimated
value, to be safe, you can also train for 1 ~ 2 epoch and then export
the .cc file, and determine the actual size in advance
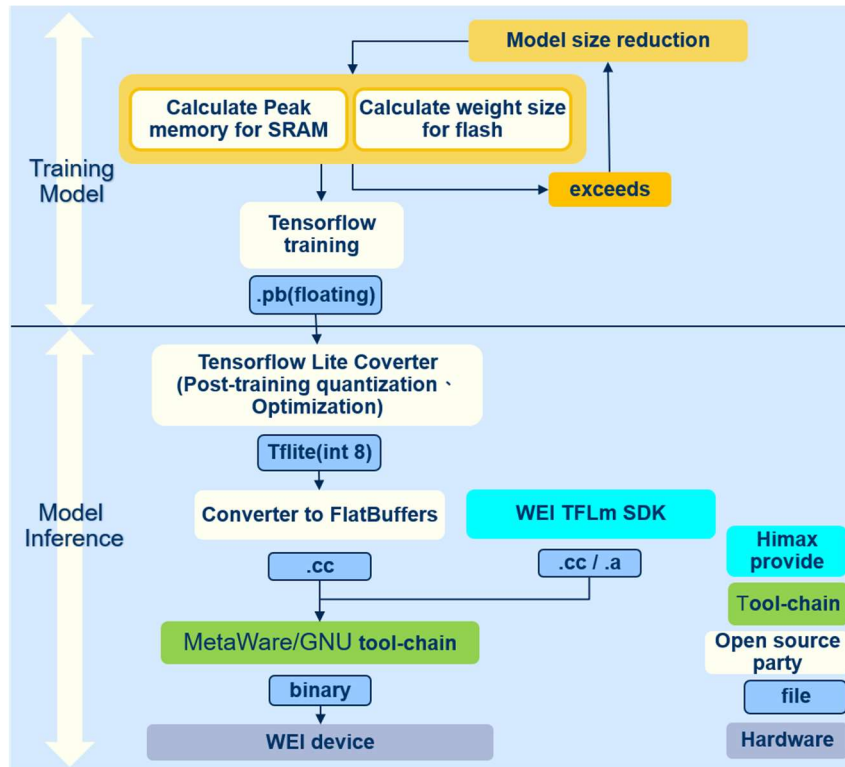
VII. Model Inference: WEI TFLm SDK

> Binaries

> Archives

> Includes

> app   → **Scenario APP(Example code)**

> arc

> board   → **Board Configurations**

> def_linker_script

> external   → **External Device**

> inc

> library   → **Prebuilt Lib**

> middleware

> obj_socket_24   → **Out Folder**

> options   → **Build File and Build Rule**

> platform   → **Platform IP,Peripheral and sys**

makefile

```
cv
  tflitemicro
    bss
    examples
    linker_script
    prebuilt_lib
      libtflitemicro_google_person.a
    tensorflow
```

TFLite_example
- Binaries
- Archives
- Includes
- app
- arc
- board
- def_linker_script
- external
- inc
- library
  - audio
  - clib
  - common
  - cv
    - tflitemicro
      - bss
      - examples
        - person_detection_experimental
          - MobileNetv2_SSD_model_data.cc
      - linker_script
        - linker_template_mw_tflitemicro.ld
      - prebuilt_lib
      - tensorflow
      - third_party
      - tflitemicro.mk
      - tflitemicro.mk.bak

Model .cc file

Linker Script

TFLμ SDK

TFLμ Makefile

VIII. Model Inference: GNU tool-chain

- The target of GNU tool-chain describe how to build binary code on glance at the model inference code.

```
##
# IC package select : QFN72
##
IC_PACKAGE_SEL = QFN72
#IC_PACKAGE_SEL = LQFP128
##
# Set toolchain
##
#TOOLCHAIN ?= mw
TOOLCHAIN ?= gnu
```

- **About MetaWare and GNU tool-chain:**
  - ❖ MetaWare and GNU tool-chain enable users to efficiently build, debug, profile and optimize their embedded software applications for ARC.
  - ❖ MetaWare tool-chain has more complete support tools available for use, but a license is required.
  - ❖ You can also use the GNU IDE and GNU tool-chain for free. Its will not affect the next steps tutorial.
    - ◆ Install "arc_gnu_2020.09_ide_win_install.exe" to use GNU tool-chain.
- **Model .cc file**
  - ❖ "/TFLite_example/library/cv/tflitemicro/examples/person_detection_experimental/ MobileNetv2_SSD_model_data.cc"

    `const unsigned char MobileNetv2_SSD_model_data[]`
  - ❖ You can replace with the new model .cc file you generated.
- **TFLμ library**

  tflitemicro_algo.cc

```c
#include <library/cv/tflitemicro/tensorflow/lite/c/common.h>
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"
```

- **The mobilenetV2 SSD model requires at least 320KB(276 KB Peak memory+44KB) of tensor arena to store all runtime resources.**
  tflitemicro_algo.cc

  ```
  constexpr int tensor_arena_size = 350*1024;
  ```

  - ❖ **44KB contains model input tensor and output tensor.**
    - ■ **The input image shape of this model is [192, 192] and output shape is [486,45].**
  - ❖ **layerWidths= [12,12,12,6,3,3]**
  - ❖ **numBoxes=[3,3,3,3,3,3]**
  - ❖ **0~9 + background + cx + cy + h + w=15**
    - ▪ **layerWidths^2=486**
    - ▪ **numBoxe *15=45**

  tflitemicro_algo.cc

  ```
  #define HIMAX_INPUT_SIZE_X (192)
  #define HIMAX_INPUT_SIZE_Y (192)
  ```

- **Allocate runtime resources.**
  tflitemicro_algo.cc

  ```
  interpreter->AllocateTensors();
  ```
- **Run the model inference.**
  tflitemicro_algo.cc

  ```
  TfLiteStatus invoke_status = interpreter->Invoke();
  ```

- **Run-time memory consumption using SRAM**
  - ❖ **Set SRAM consumption based on tensor arena size.**
    linker_template_mw_tflitemicro.ld
  - ❖ **SRAM parameter**
    - ▪ **EXT_RAM_APPDATA_START = 0x20020800**
    - ▪ **Change parameter SYSTEM0 to resize SRAM consumption.**
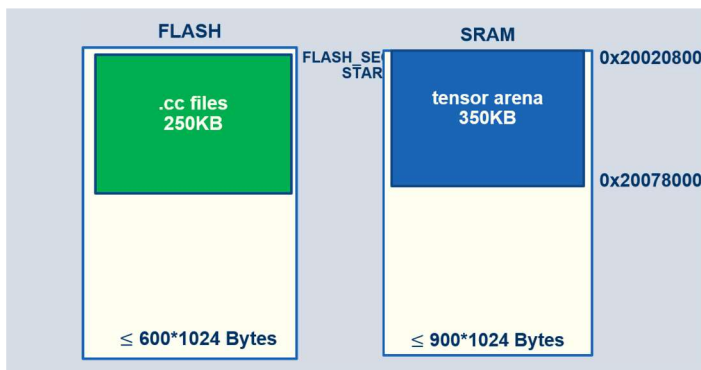   - ❖ **LENGTH =350*1024 Bytes (≤ 900*1024 Bytes) = 0x57800**

```
#if (EXT_RAM_SIZE != 0)
  SYSTEM0 : ORIGIN = EXT_RAM_APPDATA_START, LENGTH = 0x57800
```

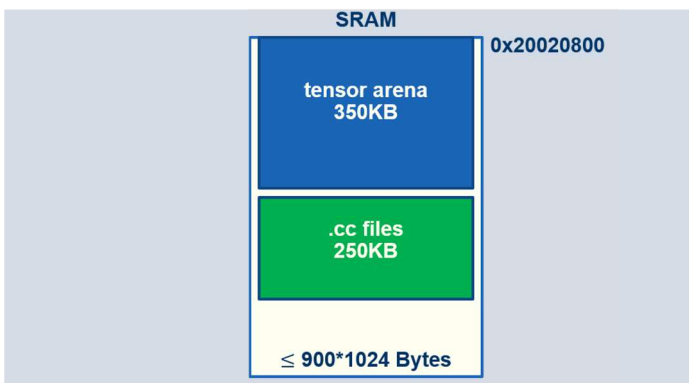**SRAM**

tensor arena 350KB	0x20020800
	0x20078000
≤ 900*1024 Bytes	
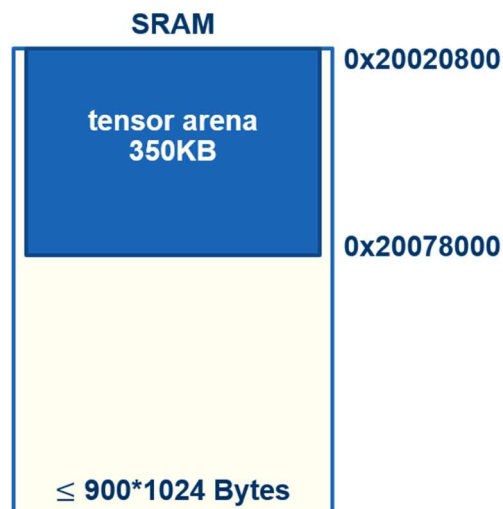
- **Model .cc files are stored in Flash or SRAM.**
  linker_template_mw_tflitemicro.ld

  ❖ **Select Flash or SRAM(define、undefine FLASH_AS_SRAM)**

```
#ifdef FLASH_AS_SRAM
        GROUP : {
    .rodata ALIGN(4): {
        MobileNetv2_SSD_model_data.o (TYPE rodata)
            }
    } > FLASH_SECTOR
#else
    GROUP BLOCK(4): {
            .rodata.MobileNetv2_SSD_model_data?:{}
    } > SYSTEM1
#endif
```
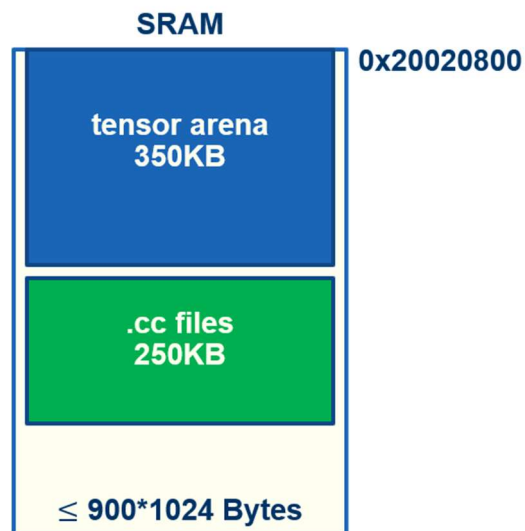

OR

❖ For SRAM, change parameter SYSTEM0 to resize SRAM
consumption.

       linker_template_mw_tflitemicro.ld

▪ LENGTH =350*1024 Bytes +220*1024 Bytes ($\leq$
900*1024 Bytes) = 0x8E800

```
#ifndef FLASH_AS_SRAM
  SYSTEM0 : ORIGIN = EXT_RAM_APPDATA_START, LENGTH = 0x8E800
#endif
```

SRAM

0x20020800

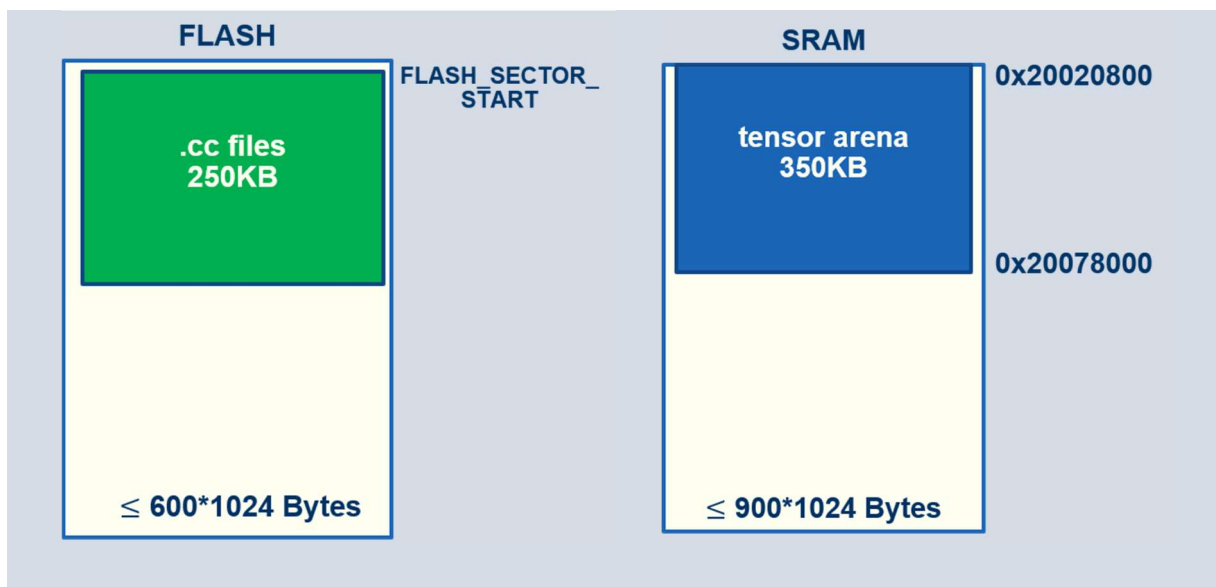tensor arena
350KB

.cc files
250KB

$\leq$ 900*1024 Bytes

❖ For Flash, change parameter FLASH_SECTOR to resize Flash
  consumption.

  linker_template_mw_tflitemicro.ld

  ▪ LENGTH =220*1024 Bytes ($\leq$ 600*1024 Bytes) =
    0x37000

```
#ifdef FLASH_AS_SRAM
  FLASH_SECTOR : ORIGIN = FLASH_SECTOR_START, LENGTH = 0x37000
#endif
```

FLASH   **SRAM**

FLASH_SECTOR_   0x20020800
START

.cc files   tensor arena
250KB   350KB

                       0x20078000

$\leq$ 600*1024 Bytes $\leq$ 900*1024 Bytes

● Using Metaware IDE output elf and map file.

● Using Image Generator tool convert elf 、map file to image file.

IX. Model Inference: Deploy on the WE-I EVK

- The target of Deploy on the WE-1 device is communications between PC and WE-1.
- About the Code
  - ❖ In MetaWare/GNU tool-chain project detect_meta.h
    - ▪ for spi write
  - ❖ In HMX_FT4222H_GUI Visual Studio project alg_meta.h
    - ▪ for spi read
- Build HMX_FT4222H_GUI
- Flash Download and execute spi slave read inference result.
- Baud rate 115200
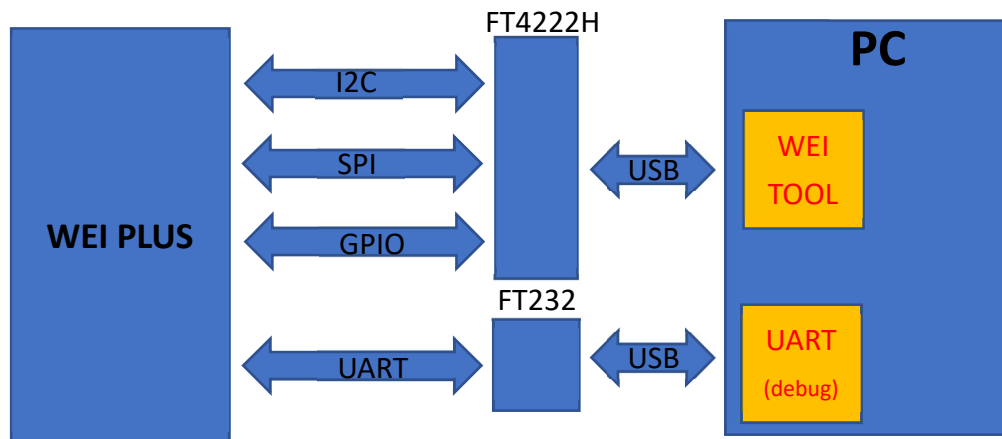
Communications between PC and WEI

Note: If you need any more information please reference the link.

■   https://www.himax.com.tw/products/intelligent-sensing/always-on-smart-sensing/

■   https://www.himax.com.tw/products/intelligent-sensing/always-on-smart-sensing/inquiry-form/