Tengine on STM32F7 Setup Manual

最新版日期 2019-06-11

OPEN AI LAB

变更记录

日期	版本	说明	作者
2019-06-11	0.1.0	初版	Bin Hu

目录

1 概述

Tengine 是 OPEN AI LAB 针对前端智能设备开发的软件开发包，核心部分是一个轻量级，模块化，高性能的 AI 推断引擎，并支持用 DLA、GPU、xPU 作为硬件加速计算资源异构加速。

针对使用更加广阔成本更低的嵌入式设备，OPEN AI LAB 推出了 Tengine-lite 软件开发包。相较于 Tengine，Tengine-Lite 占用内存更小（编译后的代码大小仅为 25K 左右），运行耗费资源更少（三层卷积网络模型运行内存仅占用 13K 左右）。Tengine-Lite 现在已经可以运行在 ARM-Cortex M7 系列芯片上。

本文档将详细说明 Tengin-Lite 如何在 STM32F769（Arm Cortex M7）芯片上，并通过对两个语音文件的识别演示 Tengine-Lite 的使用流程。

2 环境搭建

本章节将介绍运行 demo 工程所需要的软硬件环境。

2.1 硬件环境

为方便用户理解和使用 tengine，我们采用 ST 公司提供的 STM32F769I-DISCO 板作为演示硬件平台。用户亦可以采用 STM32F769 为主处理芯片的其他板子来运行 tengine，只需要对相应的交互接口做移植。

硬件准备：

1，Mirco USB 电缆线一根

2，意法半导体（ST）开发的 STM32F769I-DISCO 开发板一块

3，MicroSD 卡一张

关于硬件开发板更加详细的资料介绍可以参考文档：UM2033 User manual Discovery kit with STM32F769NI MCU。STM32F769NIH6 主频率可达 216MHz，462MDIPS，内置 2MB Flash，532KB RAM，包括：512KB 数据 SRAM、16KB 指令 TCM RAM（时间关键程序内存）和 4KB 备份 SRAM。

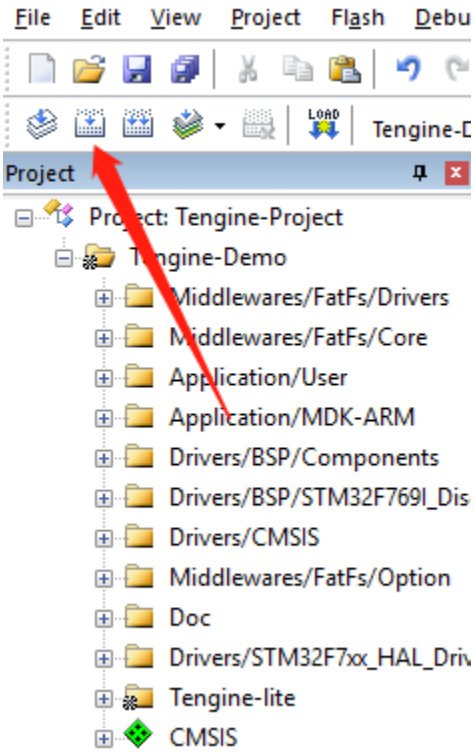
为确保收到的开发板是可以正确运行的，可以参考 http://www.cirmall.com/bbs/thread-105851-1-1.html 文档，下载官方提供的工程文件，验证开发板的功能正确性。

2.2 软件环境

我们提供了一个 Demo 工程，包括 Tengine-lite 源码、编译配置等信息。用户可以通过 git 下载工程文件。建议直接使用下述所列指令来下载 Demo 工程源码：

```
git clone --recurse-submodules ssh://git@192.168.234.8:8022/internal/cortex-m-demo.git
```

下载完成后，无需要任何更改配置，重新编译即可运行 Demo 工程。



2.3 开发工具

Demo 工程文件是用 Keil IDE 开发的，我们推荐使用 ARM 公司的 MDK Keil IDE 作为开发环境，推荐安装版本为 5.2.5。软件具体的安装步骤此处不再累述。

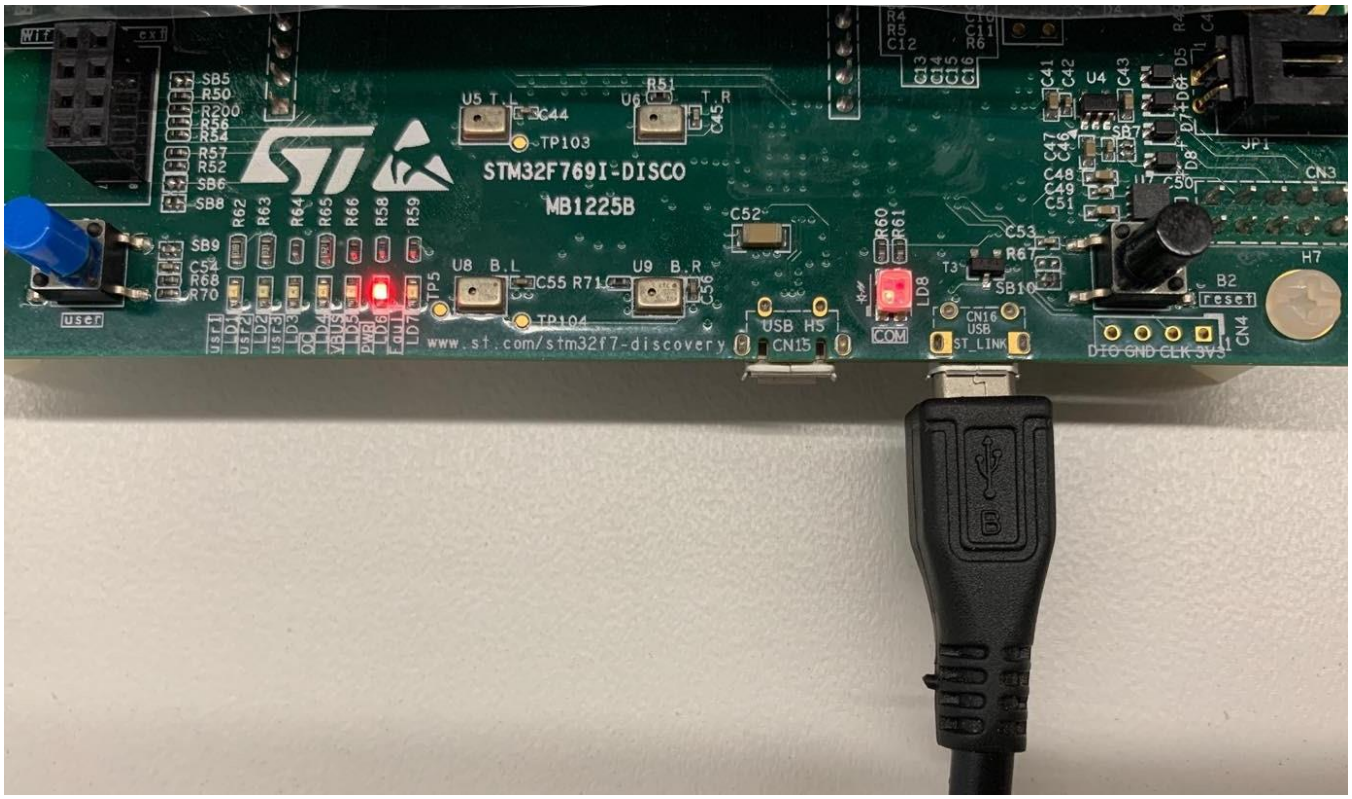
3 测试验证

3.1 音频测试文件准备

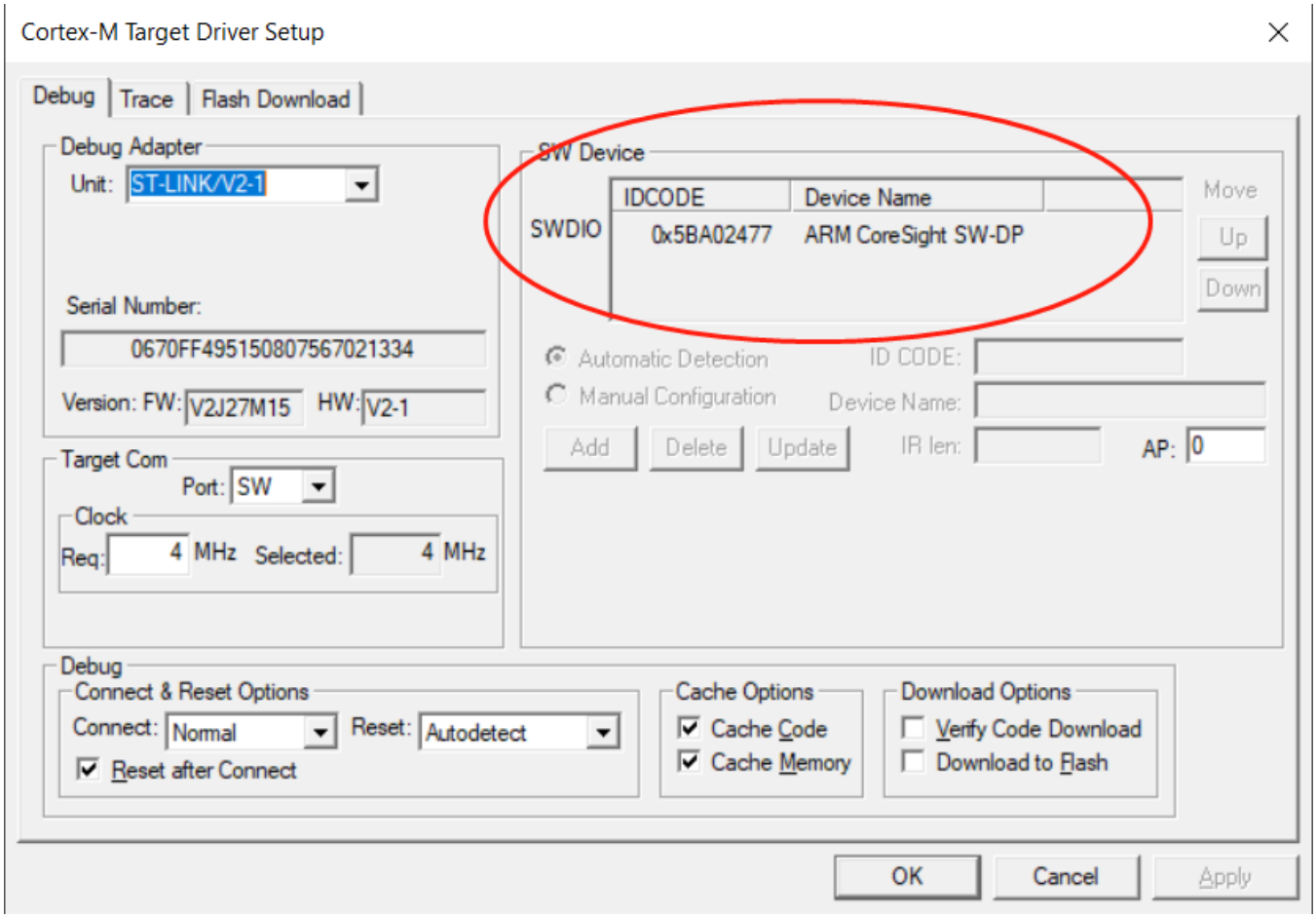
将 Demo 工程根目录下 WAV 文件夹里的三个 wav 文件 0.wav 、1.wav、2.wav 拷贝到 Micro SD 里。三个音频文件分别对应一段人声“小智”、一段小品、一首歌曲。拷贝完成后，再将 Mirco SD 卡插入到开发板的 SD 卡槽里。

3.2 硬件连接

将 Micro USB 线一端接到开发板上的 ST-Link 接口，另外一端插入电脑 USB 口。

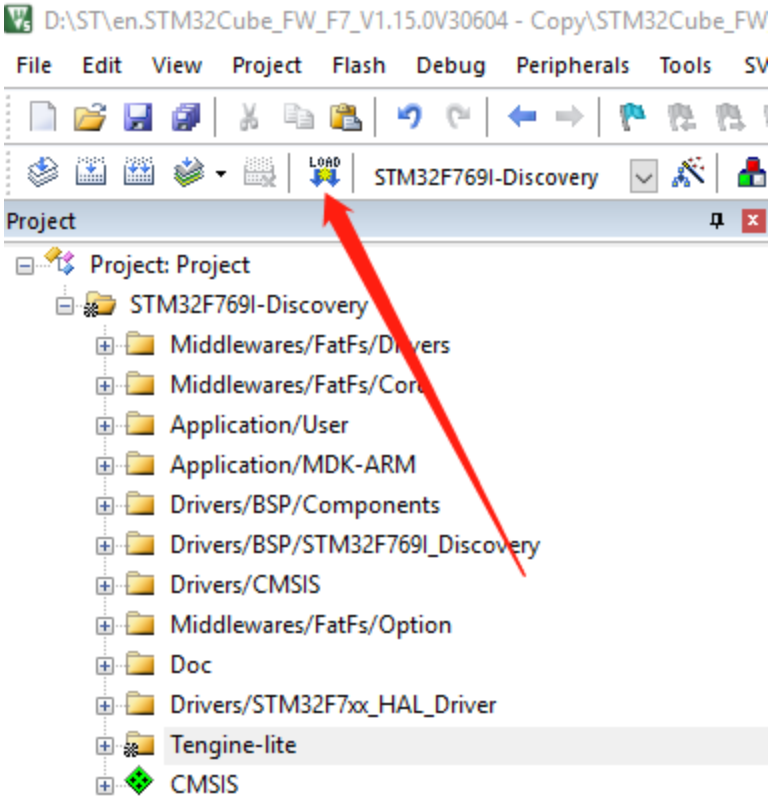


线缆连接好后，点击 Options/Debug 选项卡里，Use ST-Link Debugger 右侧的 Settings 里看到设备 ID，则说明一切正常，如下图所示：



3.3 下载运行

点击下载按钮，将程序下载到开发板上运行。



下载完成后，默认程序会自动运行。若程序没有自动运行，可以按一下开发板上的 reset 按钮。

3.4 运行结果

若 LCD 屏幕上的显示内容如下图所示，则说明整个工程是正常运行的。

屏幕上会有显示 Tengine score，返回结果有两个数值，分别表示语音为“xiaozhi”的置信度（第一个数值表示不是“xiaozhi”的概率，第二个数值表示是“xiaozhi”的概率），该置信度数值范围为 0-127。可以看到，对于三个文件，第一个文件 0.WAV 能正确识别出是“xiaozhi”，第二第三个文件能百分百识别出不是“xiaozhi”，与我们实际的音频内容相符。

4 自定义工程

为方便用户在自己的工程里运行 tengine-lite 源码，本章节将详细介绍如何在自定义工程里添加 tengine-lite 源码，并对编译部分进行必要的配置。同时对用户自定义的模型进行转换。我们以 ST 公司提供的 SDK 源码包为基础，在该工程的基础上添加 tengine-lite 源码。

4.1 模型转换

（注：当前内测版本，暂不支持 CAFFE、MXNET 等网络模型，且由于涉及到算子开发和兼容支持问题，当前的内测 tensorflow 模型，只支持 CONV2D、MAXPOOL、RELU、FC、SOFTMAX 等算子，敬请理解）。

当前阶段，Tengine-lite 无法直接加载 tensorflow 生成的模型文件，我们需要将.pb 文件进行转换，将我们关心的数据提取出来并填到对应的结构体中。基于此，我们提供了模型的自动转换脚本 pb2c.py，该脚本是 python 编写，故执行之前请确认系统中已经安装好 python。脚本存放路径在 Demo 工程根目录下的 Script 文件夹里。

该脚本将 tensorflow 生成的模型文件（.PB 格式）转换为嵌入式程序可以直接加载识别的.c 和.h 文件。

具体的使用说明如下：

```
python pb2c.py pb_file_name   graph_name
```

参数说明如下

pb_file_name ： 第一个参数为已经通过 Tensorflow 训练好得到的模型文件，必须为.PB 格式

graph_name ： 第二个参数为转换完成后的 Graph 取的名字，该名字可以用户自行定义，若没有输入该参数，Graph 将设置为默认名字 tengine-lite

下面是一条实际的转换命令，在命令行下执行：

```
python pb2c.py xz_net_only0528.pb test_graph
```

待转换完成后，当前目录下会自动生成两个文件：

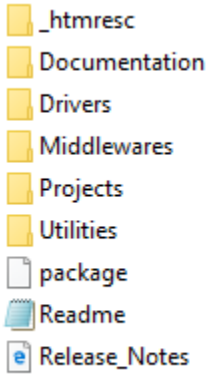
tiny_ param_generated.h
tiny_graph_generated.c

将生成的文件 copy 到 tengine-lite 工程的 tests/bin/tiny 目录下，替换掉 demo 里自带的两个文件。

4.2 选择 demo 工程

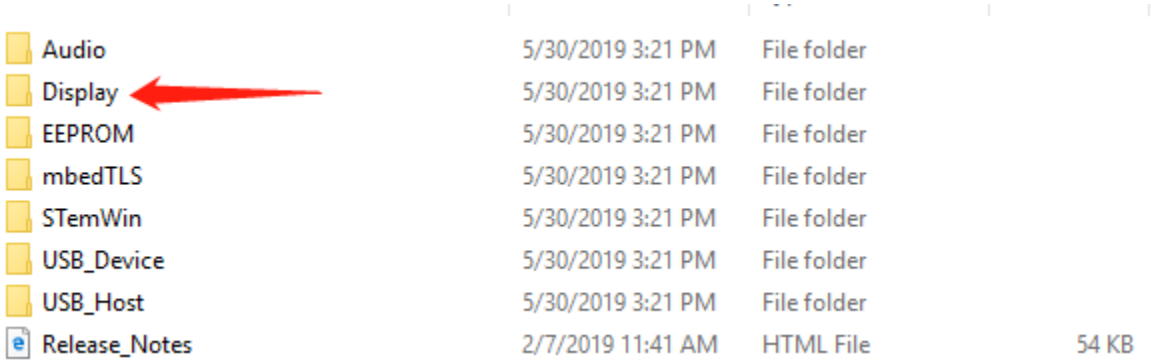
通过该链接下载 ST 提供的 SDK 源码包 en.STM32Cube_FW_F7_V1.15.0。

下载完成后解压，目录结构如下：



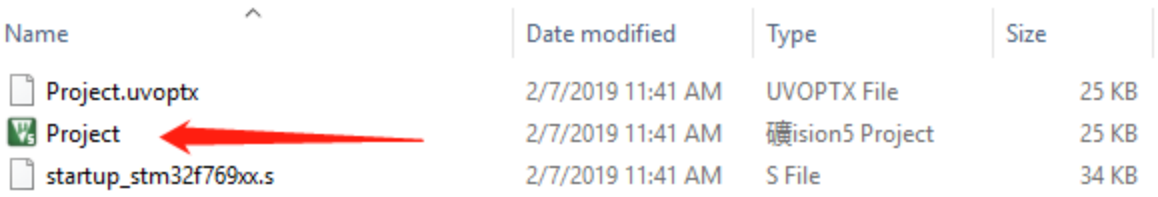
进入到 Projects\STM32F769I-Discovery\Applications 目录，

我们选择以 Display 工程为模板来添加我们的 Tengine-Lite 源码。

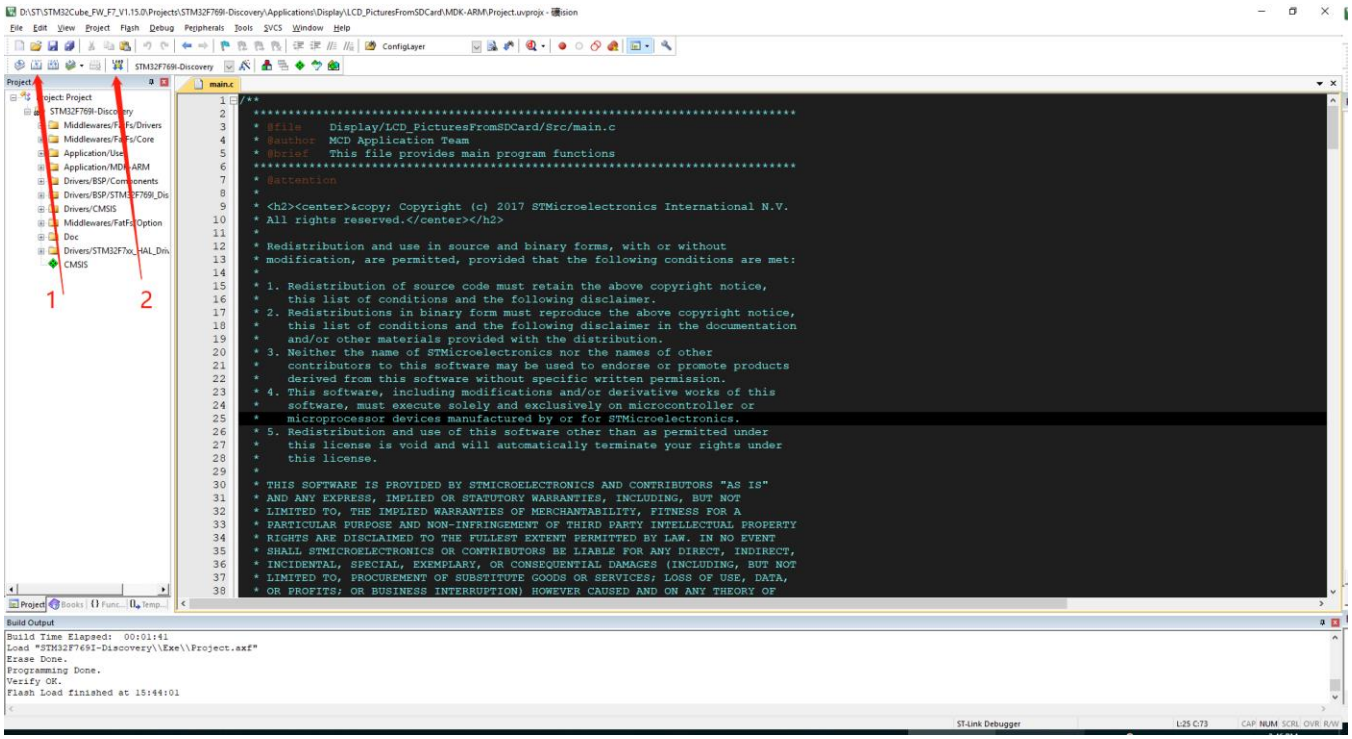


进入到 Display 目录下的工程文件，路径为：Display\LCD_PicturesFromSDCard\MDK-ARM

可以看到文件条目：



双击 Project 图标（红色箭头指向），打开 Project。



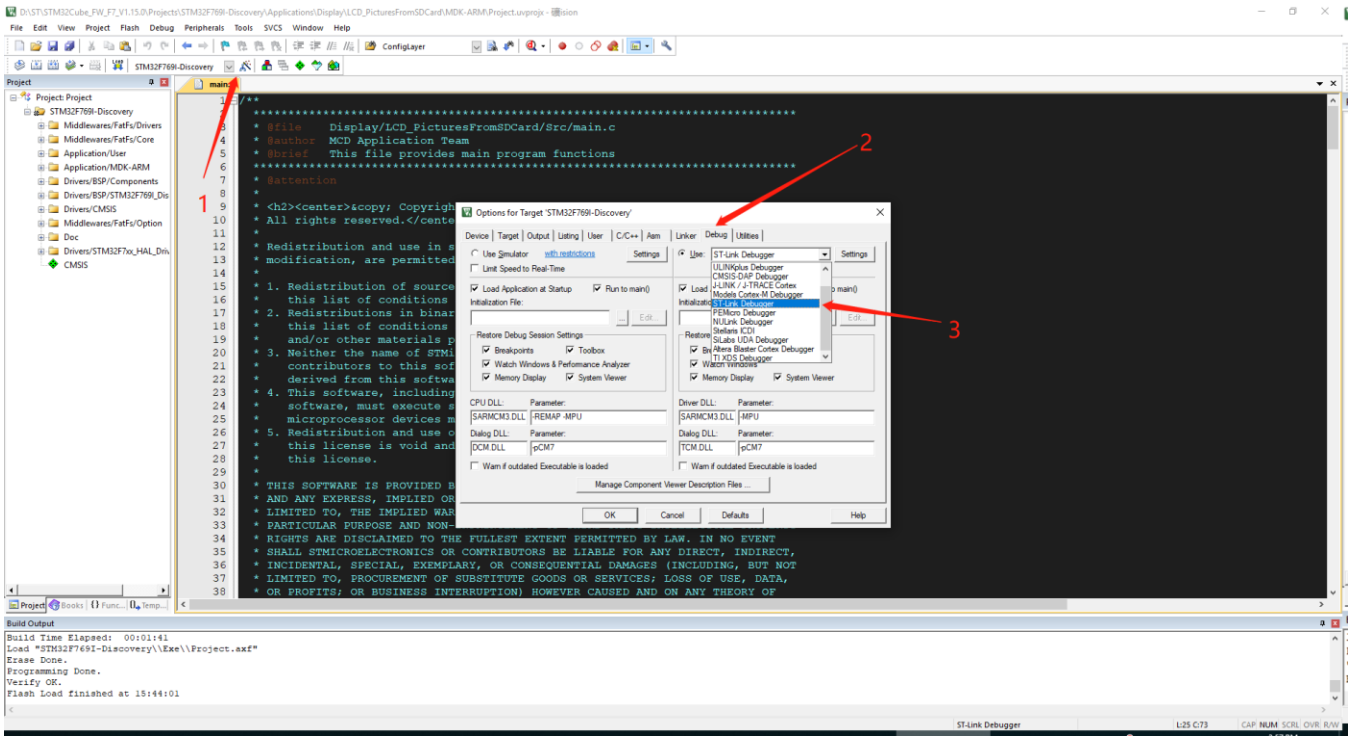
选择红色箭头 1 指向的按钮，开始编译整个工程

待编译完成后，选择箭头 2 指向的按钮，下载编译好的 image 文件，下载完成后，按下 Reset 按钮，可以看到屏幕上显示

“No Bitmap files…”

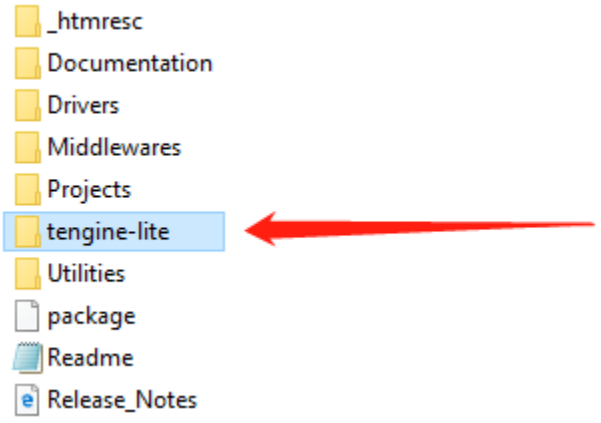
这是因为我们的 SD 卡里没有存放 bitmap 文件，我们先忽略这个显示报警，可以执行到到这一步的话说明我们的准备工程都是 OK 的。

若在下载过程中出现下载相关的报错，请检查 Options->Debug 选项卡里的下载选项是否正确勾选为 ST-Link Debugger



4.3 下载 tengine-lite 源码

推荐使用 Git 下载 Tengine-Lite 源码。本工程里我们将 tengine-lite 源码存放在项目根目录下



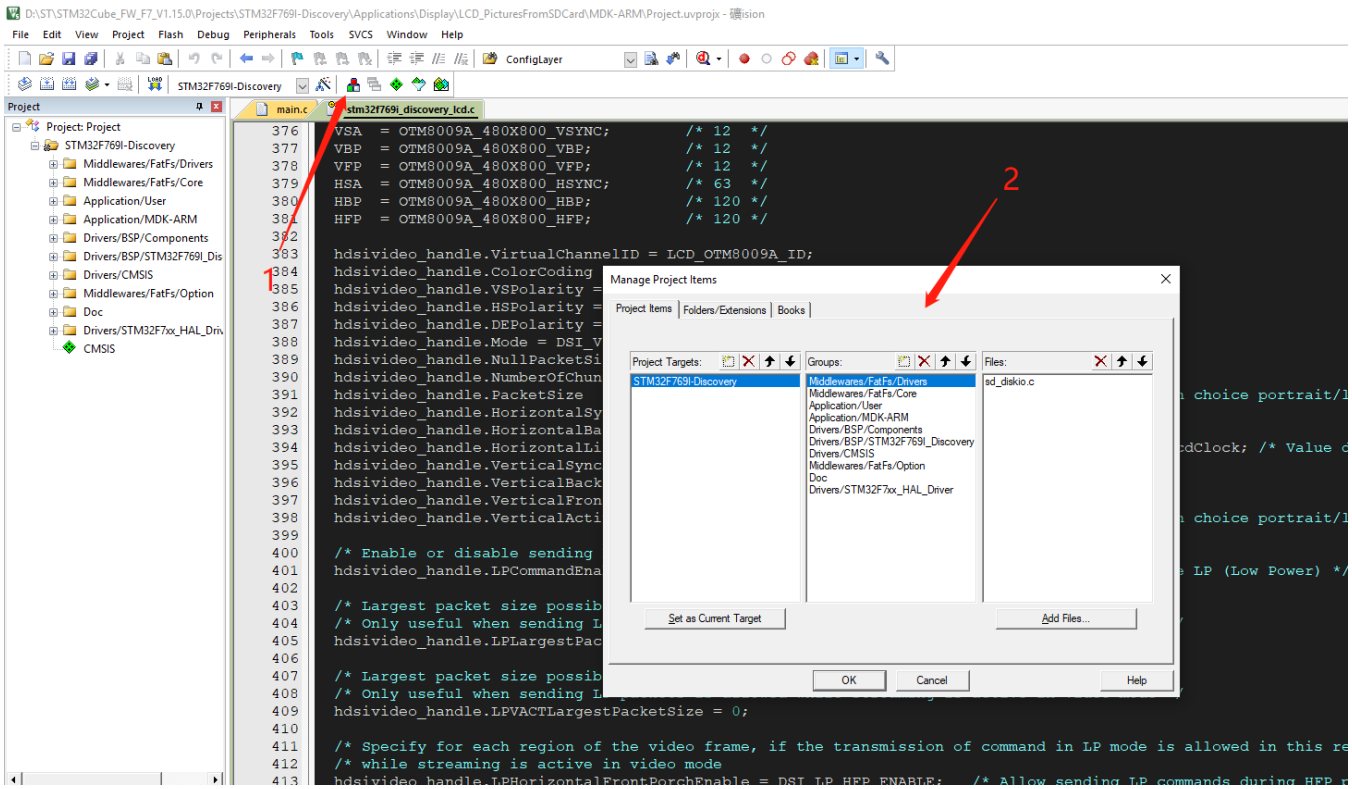
Tengine-lite 的下载路径为：

```
git clone ssh://git@192.168.234.8:8022/internal/tengine-lite.git
```

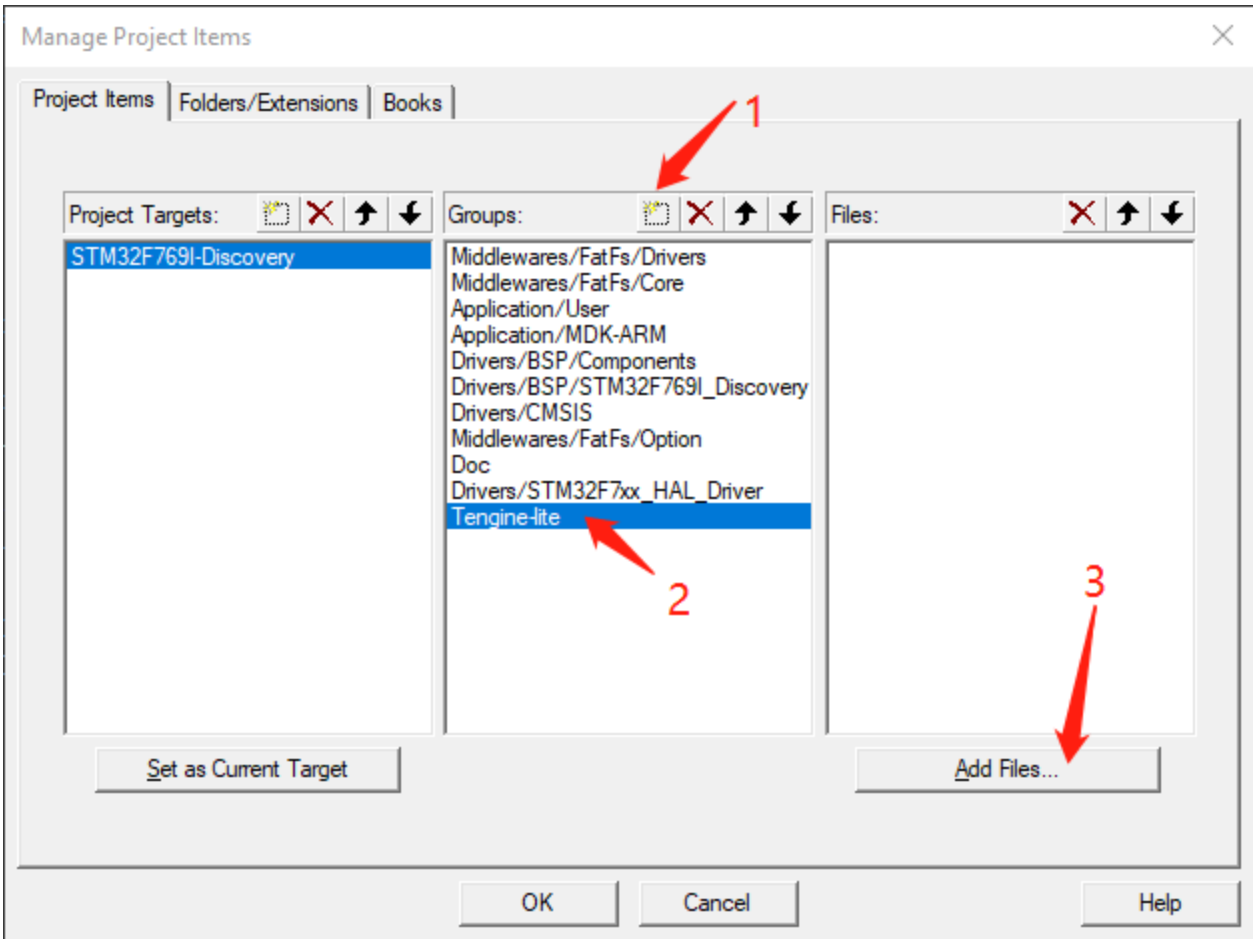
4.4 Tengine-Lite 源码集成项目工程

4.4.1源码添加

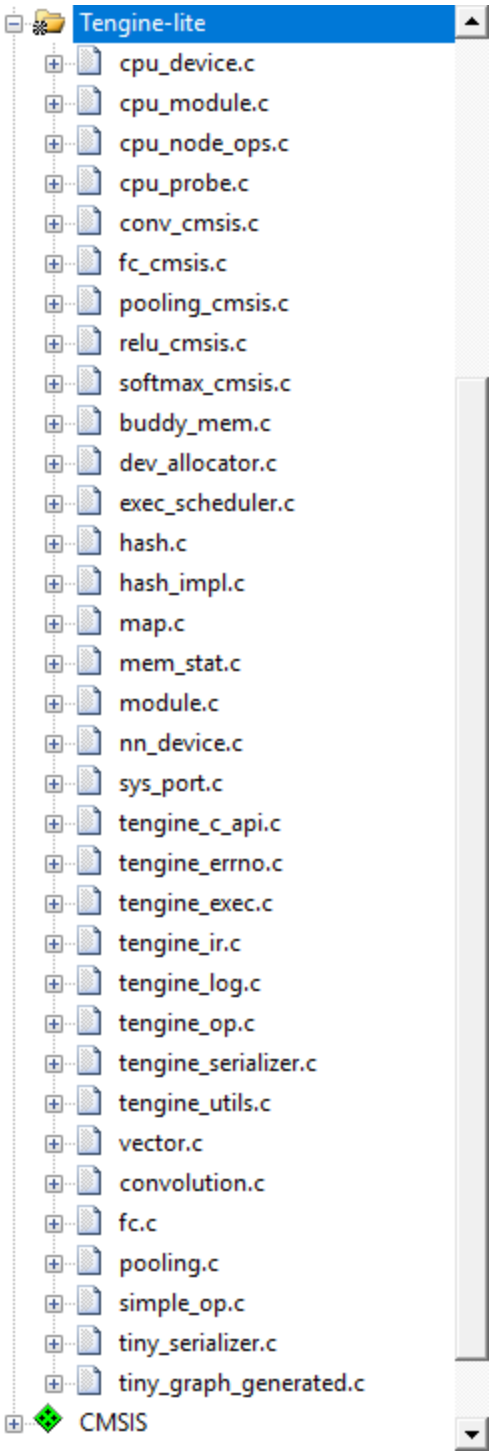
打开工程目录管理卡



新建一个文件目录，取名为 Tengine-lite,并将 tengine 的相关文件添加到该目录下，如下图所示:

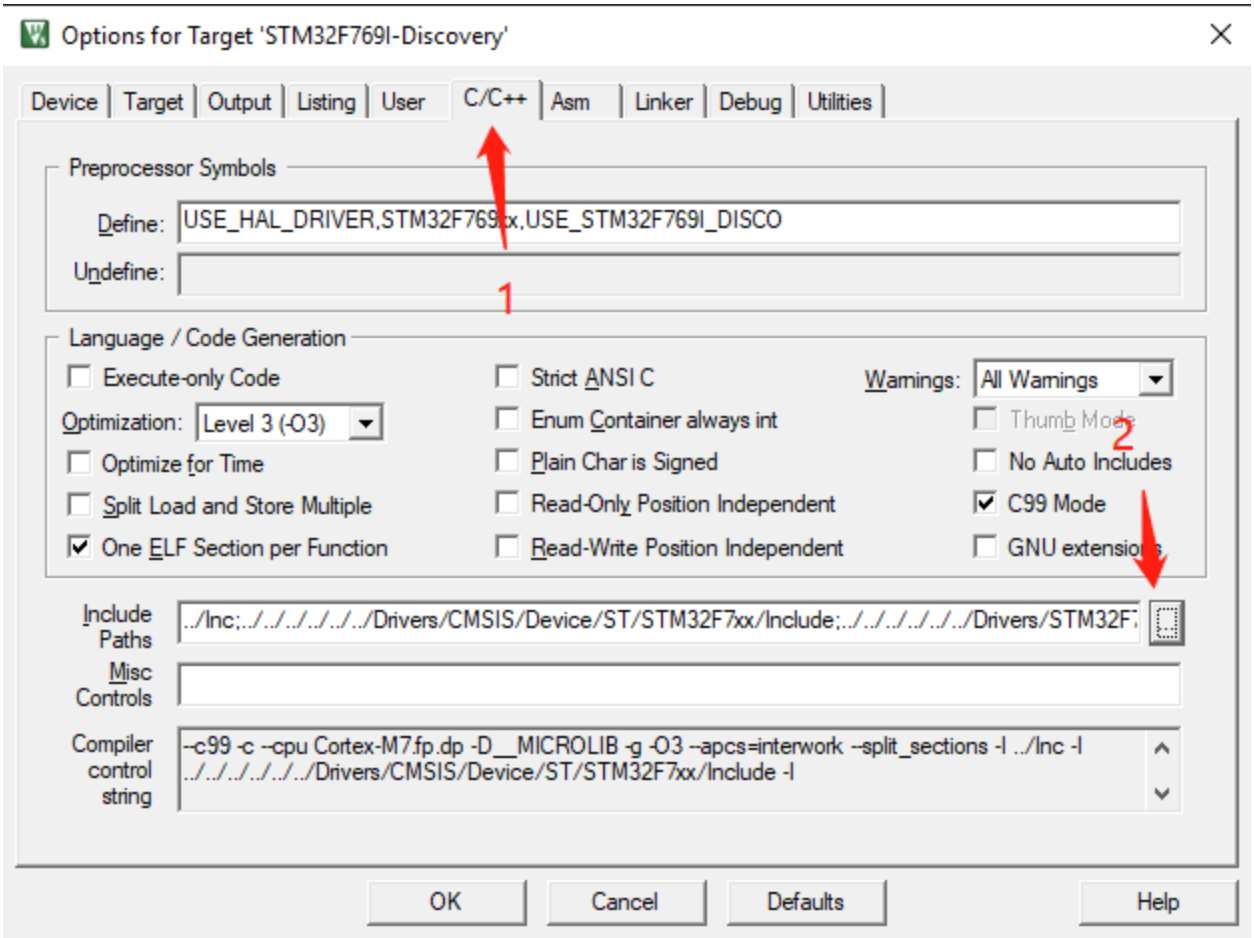


需要添加的文件集中在 src 目录下和 tests/bin/tiny 目录下，用户可以根据给出的截图信息来添加对应的文件。添加完成后，可以看到左侧的目录栏的出现了 Tengine-lite，且相关源码文件已经添加到该目录下：

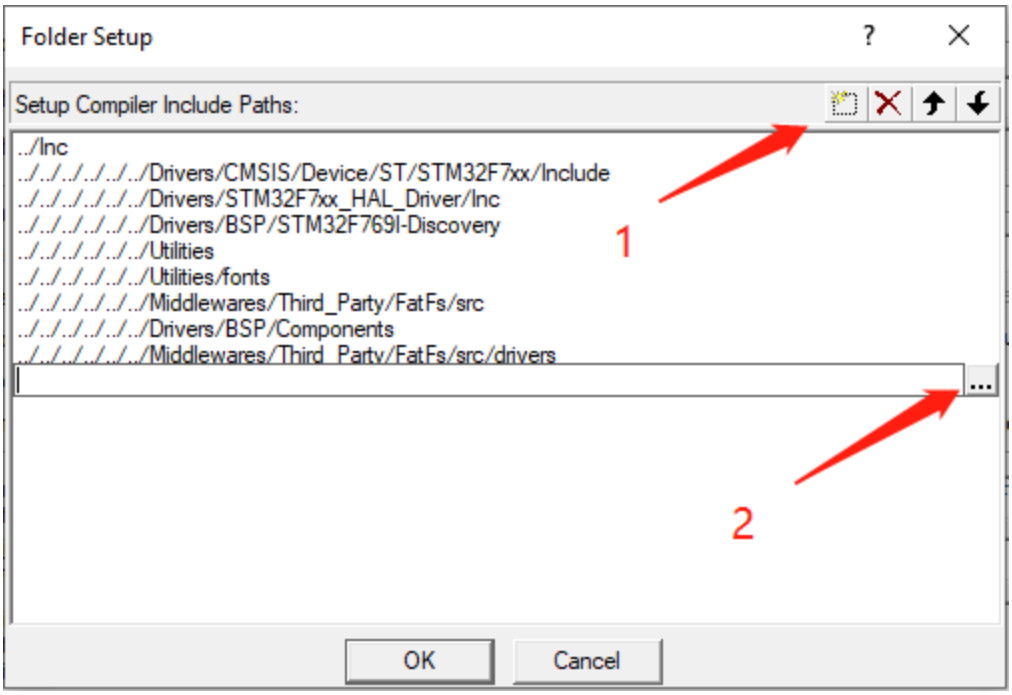


4.4.2头文件包含

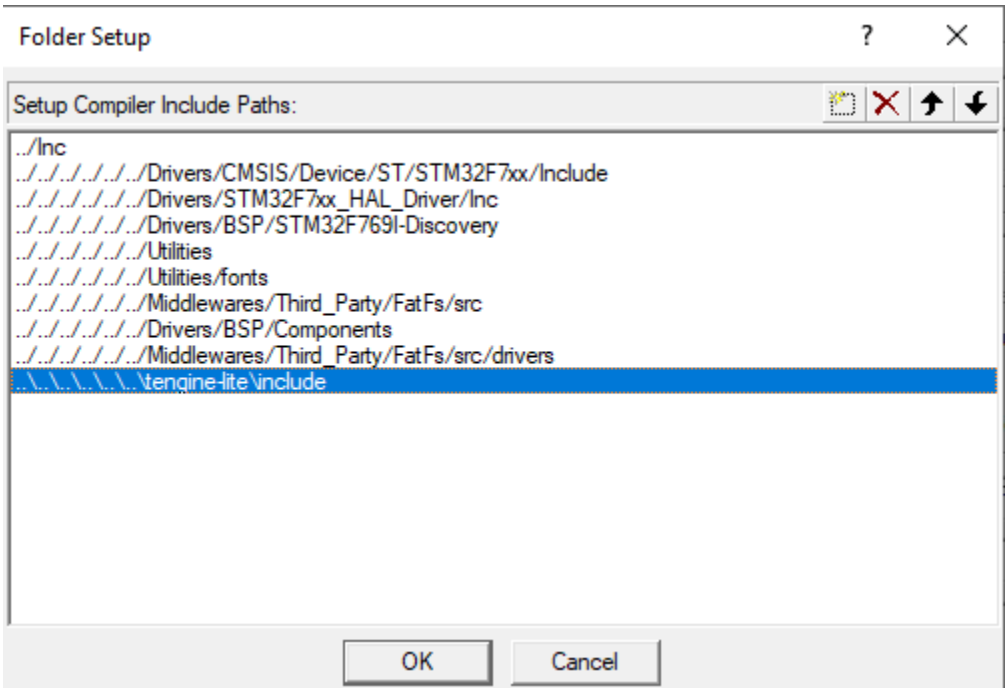
.c 文件添加后，还需要将头文件包含进来。选择 Options->C/C++，选择 Include Paths



按照如下所示把 Tengine-lite 的 include 目录包含进去



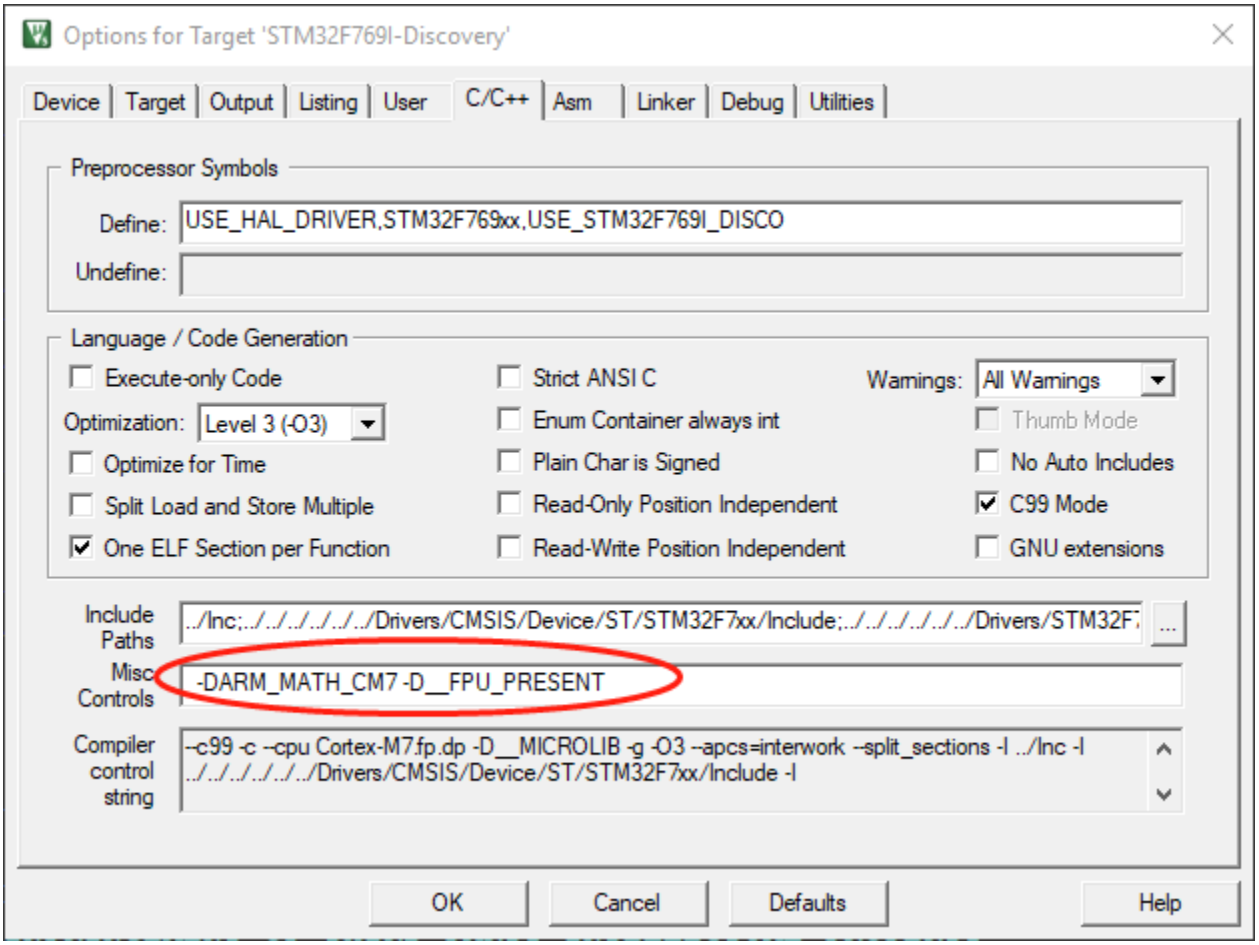
添加完成后的内容如图所示：



因为我们是针对 CM7 内核的工程，需要用到 CMSIS 提供的计算库，所以需要在 Misc Controls 里填入：

```
-DARM_MATH_CM7 -D__FPU_PRESENT
```

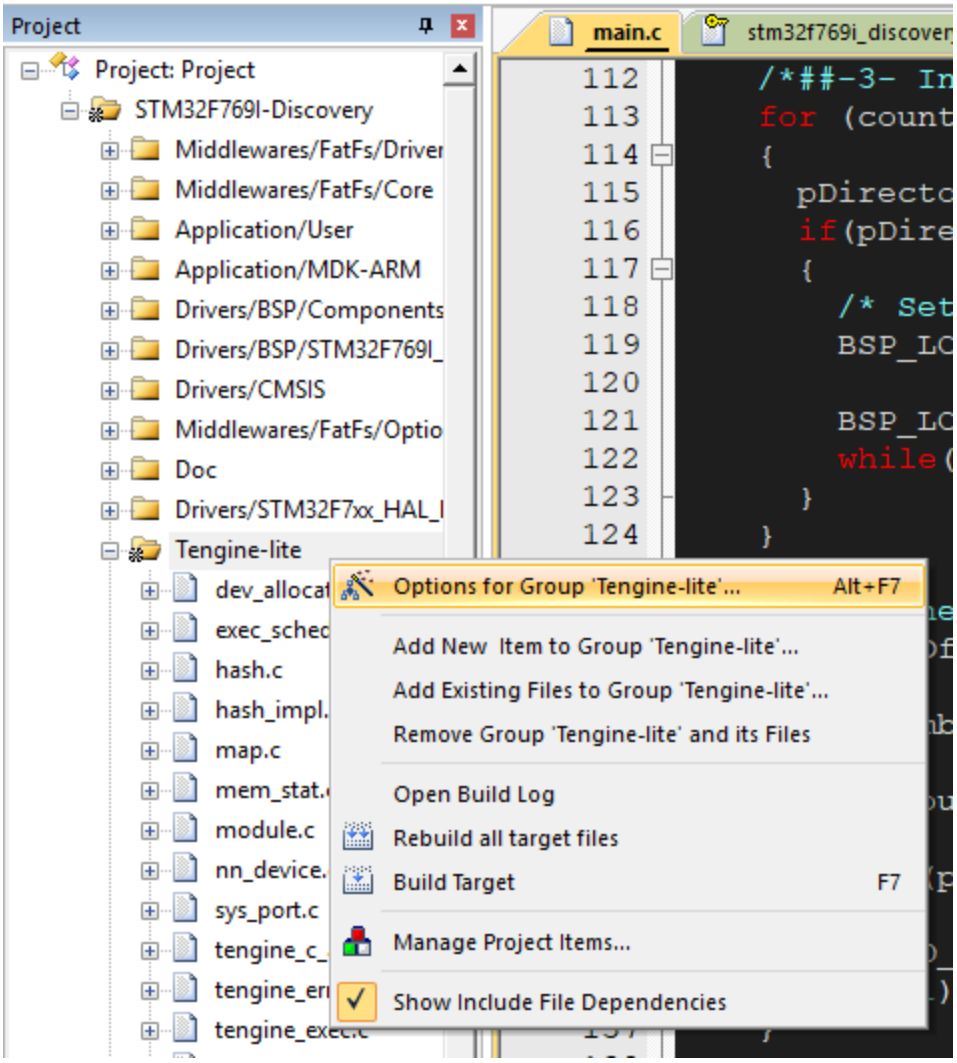
如图所示：



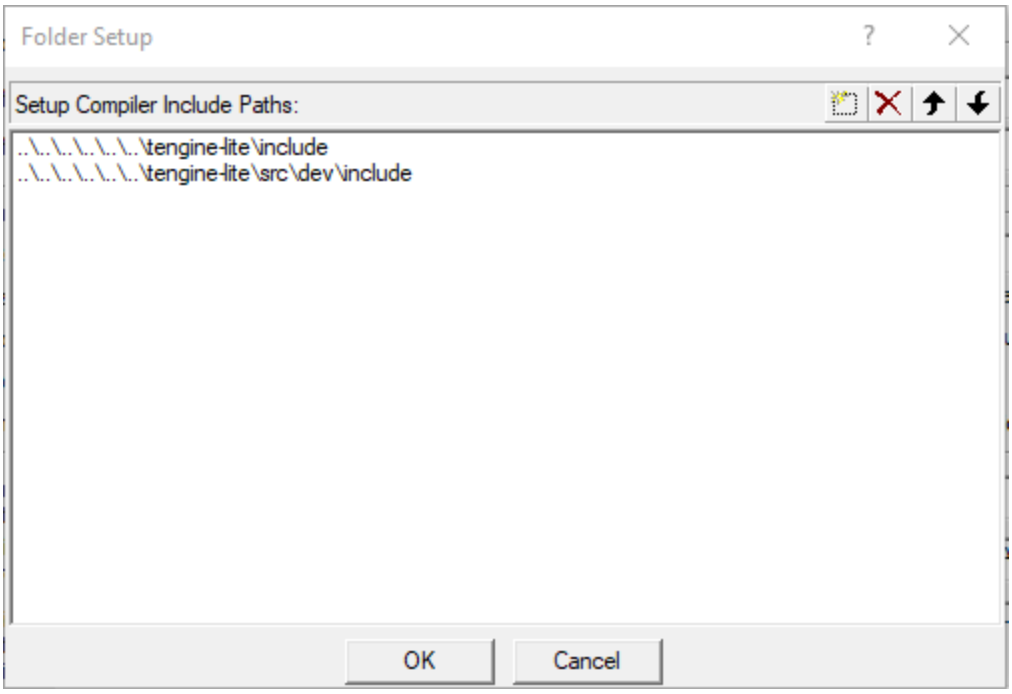
4.4.3Tengine-lite 编译设置

针对 tengine-lite 的编译，需要进行一些特殊的设置（比如 GNU 扩展等），请按照如下步骤来配置

右键选择 Tengine-lite 目录，选择第一个选项 Options for Group ‘Tengine-lite’ ：



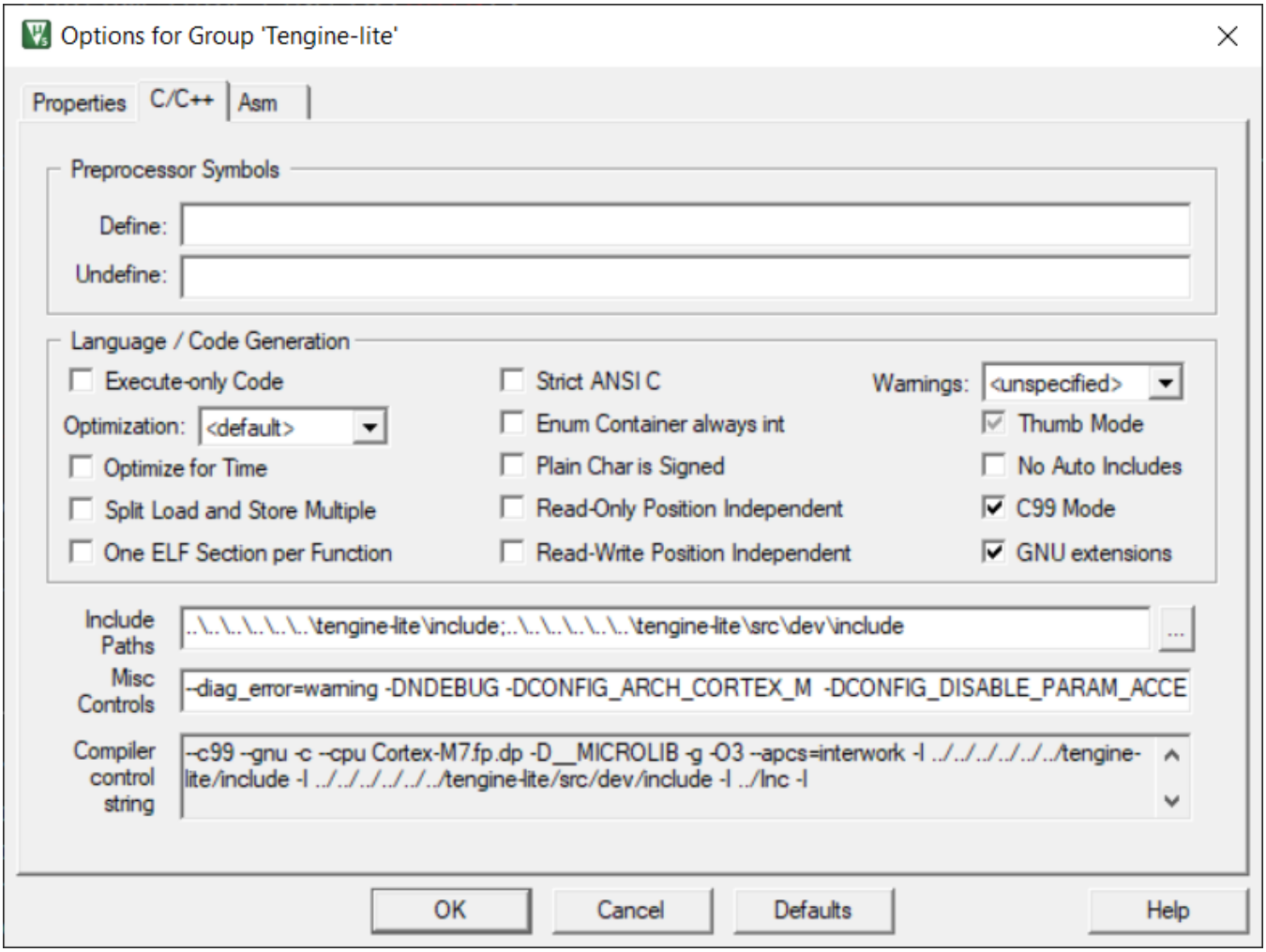
Inlcude Paths 里将 Tengine-lite 编译所需要包含的头文件包含进去，添加完成后内容如图所示:



Misc Control 的框里填入：

```
--diag_error=warning -DNDEBUG -DCONFIG_ARCH_CORTEX_M  -
DCONFIG_DISABLE_PARAM_ACCESS
```

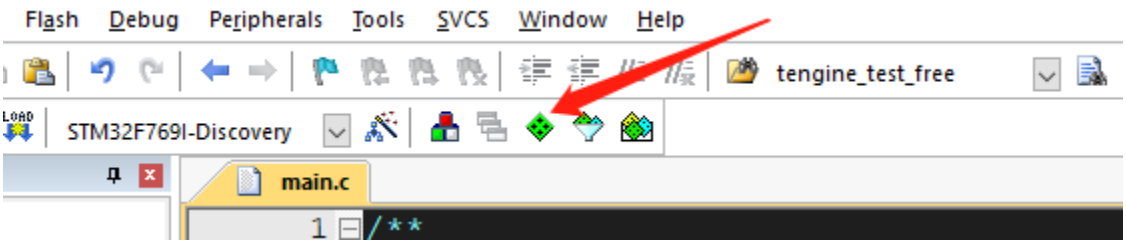
将 GNU extensions 勾选上，One ELF Section Per Fucntion 勾选掉，设置完成后，UI 显示如下所示：



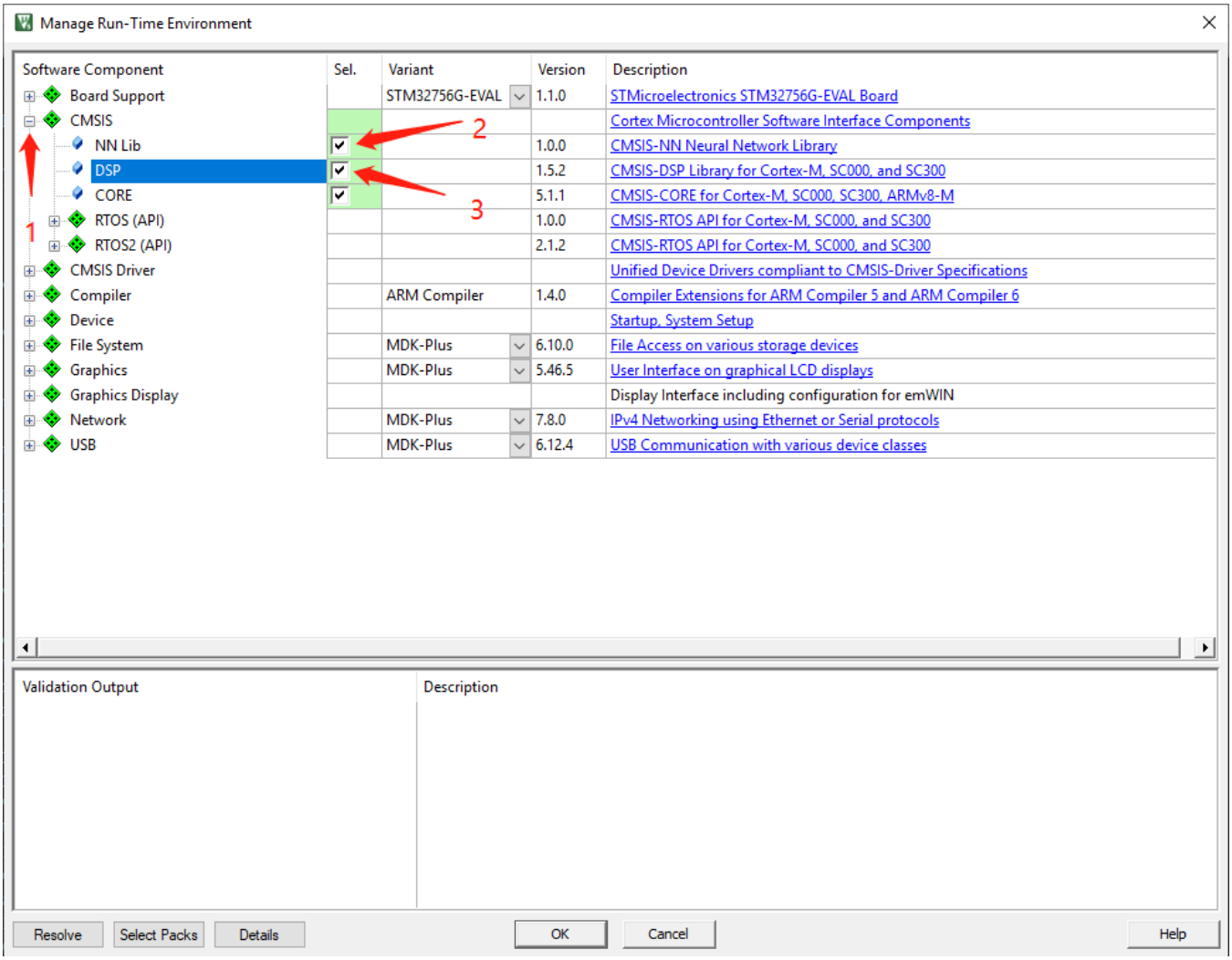
4.4.4CMSIS 的配置

项目工程需要用到 CMSIS 库里的神经网络库（NN lib）和 DSP 库支持，所以还需要勾选该库的配置，步骤如下：

点击 manage Run-Time Enviroment 图标

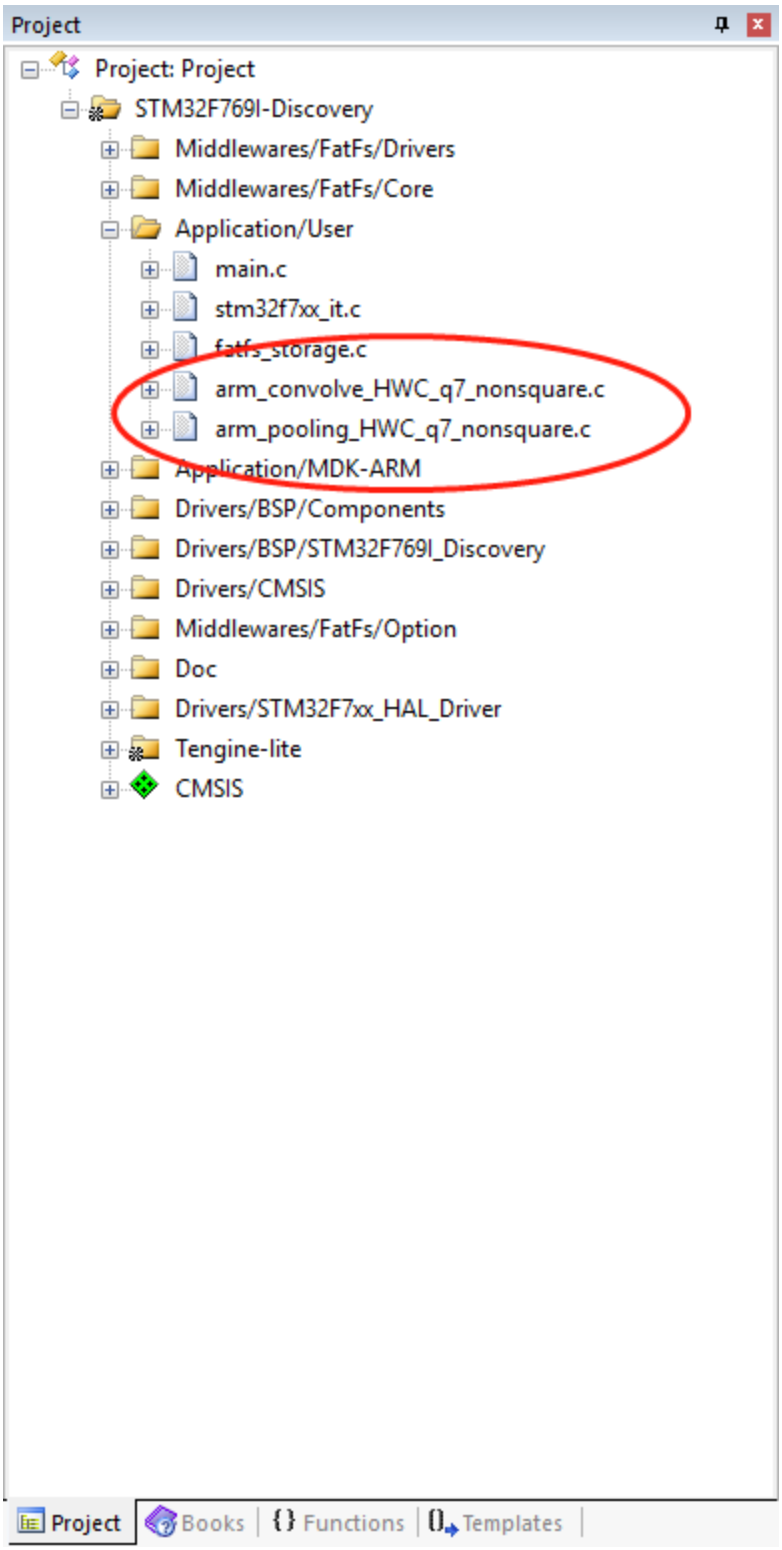


在 manage Run-Time Enviroment 图形框里，将 CMSIS 的 NN lib 和 DSP 勾选上：



以上配置完成后，考虑到当前的实现，我们对 conv 和 maxpool 两个 operator 进行了微改动（当前代码实现是基于 CMSIS 自带的 arm_convolve_HWC_q7_fast_nonsquare（）和 arm_maxpool_q7_HWC（）两个函数调整而来），所以还需要添加 conv 和 maxpool 实现的函数文件（可以先试编译一下，会有 conv_cmsis 和 pooling_cmsis 两个文件的报错）

首先将 arm_convolve_HWC_q7_nonsquare.c 和 arm_pooling_HWC_q7_nonsquare.c 复制到 Projects/STM32F769I-Discovery/Applications/Display/LCD_PicturesFromSDCard/Src 目录下（或者是其他文件夹里，此处用户可以自行定义）；接下来将 arm_convolve_HWC_q7_nonsquare.c 和 arm_pooling_HWC_q7_nonsquare.c 添加到工程的 Application/User 文件下，如图所示：



添加完成后，编译整个工程，正常情况无报错信息。至此，Tengine-lite 部分导入完成。

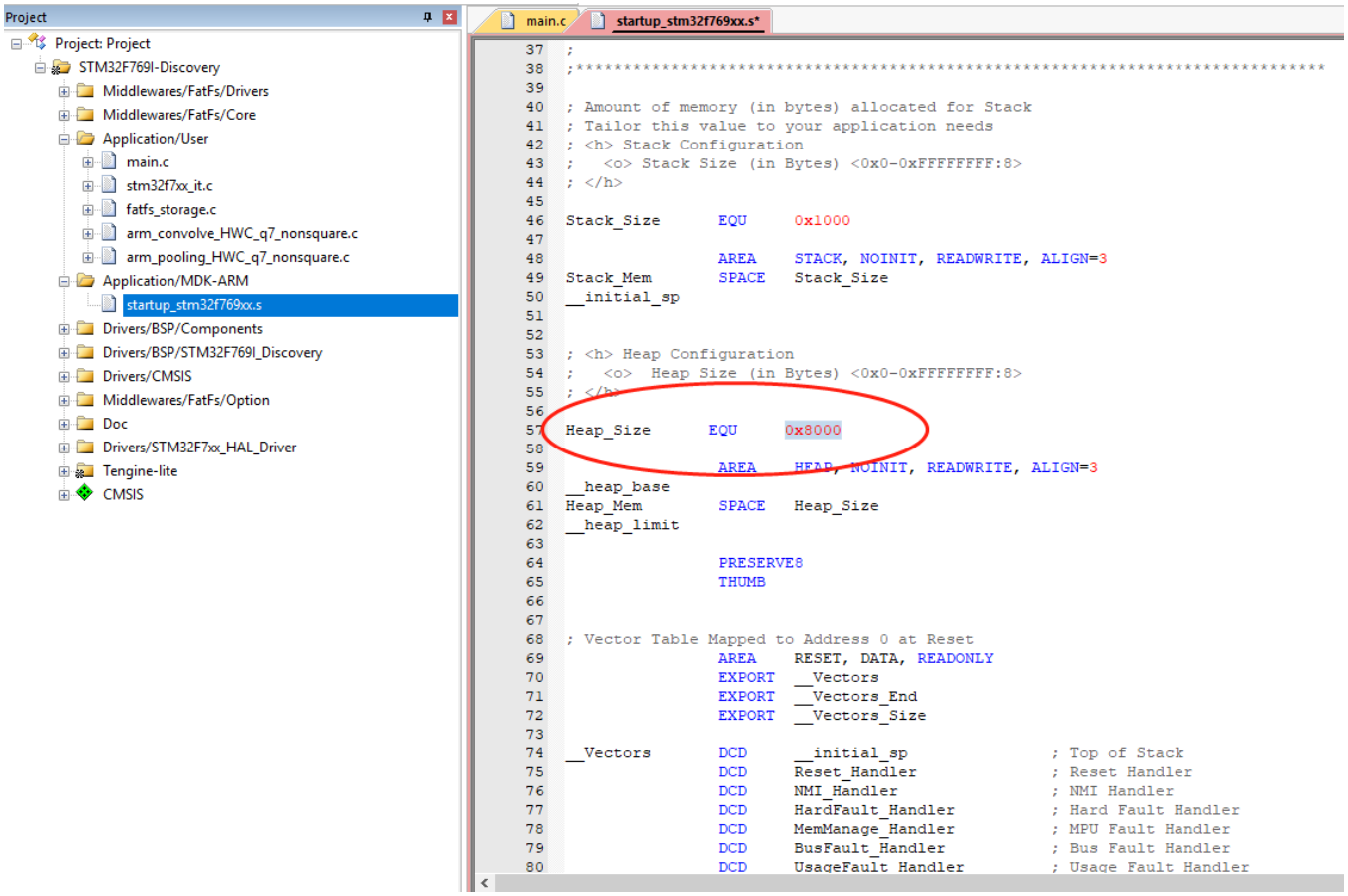
4.5 应用程序

本章节将介绍如何在项目工程里运行的自定义程序，用户可以根据需求，添加对应的文件，下面以语音处理流程简要讲述下验证流程。

4.5.1堆栈的设置

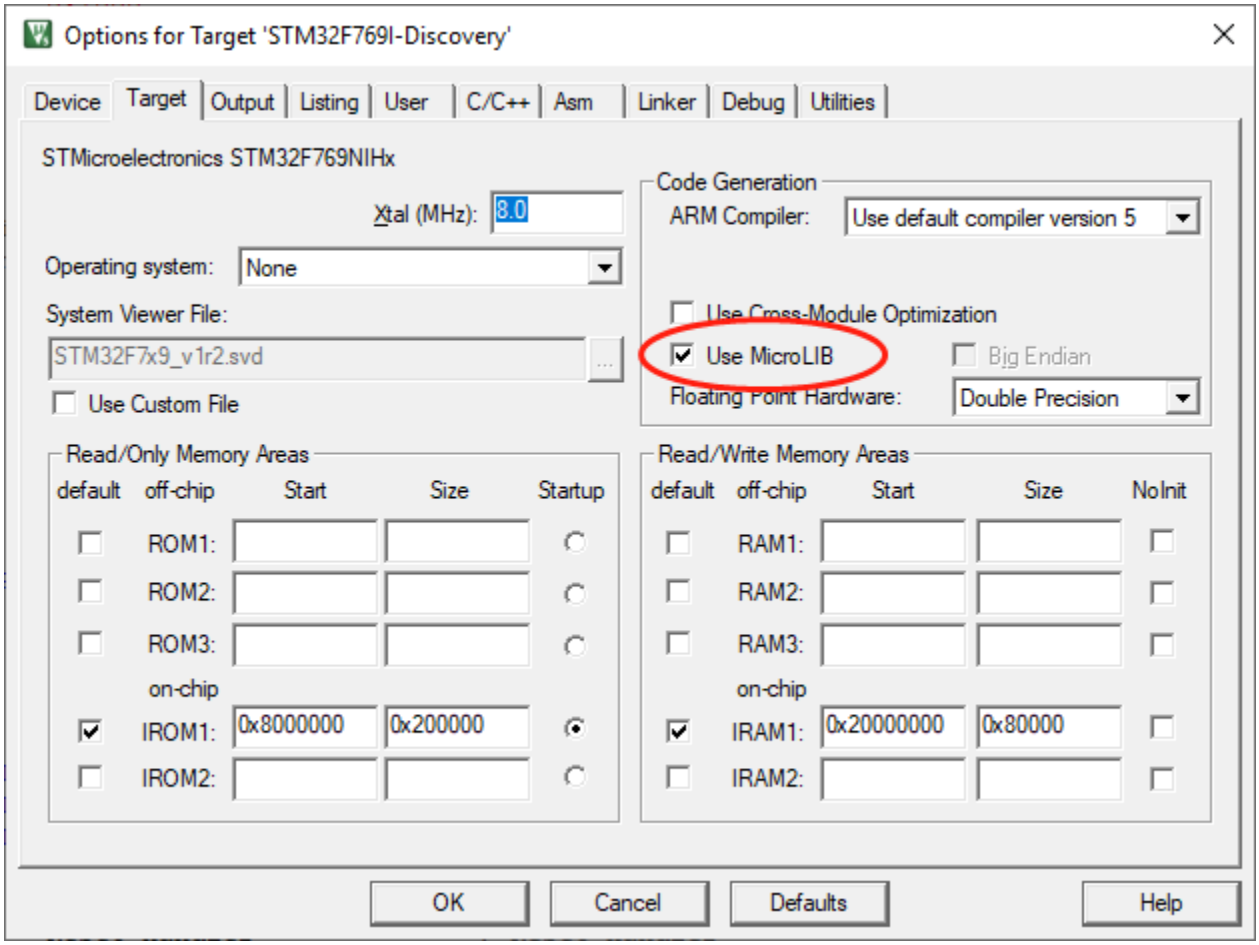
Tengine-lite 运行需要占用一定得 ram 空间（当前 demo 网络模型大约在 15K 左右），系统默认的 Heap Size = 0x600 无法满足要求，所以我们需要先更改一下 Heap Size。

此处我们将 Heap Size 更改为 0x8000，用户也可以根据自己的网络模型需要适当加大一点。Heap Size 的更改在目录结构 Application/MDK-ARM 的 startup-stm32f769xx.s 里。



4.5.2printf 重定向

为方便 debug，我们将 printf 函数重定向到 LCD 屏幕输出（或者是串口输出）。用户可以直接将我们提供的 print.c 先复制到 Projects/STM32F769I-Discovery/Applications/Display/LCD_PicturesFromSDCard/Src 文件里（或者是其他文件夹里，此处用户可以自行定义）再将文件添加到 Application/User 目录结构下。需要注意的是，重定向 printf 功能函数，需要将 options 里 Target 选项卡的 Use MicroLIB 勾选上。

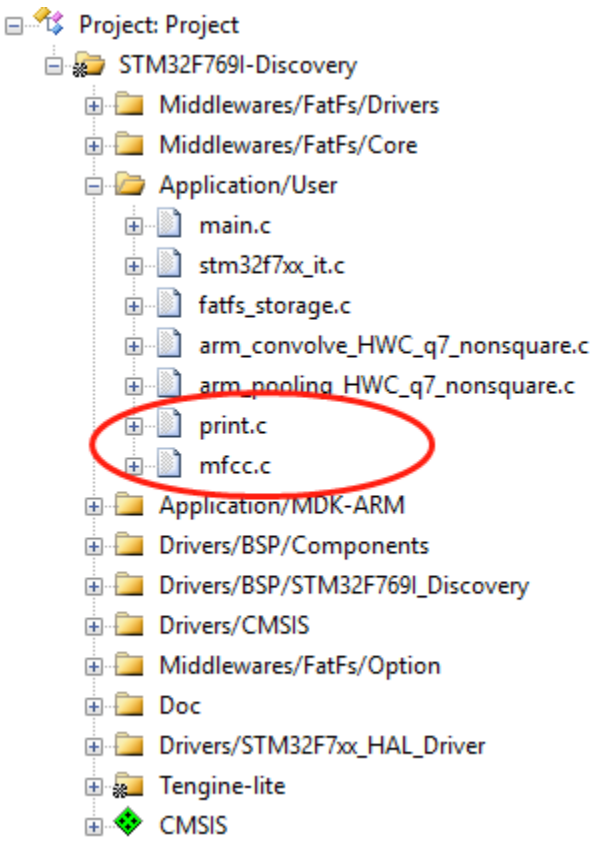


4.5.3语音特征参数文件预处理

我们提供了一个语音预处理的文件 mfcc.c，将语音文件进行预处理，用户可以将该文件复制到 Projects/STM32F769I-Discovery/Applications/Display/LCD_PicturesFromSDCard/Src

里，将 mfcc.h 文件复制到 Projects\STM32F769I-Discovery\Applications\Display\LCD_Pictures FromSDCard\Inc 目录；再将 mfcc.c 文件添加到 Application/User 目录结构下。

添加完成后，目录结构文件如下：



4.5.4主函数编写

出于简单考虑，我们建议第一次测试时候将 demo 工程里的 main.c 文件内容复制到本工程的 main.c 文件。在 demo 工程里的注释详细介绍了每一步的 tengine API 的调用。

以上步骤全部完成后，可以执行编译，正常情况下无报错。下载到开发板上，LCD 屏幕显示如下：


run-time library version: 0.1-dev
tiny graph name: test, node num: 12
final tensor_mem_list number: 0
shared memory: 200049c8 size=4096
block number: 2 align size: 16
0: 200059d0 (10100) used: 4 free: 4
1: 20008148 (2964) used: 3 free: 3
Tengine test file: 0.wav
Tengine score = 1 126, Speech words = xiaozhi
Tengine test file: 1.wav
Tengine score = 127 0, Speech words = unkown
Tengine test file: 2.wav
Tengine score = 127 0, Speech words = unkown
FREE RESOURCE DONE
ALL TEST DONE
MB1166 Rev.A
MB1166 A-02
217280465
STM32F769I-DISCO
MB1225B
www.st.com/stm32f7-discovery