Application Note

ZHCAG65 – Dec. 2025

AM275 AWE SDK 编译及自定义模块使用方法

Harry Zhao, Joe Shen *Central FAE*

摘要

Sitara™ AM275 系列是 TI 公司为下一代音频系统设计的高性能 MCU 产品，AM275 系列产品采用多核异构架构，除集成 TI 最新 C7000 系列 DSP 提供强大的浮点计算能力外，还集成多核 MCU R5F 运行高实时性任务，使用单芯片即可实现音频处理中所有的高性能音效算法。复杂的音频处理流程需要一套完整的数据流和算法框架，Audio Waver（下文中简称 AWE）是当前 AM275 系列产品最主要应用的音频框架，并提供与此适配的 AWE SDK 进行产品应用开发。

本篇应用手册主要介绍 AM275 AWE SDK 的安装编译方法以及自定义模块的开发和集成方法，可以帮助开发者高效地熟悉 SDK 编译流程，使用内置功能模块进行应用开发，或者根据不同需求定制开发高阶自定义模块。

目录

1. AWE SDK 编译

1.1 系统要求

- Windows 10 64-bit 及以上
- 最少 4GB，推荐 >8GB RAM
- 最少 10GB 硬盘空间

1.2 下载和安装

SDK 下载地址：https://www.ti.com/product/AM2754-Q1#software-development

- 下载 Windows 版本 AM275-AWE-SDK 安装程序

需要注册 TI.com 账号进行申请

- 下载 Windows 版本 AM275-FREERTOS-SDK 安装程序

无需申请，可以直接下载

- 将 SDK 安装在默认 C:\ti 路径

1.3 系统环境配置

- **Sysconfig**
  - 下载地址：https://www.ti.com/tool/download/SYSCONFIG/1.24.0.4150
  - 安装在默认路径：C:\ti
- **C7000-CGT Compiler Toolchain**
  - 下载地址：https://www.ti.com/tool/download/C7000-CGT
  - 安装在默认路径：C:\ti
- **Python3**

注 1* Python 3.x 版本为必须项，如已安装 Python 2.x 版本则需要进行升级

注 2* 以下所有的命令行需要在 Windows cmd.exe (Command Console)运行，不支持其他命令行软件

注 3* 建议不要选择 Python 3 最新子版本，3.9 和 3.10 为验证过的版本

- 下载安装路径：https://www.python.org/downloads/windows/
- 通过以下指令确认 Python 3 已经安装到电脑

  C:\> python --version

  Python 3.9.1
- 如果以上指令失败，则可以将 Python 地址添加到系统环境变量的 path 列表中，Python 默认安装在以下地址，强烈建议在添加环境变量时点击“向上”按钮将其移动到环境变量 path 列表中的第一个

  C:\Users\{your username}\AppData\Local\Programs\Python\Python39
- 关闭并重新打开 Windows 命令行
- 检查 pip 是否已经安装，默认安装 Python 时会自动安装

  C:\> python -m pip --version

  pip 21.0.1 from C:\Users\{your username}\AppData\Local\Programs\Python\Python39\lib\site-packages\pip (python 3.9)

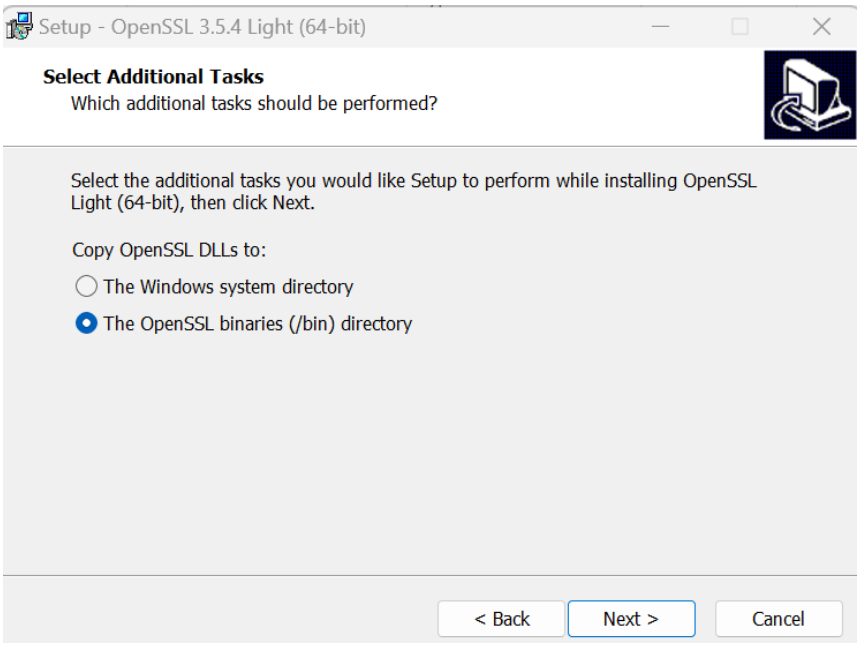
- 通过 pip 安装额外插件，如果网络环境存在防火墙，需要-proxy 后填写服务器地址和端口号，如果没有防火墙则不需要填写内容

  C:\> python -m pip install pyserial xmodem tqdm pyelftools construct --proxy={your proxy server web-link and port}

- **OpenSSL**

注 1* 在对启动文件签名时，需要用到 OpenSSL，由于 SDK 默认会生成签名 Dummy Key 的文件，如无需进行签名，仍建议安装以避免 SDK 编译报错

- 下载地址：https://slproweb.com/products/Win32OpenSSL.html
  需要下载 v1.1.1 或更高版本，可以下载安装 light 版本以占用更小的系统空间
- 安装到默认安装路径：C:\Program Files\OpenSSL-Win64\
- 如下图安装选项时选择 install binaries to /bin folder



- 在系统环境变量 path 列表中添加 OpenSSL 路径：

  C:/Program Files/OpenSSL-Win64/bin

- 在命令行中输入以下命令测试 OpenSSL 是否正常安装，示例输出如下：

  C:\> openssl version

  OpenSSL 1.1.1k 25 Mar 2021

- **Code Composer Studio (CCS)**
  - 参考以下链接下载和安装 CCS：https://software-dl.ti.com/mcu-plus-sdk/esd/AM275X/11_01_00_16/exports/docs/api_guide_am275x/CCS_SETUP_PAGE.html
  - 在系统环境变量 path 列表中添加以下 CCS 路径：C:\ti\ccs2020\ccs\utils\bin
- **TI CLANG Compiler Toolchain**
  - 下载链接：https://dr-download.ti.com/software-development/ide-configuration-compiler-or-debugger/MD-ayxs93eZNN/3.2.2.LTS/ti_cgt_armllvm_3.2.2.LTS_windows-x64_installer.exe
  - 安装在默认路径：C:\ti

1.4 SDK 编译

- 编译 Audio App
  - 在 [Audio SDK Installation]/audio_app/am275x 路径打开命令行
  - 运行 gmake 指令
  - 指令列表：

Command	Windows	Linux	Explanation
Clean	gmake clean	make clean	Cleans the build for release profile. It deletes all the files generated for release build
full clean	gmake scrub	make scrub	Cleans the build for release and debug profile. It deletes all the files generated for release and debug build
Release build	gmake	make	Builds for release profile. It generates out file which can be loaded to target using CCS. It also generates appimage which can be loaded to the target using SD-Boot.
Debug build	gmake PROFILE=debug	make PROFILE=debug	Builds for debug profile. It generates out file which can be loaded to target using CCS. It also generates appimage which can be loaded to the target using SD-Boot.

- 编译 mcu_plus_sdk

注 1* 如果使用 AM275-AWE-SDK_11.01.16.06 版本 AWE SDK，在编译前需要将 AM275-FREERTOS-SDKC_11_01_00_16 版本中 freertos_sdk_am275x_11_01_00_16\source\networking\tsn\tsn-stack\目录下所有文件拷贝到 AM275-AWE-SDK_11.01.16.06\mcu_plus_sdk\am275x\source\networking\tsn\tsn-stack\目录下

- 使用以下命令全量编译

  gmake -s -f makefile.am275x all

2. AWE 标准库添加及运行

2.1. 工具下载及安装

- 下载 matlab 的安装文件 https://www.mathworks.com/downloads/

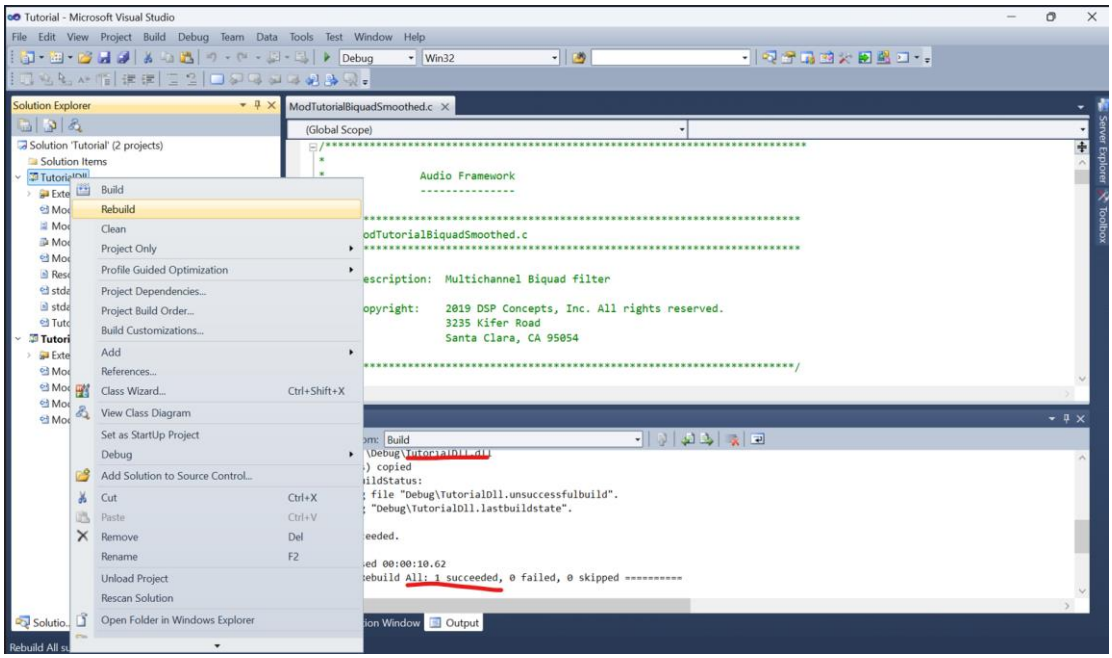
  解压 install.zip 后是一个 iso 文件，打开 iso 文件选择 setup.exe 文件进行安装。
- 安装后请输入完整的 matlab 的 license 的文件。
  修改 windows 的系统的环境变量增加” **MLM_LICENSE_FILE**”的变量并且赋值正确的 matlab 授权的用户名.
- 安装 Microsoft Visual Studio 2010 以上版本
- 安装 AWE_Designer_8.2025.1_Pro.exe

2.2. AWE 标准例程安装

- 下载 AWE 提供的例程(tutorial_module.zip)

https://documentation.dspconcepts.com/awe-designer/latest-version/deploying-custom-modules-on-awecoreos-linux-target

- 用 VS201X 打开 tutorial_modules 里面的 Tutorial.sln 的工程。然后编译产生 TutorialDll.dll。

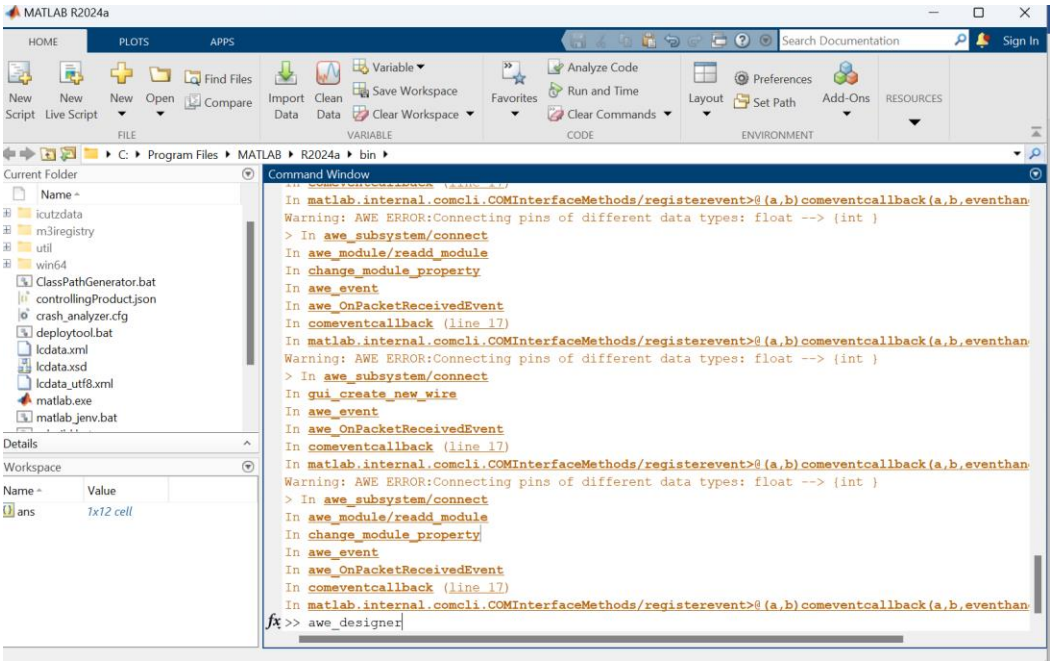


- 拷贝 TutorialDll.dll 到 AWE 的安装目录 AWE Designer 8.2025.1 Pro\Bin\win32-vc142-rel

2.3. AWE 标准例程使用方法

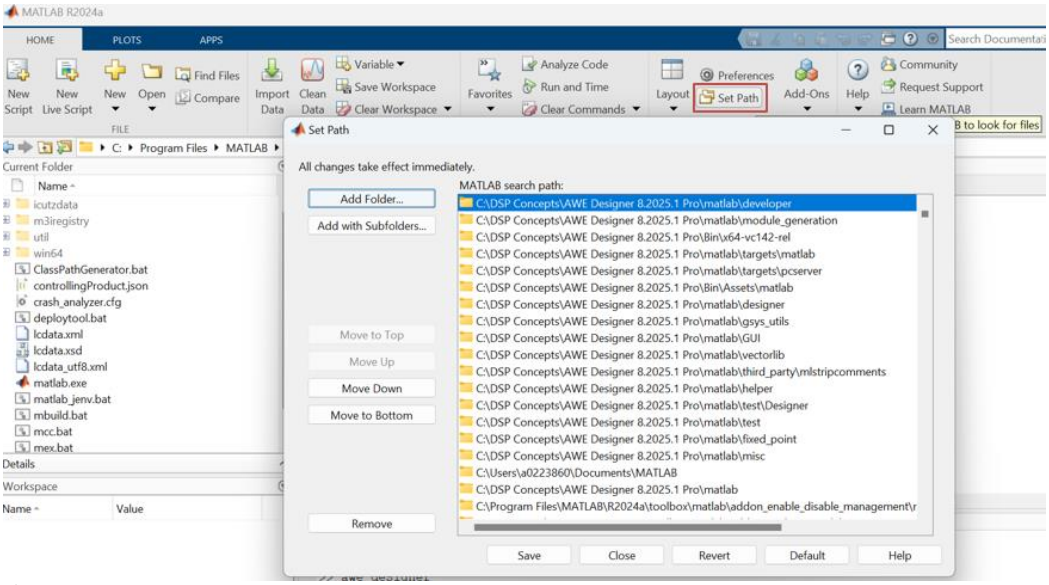
- **PC 端**
  - **步骤 1**

    在 matlab 中运行 awe designer pro



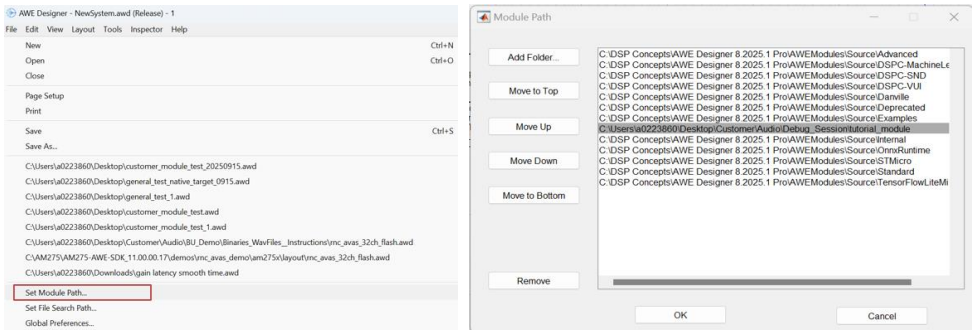
- **步骤 2**

  增加 AWE 的 matlab 的路径



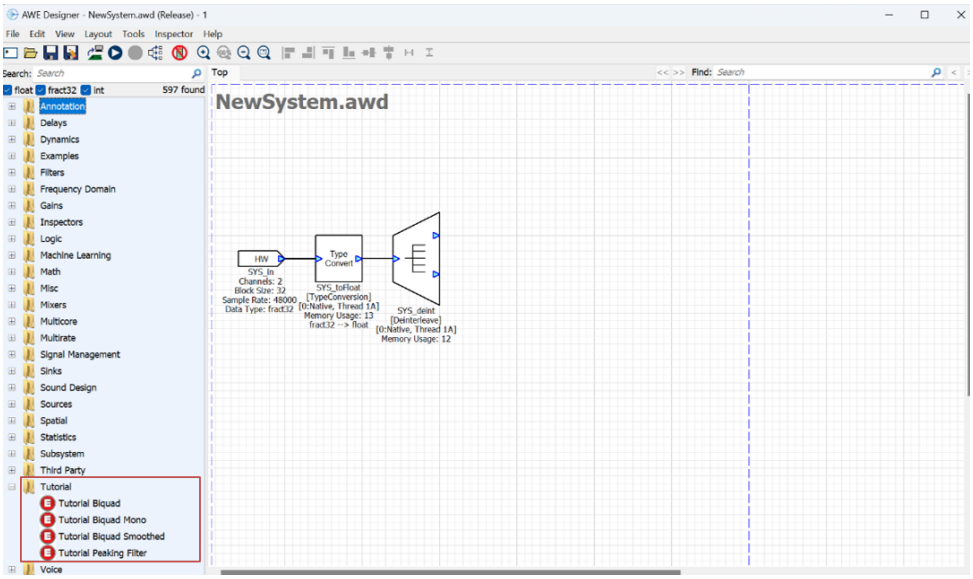
- **步骤 3**

  在 AWE 中增加 tutorial 模块路径

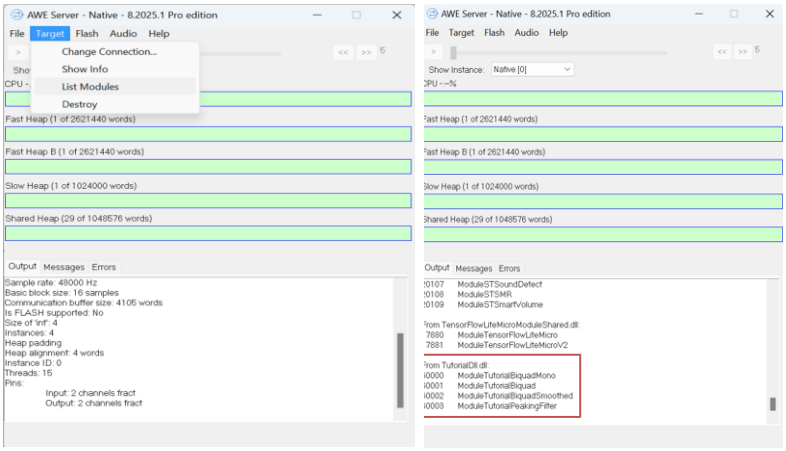


- **步骤 4**

  关闭 AWE 然后重新在 Matlab 中运行，在 AWE 的模块栏中会出现新增加的 Tutorial 的模块



使用 list module 的操作查看是否 Tutorial.dll 被正确加载了

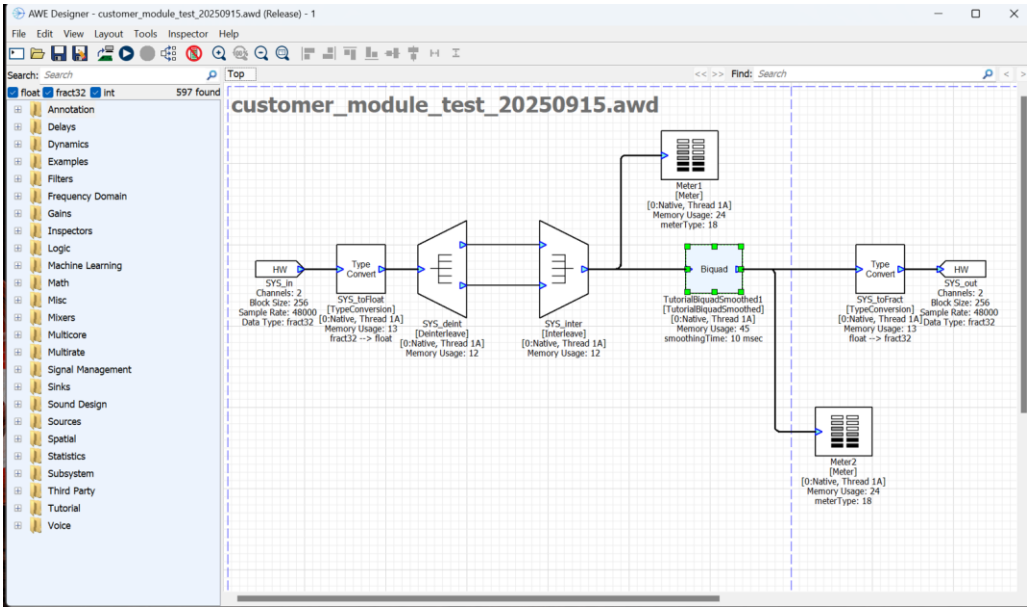


- **步骤 5**

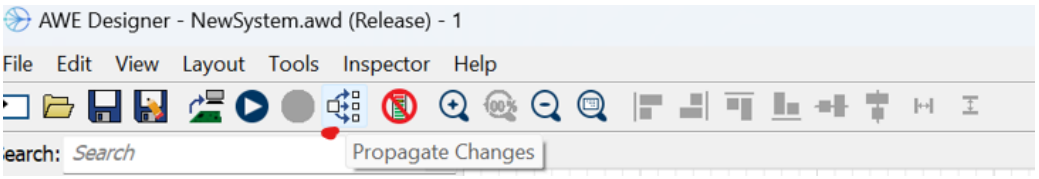
在 PC 上制作一个带有 Tutorial 模块的 AWE 的例程

选择 Target - Change Connection - native

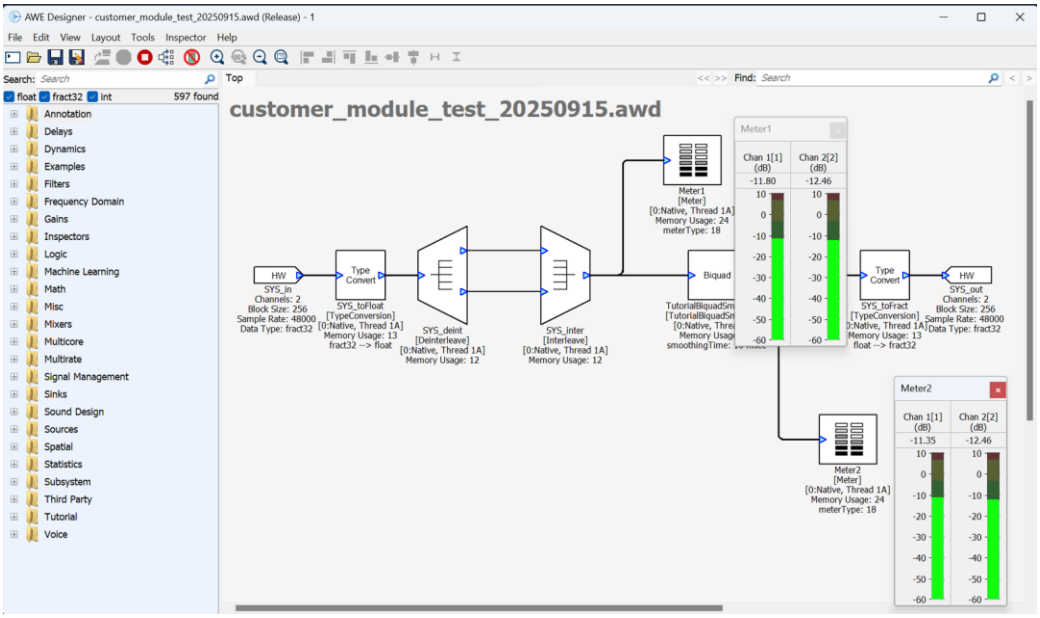
创建一个带有 Tutorial 模块的数据流



点击 Propagate Changes 按钮检查搭建的数据流是否正确。



点击  运行数据流查看新增加的 Tutorial 模块的输入输出是否正常工作。



3. 自定义 AWE 模块

完成以上验证步骤后，可以在 AWE 中根据特殊功能要求自定义模块。

3.1. 基于标准模块添加功能

- **AM275 板端**
  - **步骤 1**
    准备 Ubuntu 22.04 的 Linux PC，下载 AWE 提供的例程(tutorial_module.zip)
    https://documentation.dspconcepts.com/awe-designer/latest-version/deploying-custom-modules-on-awecoreos-linux-target

    下载 TI C7000 Compiler (4.1.0 LTS)并安装
    https://dr-download.ti.com/software-development/ide-configuration-compiler-or-debugger/MD-707zYe3Rik/4.1.0.LTS/ti_cgt_c7000_4.1.0.LTS_linux-x64_installer.bin

    进入 tutorail_module 文件夹创建并且使用如下命令创建 git 初始版本
    git init .
    git add .
    git commit -sm “tutorial init”

    使用如下命令增加附件的修改
    git am “0001-awe-enable-target-BUILD-on-AM275.patch”

    0001-awe-enable-tar
    get-BUILD-on-AM275
  - **步骤 2**
    参照如下命令使用 cmake 编译生成静态库 libdspc_tutorial.a.

```
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module$ export CGT7X_ROOT=/home/cnh20399/ti/ti-cgt-c7000_4.1.0.LTS
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module$ export PATH=${CGT7X_ROOT}/bin:${PATH}
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module$ mkdir -p build
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module$ cd build
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module/build$ cmake ..
=== /home/cnh20399/ti/ti-cgt-c7000_4.1.0.LTS/bin/cl7x
C FLAGS: -mv7524 --mma_version=2_256 --abi=eabi -O3 --opt_for_speed=4 --diag_warning=225 --diag_wrap=off --display_error_number --preproc_with_compile --define=_INLINE
-- The C compiler identification is TI 4.1.0
-- The CXX compiler identification is TI 4.1.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /home/cnh20399/ti/ti-cgt-c7000_4.1.0.LTS/bin/cl7x - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /home/cnh20399/ti/ti-cgt-c7000_4.1.0.LTS/bin/cl7x - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/cnh20399/worksource/Audio/awe_custom_module/tutorial_module/build
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module/build$ make
Scanning dependencies of target dspc_tutorialObjects
[ 20%] Building C object CMakeFiles/dspc_tutorialObjects.dir/source/ModTutorialBiquad.c.o
[ 40%] Building C object CMakeFiles/dspc_tutorialObjects.dir/source/ModTutorialBiquadMono.c.o
[ 60%] Building C object CMakeFiles/dspc_tutorialObjects.dir/source/ModTutorialBiquadSmoothed.c.o
[ 80%] Building C object CMakeFiles/dspc_tutorialObjects.dir/source/ModTutorialPeakingFilter.c.o
[ 80%] Built target dspc_tutorialObjects
[100%] Linking C static library libdspc_tutorial.a
[100%] Built target dspc_tutorial
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module/build$
```

- **步骤 3**

  拷贝步骤 2 中生成的静态库到如下 AM275 的 SDK 路径：

  AM275-AWE-SDK_11.00.00.17/packages/dspc/c7x/am275x/AWECore/Lib/libdspc_tutorial.a

  在 AM275-AWE-SDK_11.00.00.17/audio_app/am275 中增加修改以下 patch 的修改

  0001-am275-add-awe-libdspc_tutorial-lib-into-AM275.patch

  0001-am275-add-aw
  e-libdspc_tutorial-lib-

  重新编译

  make scrub & make clean & make all

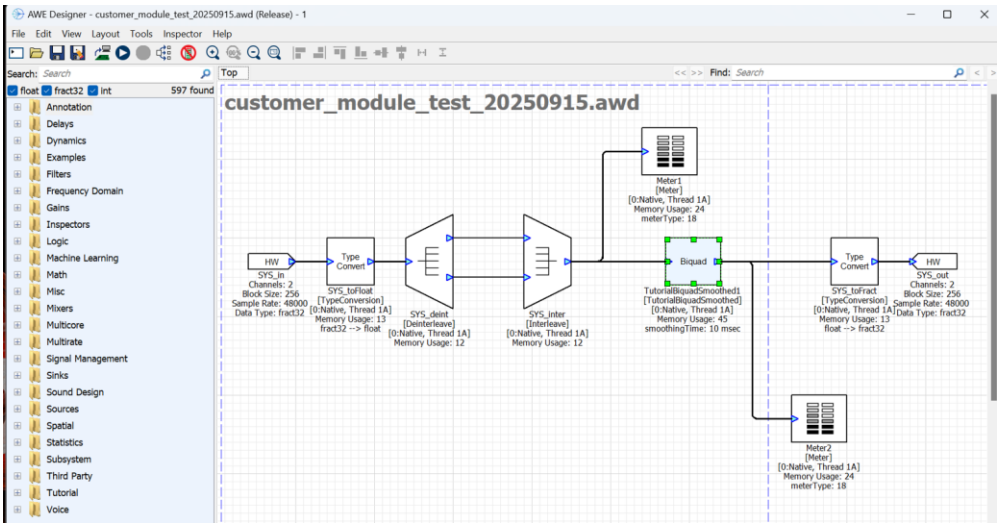
- **步骤 4**

  通过 C7x 的 map 文件 audioApp_c7x_0.release.map 检查 libdspc_tutorial 是否已经成功编入 C7x

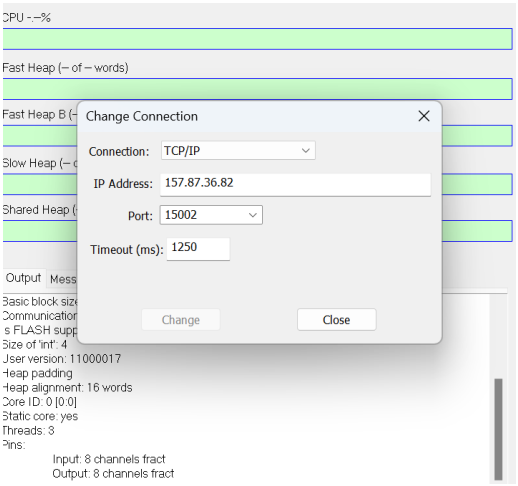
```
+--+-------------------------------------------------------+---------+---------+---------+
  Total:                                                   846904    118035    1033

/home/cnh20399/worksource/Audio/am275-11.0/AM275-AWE-SDK_11.00.00.17/packages/dspc/c7x/am275x/AWECore/Lib/libdspc_tutorial.a
  ModTutorialPeakingFilter.c.o                             1536      96        0
  ModTutorialBiquadSmoothed.c.o                            1152      88        0
  ModTutorialBiquad.c.o                                    640       88        0
  ModTutorialBiquadMono.c.o                                320       96        0
+--+-------------------------------------------------------+---------+---------+---------+
  Total:                                                   3648      368       0
```

- **步骤 5**

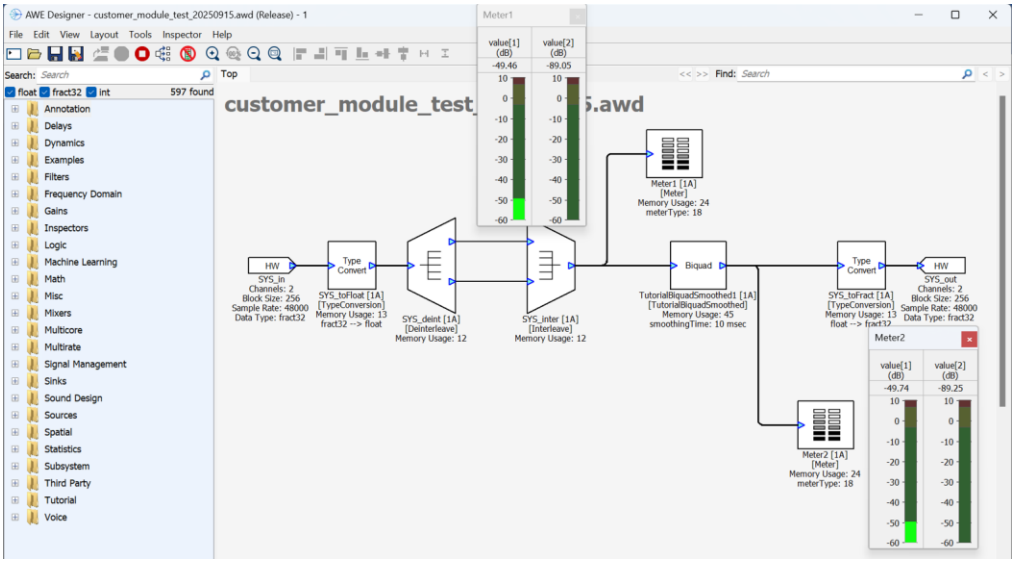
  PC 端通过 Matlab 运行 AWE 并且创建一个带有 libdspc_tutorial 模块的示例



修改 IP 链接 Target - Change Connection – TCP/IP



AM275 的 EVM 上电并且运行步骤 3 的生成的程序，PC 端的 AWE 软件通过 IP 地址成功连接到 AM275 EVM 后运行数据流，AM275 使能输入源后检查新增加的模块的输入和输出是否正常。



- **步骤 6**

  拷贝 AM275 SDK 的 mcu_plus_sdk/am275x/source/kernel/dpl/DebugP.h 到 tutorial_module/include/Targets/ 路径下。

  参考以下修改在 tutorial_module/source/ModTutorialBiquadSmoothed.c 文件中加入输出打印

```
cnh20399@cnh20399-HP-EliteBook-830-G6:~/worksource/Audio/awe_custom_module/tutorial_module/build$ git diff
diff --git a/source/ModTutorialBiquadSmoothed.c b/source/ModTutorialBiquadSmoothed.c
index a728699..ae06f7d 100644
--- a/source/ModTutorialBiquadSmoothed.c
+++ b/source/ModTutorialBiquadSmoothed.c
@@ -31,7 +31,7 @@
 #include "Framework.h"
 #include "Errors.h"
 #include "ModTutorialBiquadSmoothed.h"

+#include "DebugP.h"

 #ifdef __cplusplus
 extern "C" {
@@ -131,7 +131,7 @@ void awe_modTutorialBiquadSmoothedProcess(void *pInstance)
     FLOAT32 *src, *dst;

     // Update the filter coefficients once at the start of processing

+    DebugP_log("%s \r\n", __func__);
     {
         FLOAT32 smoothingCoeff = S->smoothingCoeff;
         INT32 i;
@@ -187,7 +187,9 @@ UINT32 awe_modTutorialBiquadSmoothedSet(void *pInstance, UINT32 mask)
     WireInstance **pWires = ClassModule_GetWires(S);
     FLOAT32 sampleRate = ClassWire_GetSampleRate(pWires[0]);
     INT32 blockSize = ClassWire_GetBlockSize(pWires[0]);

+
+    DebugP_log("%s \r\n", __func__);
+
     if (mask & MASK_TutorialBiquadSmoothed_smoothingTime)
     {
         S->smoothingCoeff = 1.0f - expf(-1.0f/((sampleRate / blockSize) * 0.001f * S->smoothingTime));
```

  重复上述步骤 3 到步骤 5 可以看到对应的日志输出。

```
Mdio_open:318
Open MAC port 1
EnetPhy_bindDriver:1873
Open MAC port 2
EnetPhy_bindDriver:1873
PHY 0 is alive
PHY 3 is alive
Starting lwIP, local interface IP is dhcp-enabled
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-0 : 70:ff:76:1d:ec:f2
[0]Enet IF UP Event. Local interface IP:0.0.0.0
[LWIPIF_LWIP] Enet has been started successfully
Cpsw_handleLinkUp:1456
MAC Port 2: link up
[0]Network Link UP Event
[0]Enet IF UP Event. Local interface IP:157.87.36.82
mcu/r5f tuning server : starting tcp server ...
mcu/r5f tuning server : Configuring IPC...
mcu/r5f tuning server : registering IPC notify clients...
mcu/r5f tuning server : sending mcu/r5f ipc_notify to code 5
mcu/r5f tuning server : sending mcu/r5f ipc_notify to code 6
mcu/r5f tuning server : Configuring IPC...
mcu/r5f tuning server : waiting for remote core
Remote Core : 5 ready
[c75ss0-0]     15.917332s : Primary core tuning server ready...
[c75ss0-0]     15.917341s : Version: 11.00.00.17
Audio Weaver Connected ...
[c75ss0-0]     27.883696s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]     27.892676s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]     27.894675s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]     28.027601s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.032932s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.038264s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.043596s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.048929s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.054261s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.059593s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.064925s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.070258s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.075590s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.080923s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.086255s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.091587s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.096920s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.102252s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.107584s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.112917s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.118249s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.123582s : awe_modTutorialBiquadSmoothedProcess
[c75ss0-0]     28.128915s : awe_modTutorialBiquadSmoothedProcess
```

3.2. 性能优化

- **DSPLIB 集成**

  通过以下示例实现把 DSPLib 的优化过的算子集成到 AWE 的 lib 中，附件的 patch 是增加一个 DSPLIB_add 的函数。增加修改后按照上述的编译方法产生静态库并且链接到 C7x 中

0001-add-DSPLIB-API-into-awe.patch

通过统计函数 CycleCounterP_getCount64 可以获得代码运行消耗的 cycle 数量，进而得到代码运行的时间。在下图日志中，opt 是 DSPLib 优化过的运行时间，nat 是标准 C 的运行时间，通过对比可以发现，经过优化后，代码的执行速度得到大幅提升。

Native C Execution Cycles	DSPLib Execution Cycles
752	217

```
icu/r5f tuning server : Configuring IPC...
icu/r5f tuning server : waiting for remote core
emote Core : 5 ready
[c75ss0-0]    16.052481s : Primary core tuning server ready...
[c75ss0-0]    16.052489s : Version: 11.00.00.17
udio Weaver Connected ...
[c75ss0-0]   125.336842s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]   125.345852s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]   125.348831s : awe_modTutorialBiquadSmoothedSet
[c75ss0-0]   125.462232s : _x_input = 0x000000007E003740, _y_input = 0x000000007E003B80, _output = 0x000000007E003FC0, _
[c75ss0-0]   125.462254s : outputnat = 0x000000007E004400
[c75ss0-0]   125.462268s : opt add init cycle 338, start = 195044324536, end = 195044324874
[c75ss0-0]   125.462290s : opt add exec cycle 217, start = 195044347240, end = 195044347457, status_opt = 0
[c75ss0-0]   125.462313s : nat add init cycle 70, start = 195044370509, end = 195044370579
[c75ss0-0]   125.462335s : nat add exec cycle 752, start = 195044391333, end = 195044392085, status_nat = 0
[c75ss0-0]   125.462360s : 1-Warning at: row=0, col=0, val1=-0.953292, val2=-0.953292
[c75ss0-0]   125.462461s : TI_compare_mem_2D_float status_ref_vs_opt = 0
[c75ss0-0]   125.462474s : 1-Warning at: row=0, col=0, val1=-0.953292, val2=-0.953292
[c75ss0-0]   125.462573s : TI_compare_mem_2D_float status_ref_vs_nat = 0
[c75ss0-0]   125.462584s : ae_vecvaddf Done
[c75ss0-0]   132.536824s : awe_modTutorialBiquadSmoothedSet
```

- **内存优化**

为更进一步优化程序的执行时间，可以通过以下针对 linker 文件的修改把关键函数的代码段，数据段都通过编译器指定到访问速度更快的内存上。

```
diff --git a/audio_app/am275/c75_0/ti-c7000/linker.cmd b/audio_app/am275/c75_0/ti-c7000/linker.cmd
index 6c9d5e2c..98a686f3 100644
--- a/audio_app/am275/c75_0/ti-c7000/linker.cmd
+++ b/audio_app/am275/c75_0/ti-c7000/linker.cmd
@@ -90,4 +90,45 @@ SECTIONS
    /* AVB shared memory buffer */
    .asdk_avb_rx_buffer (NOLOAD)  > SHM_AVB_PCM_DATA_RX
    .asdk_avb_tx_buffer (NOLOAD)  > SHM_AVB_PCM_DATA_TX
+
+   /* Customized modules */
+   .customized_mod:
+   {
+      libdspc_tutorial.a(.text)
+      libdspc_tutorial.a(.const)
+      libdspc_tutorial.a(.data)
+      libdspc_tutorial.a(.bss)
+      DSPLIB_C7524.lib<DSPLIB_add.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_add_ci.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_add_ci.cpp.o>(.const)
+      DSPLIB_C7524.lib<DSPLIB_add_cn.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_addConstant.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_addConstant_ci.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_addConstant_ci.cpp.o>(.const)
+      DSPLIB_C7524.lib<DSPLIB_addConstant_cn.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_mulConstant.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_mulConstant_ci.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_mulConstant_ci.cpp.o>(.const)
+      DSPLIB_C7524.lib<DSPLIB_subConstant.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_subConstant_ci.cpp.o>(.text)
+      DSPLIB_C7524.lib<DSPLIB_subConstant_ci.cpp.o>(.const)
+      DSPLIB_C7524.lib<DSPLIB_mul.cpp.o>(.text)
```

```
+       DSPLIB_C7524.lib<DSPLIB_mul_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_mul_ci.cpp.o>(.const)
+       DSPLIB_C7524.lib<DSPLIB_sub.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_sub_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_sub_ci.cpp.o>(.const)
+       DSPLIB_C7524.lib<DSPLIB_dotprod.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_dotprod_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_dotprod_ci.cpp.o>(.const)
+       DSPLIB_C7524.lib<DSPLIB_matMul.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_matMul_cn.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_matMul_cn.cpp.o>(.data)
+       DSPLIB_C7524.lib<DSPLIB_matMul_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_matMul_generic_size_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_matMul_generic_size_ci.cpp.o>(.const)
+       DSPLIB_C7524.lib<DSPLIB_matMul_N_unroll_ci.cpp.o>(.text)
+       DSPLIB_C7524.lib<DSPLIB_matMul_N_unroll_ci.cpp.o>(.const)
+   } > L2SRAM
+
 }
```

4. 总结

Sitara™ AM275 系列是 TI 为下一代音频系统设计的高性能 MCU 产品，其中集成 TI 最新 C7000 系列 DSP，拥有强大的浮点计算能力，可以实现音频中所有的高性能音效算法。复杂的音频处理流程需要一套完整的数据流和算法框架，Audio Waver（下文中简称 AWE）是当前 AM275 系列产品最主要应用的音频框架，并提供与此适配的 AWE SDK 进行产品应用开发。本篇系统介绍了 AM275 AWE SDK 的基础安装编译方法以及自定义模块的开发和集成方法。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月