# TI DeepLearning Library环境配置笔记

## 概述

TI DeepLearning Library是TI Vision SDK下的一个组件。组件安装位置位于

/PROCESSOR\_SDK\_VISION\_03\_04\_00\_00/ti\_components/algorithms/REL\_TIDL.01.01.01.00（或者PROCESSOR\_SDK\_VISION\_03\_01\_00\_00ti\_components/algorithms/REL\_TIDL.00.08.00.00）

对应的使用说明文档为TIDeepLearningLibrary\_UserGuide.pdf

https://usermanual.wiki/Document/TIDeepLearningLibraryUserGuide.1793363394/html#

## 关键词

* **TIDL** Ti DeepLearning Library
* **XDAIS** eXpressDSP Algorithm Interface Standard
* **IVISION** an extension of XDAIS
* **TDAx** TI's Driver Asistance System-on-Chip family
* **vxLib** Ti's Vision Processing Functions Library for C programmers using c66x devices
* **TIDL嵌入式二进制程序：**运行于TI处理器（C6000、EVE）上的二进制程序，一般以.out为扩展名，由gmake编译产生
* **TIDL PC模拟程序：**运行于PC端，用于模拟TI处理器上深度学习计算的程序，一般以.exe为扩展名，由gmake附加特殊编译指令产生。

## **安装配置说明**

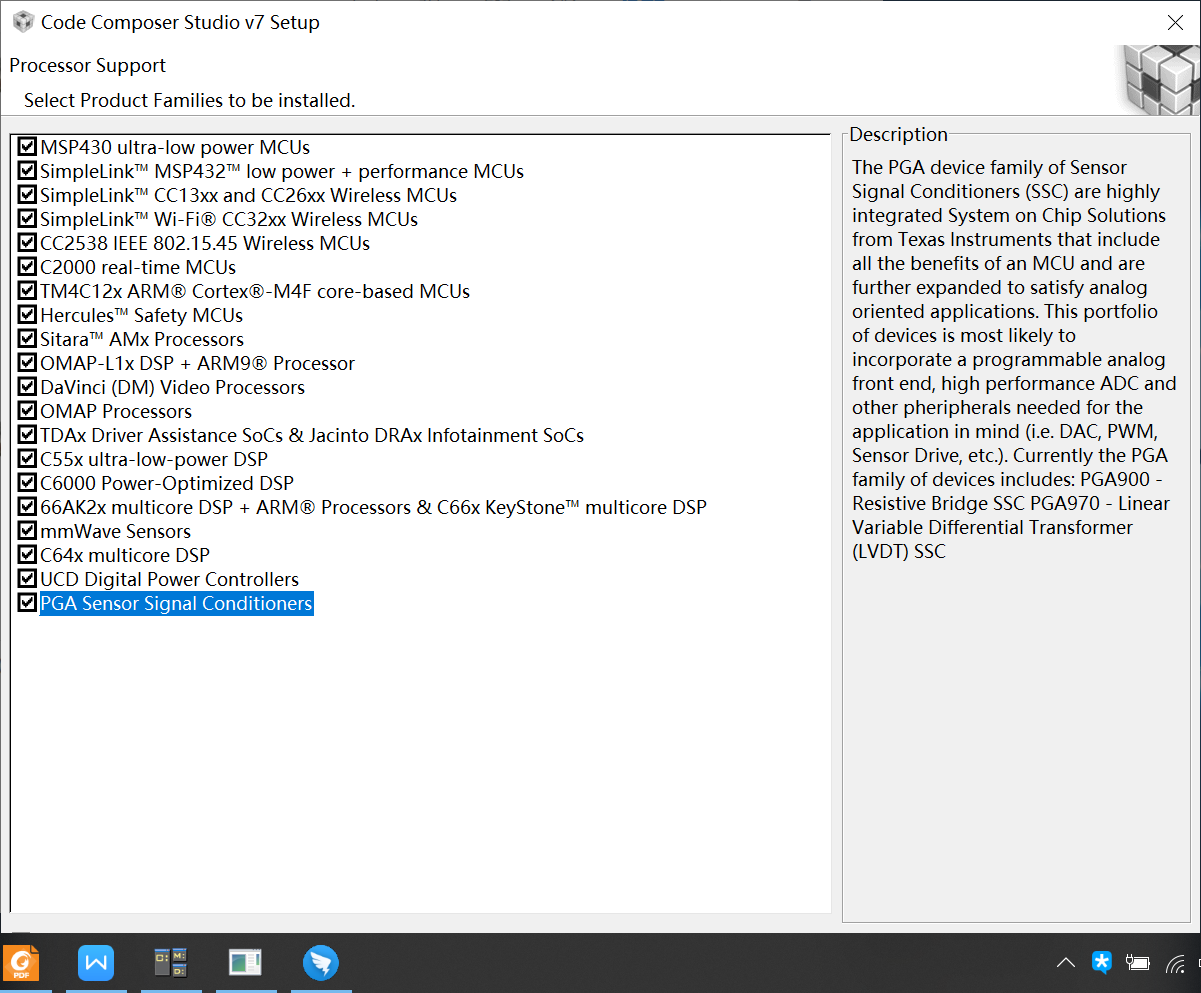
1. 安装visual studio 2013，选择默认安装配置
2. **安装TDA2x\_TI\_VSION\_SDK 3.04版**，执行 PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\_setupwin.exe，选择默认安装配置,安装位置选择D:\

**注**：该版本为TIDL官方最新最稳定的版本，支持TIDL嵌入式二进制程序和PC模拟程序的编译生成和简单的测试。但缺少tidl\_algo核心计算的代码，只有编译好的lib。此外，已查明tidl\_import工具的batchnorm层存在bug。

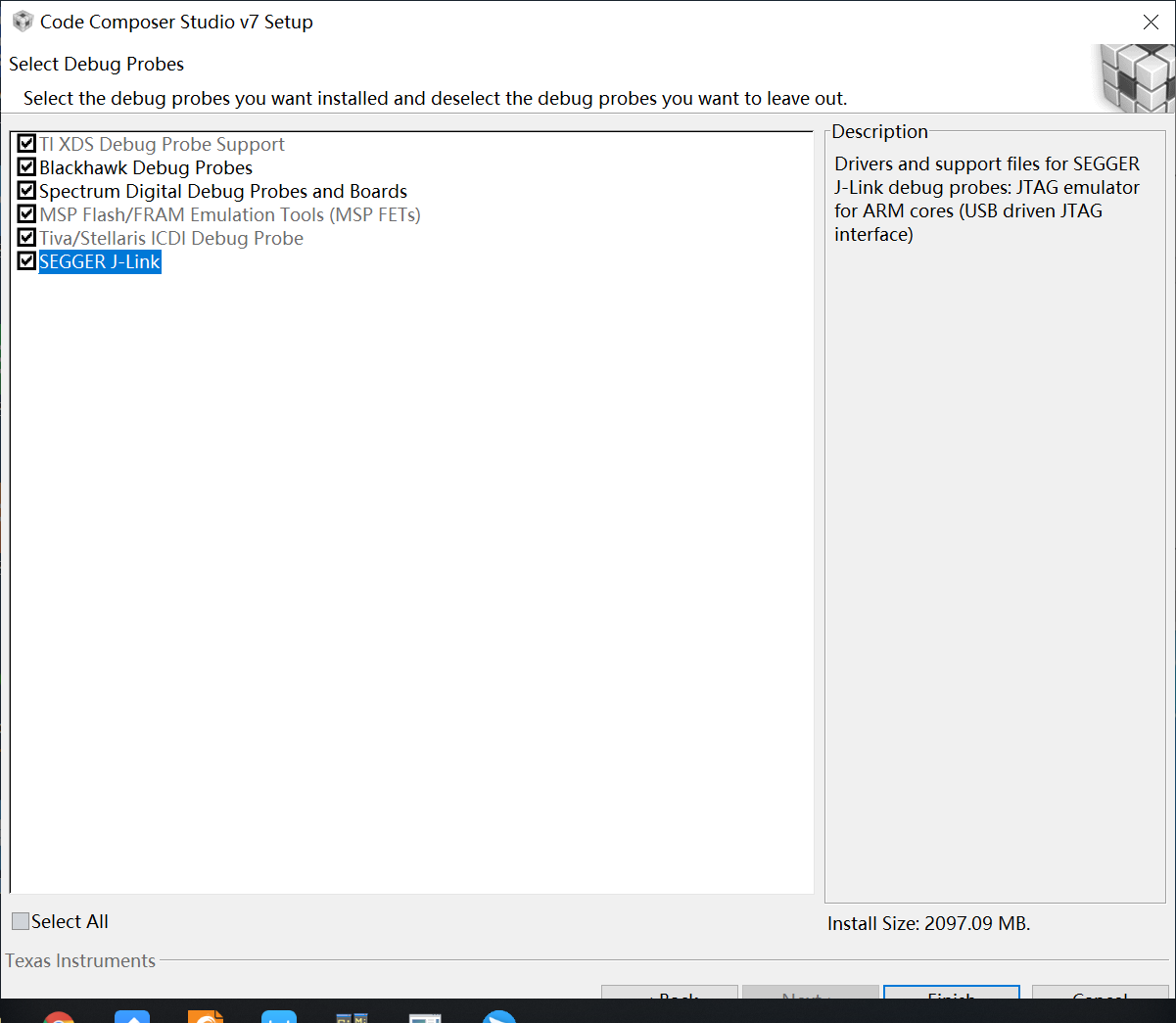
1. **此步骤可以和步骤4同步进行。安装TDA2x\_TI\_VSION\_SDK 3.01版**，执行PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\_setupwin.exe，选择默认安装配置,安装位置选择D:\（该版本包含tidl\_algo核心代码，可以编译生成EVE平台嵌入式二进制程序和PC模拟程序，支持任意修改代码。但不支持DSP平台的嵌入式二进制程序编译，因为缺少某些头文件）

注：因3.01版可操作性较强，后面安装配置以3.01为例进行解说。3.04版的编译配置与3.01类似。

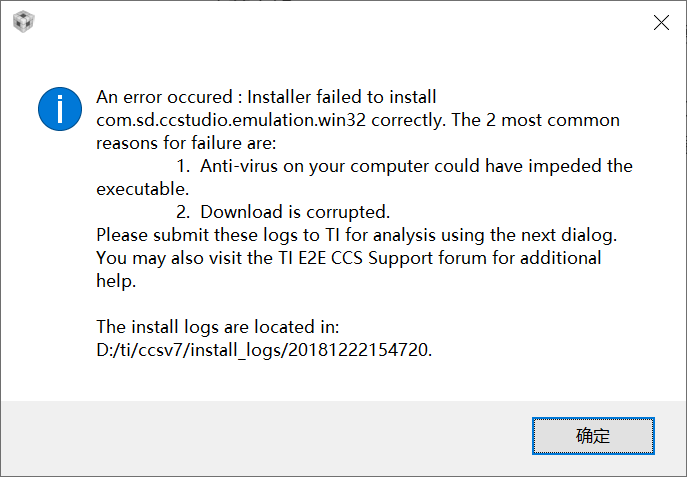
1. **安装CCSv7**，暂停Semantic Endpoint Protection，运行ccs\_setup\_7.3.0.00019.exe.



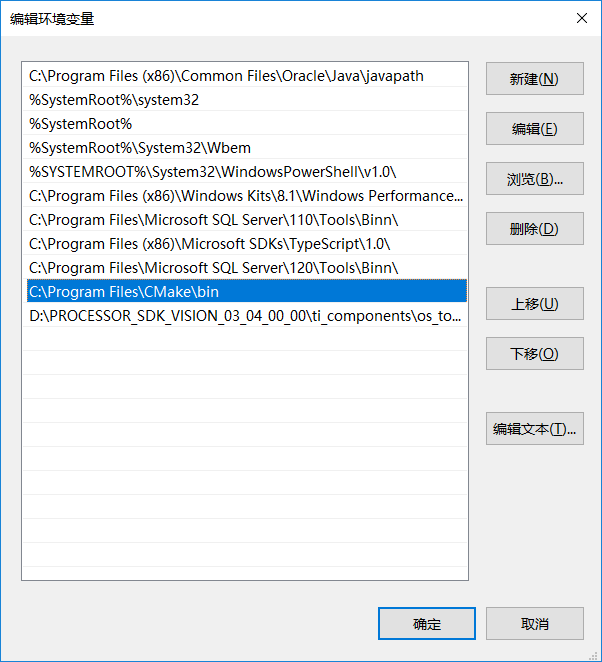
选择device family时可以全部选上。



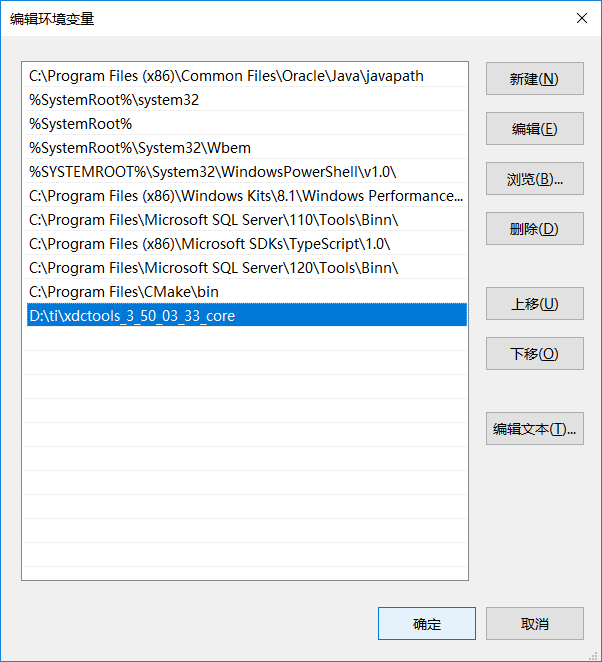
Debug probe只选择第一个，其他不用选，不然会报错：



1. **安装cmake，**运行cmake-3.9.0-rc3-win64-x64.msi，安装过程中如果提问是否设置环境变量，选择为所有用户设置环境变量，安装完成后可以看到环境变量设置结果：



1. **配置gmake环境变量：**

将gmake环境变量配置为当前系统内xdctools安装的目录下，例如选择D:\ti\xdctools\_3\_50\_03\_33\_core\。

1. **配置DMAUTILS环境变量：**

在系统环境变量中添加“DMAUTILS\_PATH”变量，例如在本机中指向“D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.DMAUTILS.00.08.00.02\dmautils”.

（指向PROCESSOR\_SDK\_VISION\_03\_04\_00\_00内的dmautils也可）

1. **配置XDAIS环境变量**

将XDAIS程序包放置于CCSv7目录下，在系统环境变量中添加“XDAIS\_PATH”变量，指向“D:\ti\ccsv7\xdais\_7\_24\_00\_04”.

1. **配置ARP32和C6000环境变量**

在系统环境变量中添加“DSP\_TOOLS”变量，指向“D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\C6000\_7.4.2”.

（指向PROCESSOR\_SDK\_VISION\_03\_04\_00\_00也可）

在系统环境变量中添加“ARP32\_TOOLS”变量，指向“D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\arp32\_1.0.7”.

（指向PROCESSOR\_SDK\_VISION\_03\_04\_00\_00也可）

1. **配置VC环境变量**

在运行编译前，需要对当前活动会话配置VC环境变量，否则编译时可能会报出“'LINK' 不是内部或外部命令，也不是可运行的程序

或批处理文件。”这样的错误。

要设置环境变量，只需要在当前控制台的活动会话中执行“E:\"Program Files (x86)"\"Microsoft Visual Studio 12.0"\VC\vcvarsall.bat”即可。

1. **编译测试样例程序：**

一共有两个版本的程序都需要进行编译，3.1和3.4的：

3.1编译：

将下面文件中对应变量修改如下：D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.TIDL.00.08.00.00\makerules\config.mk

DSP\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\C6000\_7.4.2"

ARP32\_TO

DSP\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\C6000\_7.4.2"

ARP32\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\arp32\_1.0.7"

TIARM\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\ti-cgt-arm\_16.9.2.LTS"

XDAIS\_PATH ?="C:\ti\ccsv7\xdais\_7\_24\_00\_04"

UTILS\_PATH ?="C:\ti\ccsv7\utils\cygwin"

VXLIB\_PATH ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\vxlib\_c66x\_1\_1\_1\_0"

DMAUTILS\_PATH ?=$(ALGBASE\_PATH)/dmautils

ALGFRAMEWORK\_PATH ?=$(ALGBASE\_PATH)/algframework

MATHLIB\_INSTALL\_DIR ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\mathlib\_c66x\_3\_1\_0\_0"OLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\arp32\_1.0.7"

TIARM\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\cg\_tools\windows\ti-cgt-arm\_16.9.2.LTS"

XDAIS\_PATH ?="C:\ti\ccsv7\xdais\_7\_24\_00\_04"

UTILS\_PATH ?="C:\ti\ccsv7\utils\cygwin"

VXLIB\_PATH ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\vxlib\_c66x\_1\_1\_1\_0"

DMAUTILS\_PATH ?=$(ALGBASE\_PATH)/dmautils

ALGFRAMEWORK\_PATH ?=$(ALGBASE\_PATH)/algframework

MATHLIB\_INSTALL\_DIR ?="D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\mathlib\_c66x\_3\_1\_0\_0"

进入以下路径： D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.TIDL.00.08.00.00\modules\ti\_dl\test

在当前Makefile目录下运行 gmake CORE=eve all 或者 gmake CORE=dsp all；编译通过即证明编译环境设置正确。

3.4编译

打开以下文件，D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\algorithms\REL.TIDL.01.01.01.00\makerules\config.mk

将对应变量修改：

DSP\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\cg\_tools\windows\C6000\_7.4.2"

ARP32\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\cg\_tools\windows\arp32\_1.0.7"

TIARM\_TOOLS ?="D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\cg\_tools\windows\ti-cgt-arm\_16.9.2.LTS"

XDAIS\_PATH ?="C:\ti\ccsv7\xdais\_7\_24\_00\_04"

UTILS\_PATH ?="C:\ti\ccsv7\utils\cygwin"

VXLIB\_PATH ?="D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\algorithms\vxlib\_c66x\_1\_1\_1\_0"

DMAUTILS\_PATH ?=$(ALGBASE\_PATH)/dmautils

ALGFRAMEWORK\_PATH ?=$(ALGBASE\_PATH)/algframework

MATHLIB\_INSTALL\_DIR ?="D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\algorithms\mathlib\_c66x\_3\_1\_0\_0"

SHOW\_COMMANDS ?= 0

LINUXENV ?= x86

LINUX\_BUILD\_TOOLS ?="/home/uid/tools/linaro-2017.11"

进入以下路径： D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\algorithms\REL.TIDL.01.01.01.00\modules\ti\_dl\test

在当前Makefile目录下运行 gmake CORE=eve all 或者 gmake CORE=dsp all；编译通过即证明编译环境设置正确。

备注：PROCESSOR\_SDK\_VISION\_03\_04\_00\_00版本工程中只能进行DSP和EVE核的二进制文件生成和调试，无法进行PC端仿真，因为缺少algo文件夹；因此后续的仿真模拟工作需要在PROCESSOR\_SDK\_VISION\_03\_01\_00\_00中进行

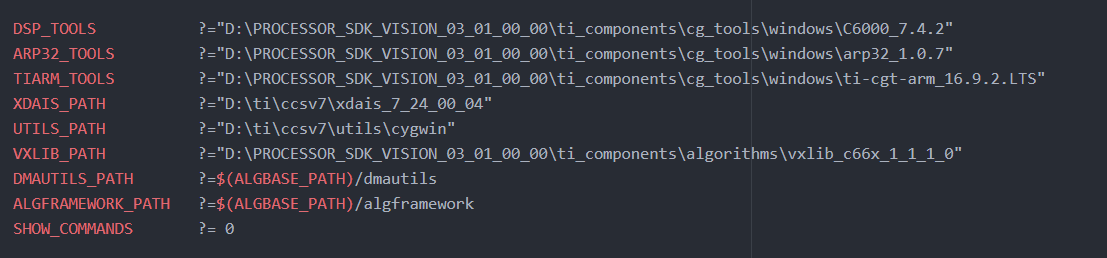
1. **配置PC仿真平台**

* 进入

D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.TIDL.00.08.00.00\modules\ti\_dl

* 此步骤需要在系统自带cmd下运行“E:\"Program Files (x86)"\"Microsoft Visual Studio 12.0"\VC\vcvarsall.bat”将VC环境变量设置到当前工作环境（vs命令行下可不需要，powershell环境下该命令无效）
* 首先运行“gmake CORE=eve ”，此步骤会生成host emulation 模式下需要使用的vcop\_cnn\_argmax\_kernel.h等文件，如果不执行这一步，后续编译会缺少文件。
* cd..\algo
* gmake CORE=eve TARGET\_BUILD=debug TARGET\_PLATFORM=PC，生成PC仿真平台eve核debug对应的TIDL算法库：生成的文件存放于”..\lib\PC\eve\debug\tidl\_algo.lib”

备注：在运行上述命令前，需要修改“..\..\makerules\config.mk”内的路径变量



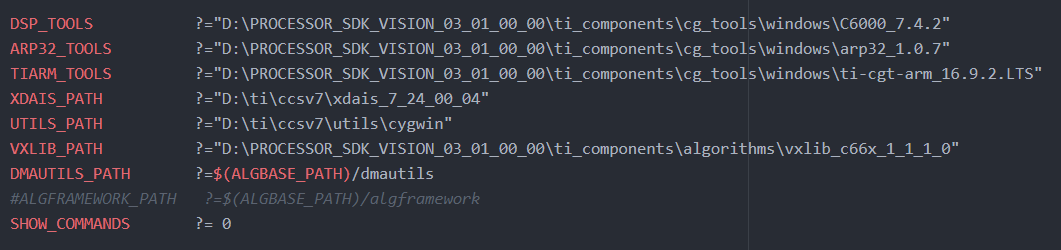
（遗留问题：在CORE=eve时编译可以正常通过，但在CORE=dsp时，编译时报错".\.\src\tidl\_spatialPooling.c", line 38: fatal error: could not open source file "VXLIB\_cnn\_maxPooling\_2x2\_skip2\_i8s\_o8s.h"，表明不支持dsp平台编译）

* 编译DMA

进入

“D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.DMAUTILS.00.08.00.02\dmautils”路径，运行“gmake CORE=eve TARGET\_BUILD=debug TARGET\_PLATFORM=PC”生成dmautils.lib

备注：编译前仍然需要修改"..\config.mk"，将其中的相关路径配置好



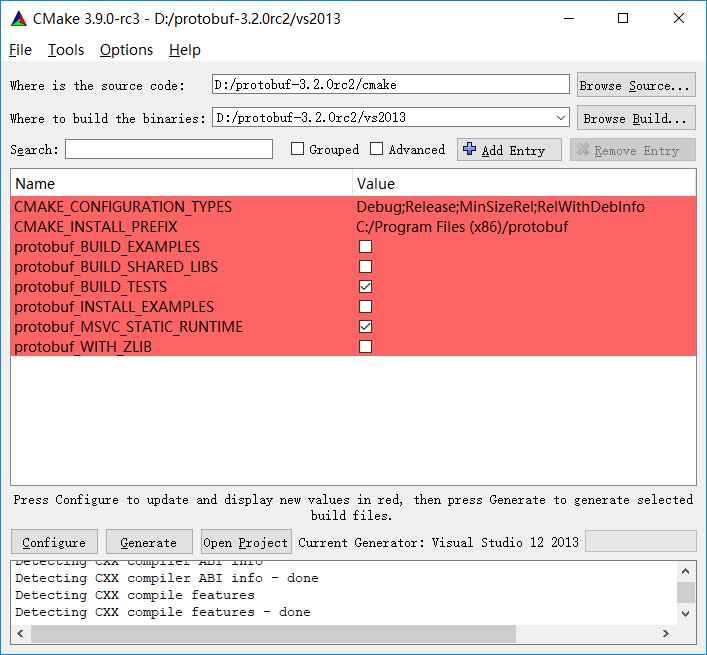
（遗留问题：在CORE=eve时编译可以正常通过，但在CORE=dsp时，编译时报错LINK1112：x86与x64模块不符；此问题直接导致test程序无法运行dsp模拟，因无法加载支持dsp模拟的dmautils.lib）.

刘洋编译dsp版本时没有问题。

1. **编译配置Protobuf（ Google Protocol Buffer (简称 Protobuf)**

是 Google 公司内部的混合语言数据标准，目前已经正在使用的有超过 48,162 种报文格式定义和超过 12,183 个 .proto 文件。他们用于 RPC 系统和持续数据存储系统。**）**

* 解压缩protobuf压缩包，打开cmake工具，清除cmake的cache后，将source code栏指向protobuf内的cmake文件夹内，再将binaries栏指向某个希望生成vs2013工程的位置
* 分别点击configure、Generate、Open Project



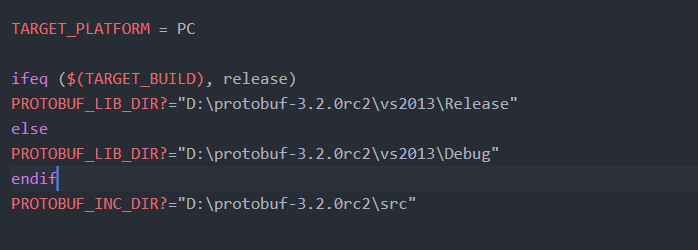
* 在打开的vs2013工程内，编译debug和release库即可。

1. **编译运行tidl\_model\_import.out.exe**

* 切换至路径

“D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.TIDL.00.08.00.00\modules\ti\_dl\utils\tidlModelImport”

* 修改该路径下的Makefile内的三个路径设置：



ifeq ($(TARGET\_BUILD), release)

PROTOBUF\_LIB\_DIR?="D:\protobuf-cpp-3.2.0rc2\vs2013\Release"

else

PROTOBUF\_LIB\_DIR?="D:\protobuf-cpp-3.2.0rc2\vs2013\Debug"

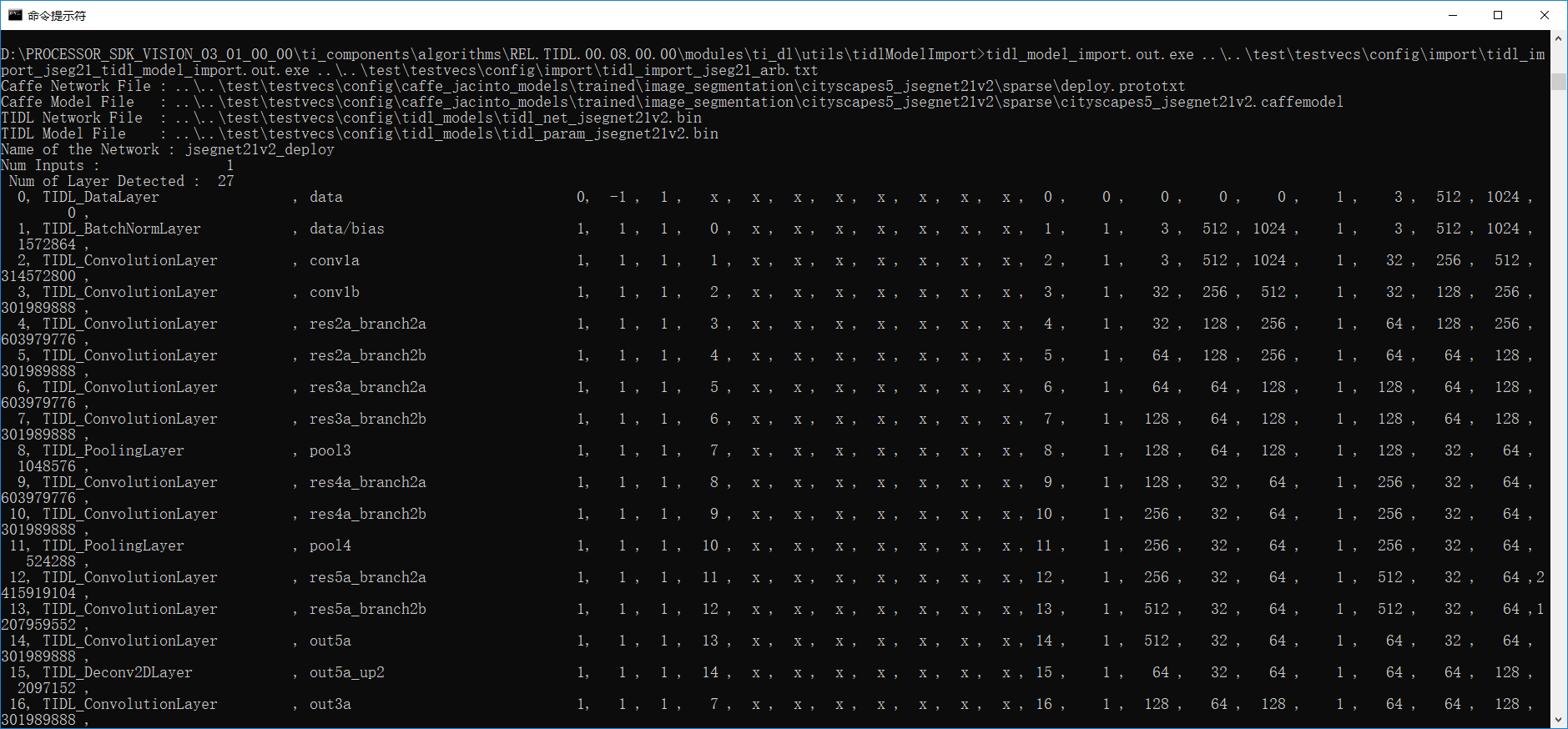
endif

PROTOBUF\_INC\_DIR?="D:\protobuf-cpp-3.2.0rc2\src"

在命令行中输入：E:\"Program Files (x86)"\"Microsoft Visual Studio 12.0"\VC\vcvarsall.bat

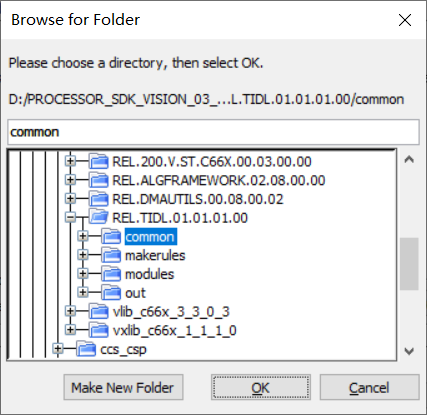
* 配置完成后，在该路径下运行“gmake CORE=eve TARGET\_BUILD=debug TARGET\_PLATFORM=PC”，编译后生成“tidl\_model\_import.out.exe”文件
* 在当前目录下运行

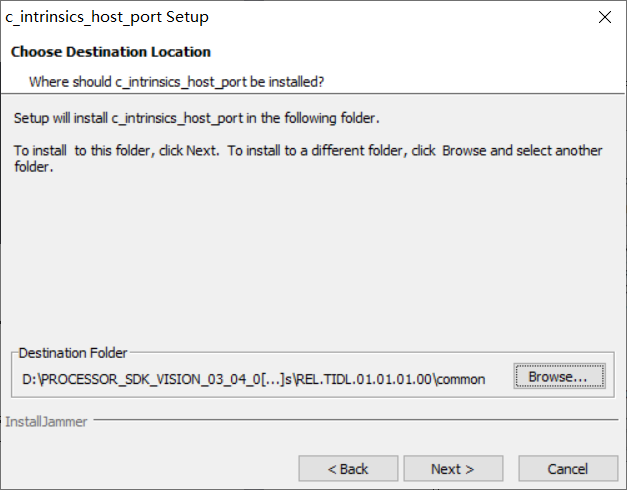
“tidl\_model\_import.out.exe ..\..\test\testvecs\config\import\tidl\_import\_jseg21.txt”，程序会读取caffe模型数据，并进行相应的处理，给出输出结果。（需要提前到<https://github.com/tidsp/caffe-jacinto-models>下载caffe模型）



15, 编译eve\_test\_dl\_algo.out.exe

1. 安装c\_intrinsics\_host\_port-0\_91\_00-Setup.exe 安装路径选择tidl common 路径





1. 编译eve\_test\_dl\_algo.out.exe 和 dsp\_test\_dl\_algo.out.exe

{TIDL\_install\_path}： D:\PROCESSOR\_SDK\_VISION\_03\_01\_00\_00\ti\_components\algorithms\REL.TIDL.00.08.00.00\modules\ti\_dl

3.4版本：

D:\PROCESSOR\_SDK\_VISION\_03\_04\_00\_00\ti\_components\algorithms\REL.TIDL.01.01.01.00\modules\ti\_dl

VS2013环境变量 E:\"Program Files (x86)"\"Microsoft Visual Studio 12.0"\VC\vcvarsall.bat

{TIDL\_install\_path} > gmake CORE=eve TARGET\_BUILD=debug

TARGET\_PLATFORM=PC all

TIDL\_install\_path} > gmake CORE=dsp TARGET\_BUILD=debug

TARGET\_PLATFORM=PC all

上面两句话可以生成host emulation的可执行程序

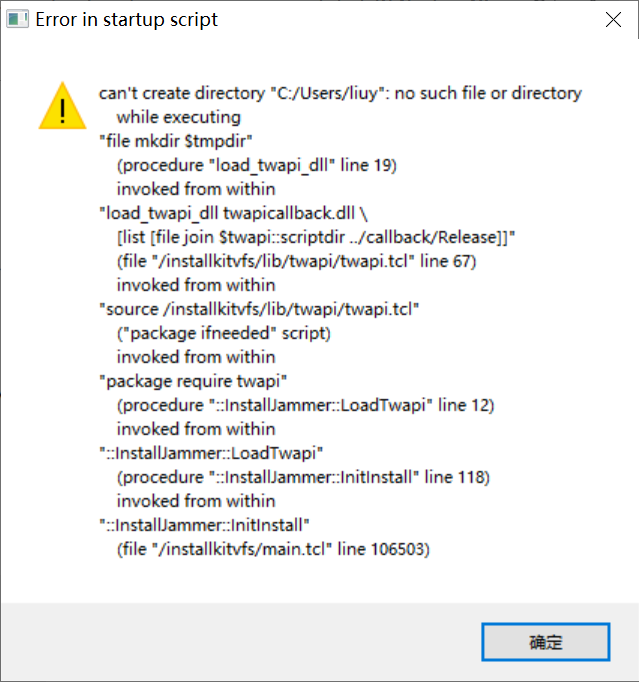
{TIDL\_install\_path}\test\out\eve\_test\_dl\_algo.out.exe

{TIDL\_install\_path}\test\out\dsp\_test\_dl\_algo.out.exe

至此，TIDL windows开发环境的安装配置完成。

Bug List:

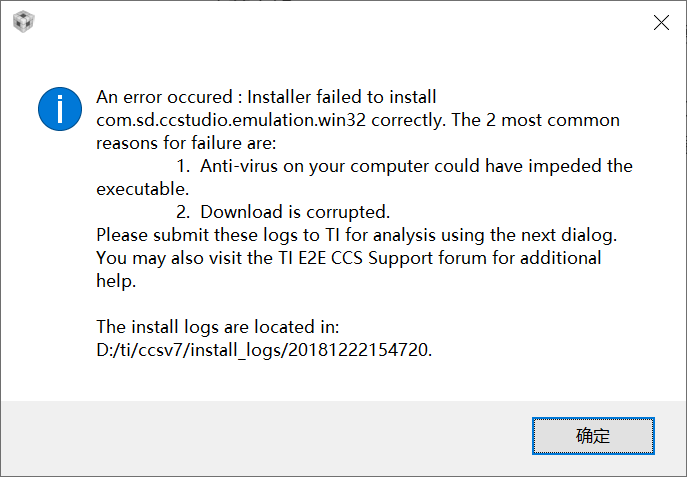
1,安装**TDA2x\_TI\_VSION\_SDK 3.01时报错如下：**



解决办法：

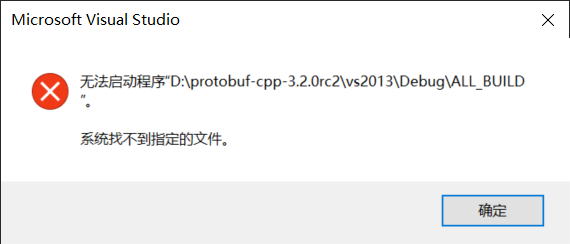
将**TDA2x\_TI\_VSION\_SDK 3.01.exe**复制到c盘根目录下，再以管理员权限运行就好了。

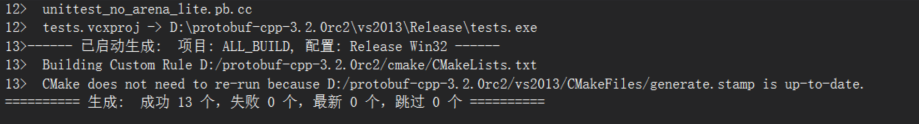
1. 安装ccs v7时报错：



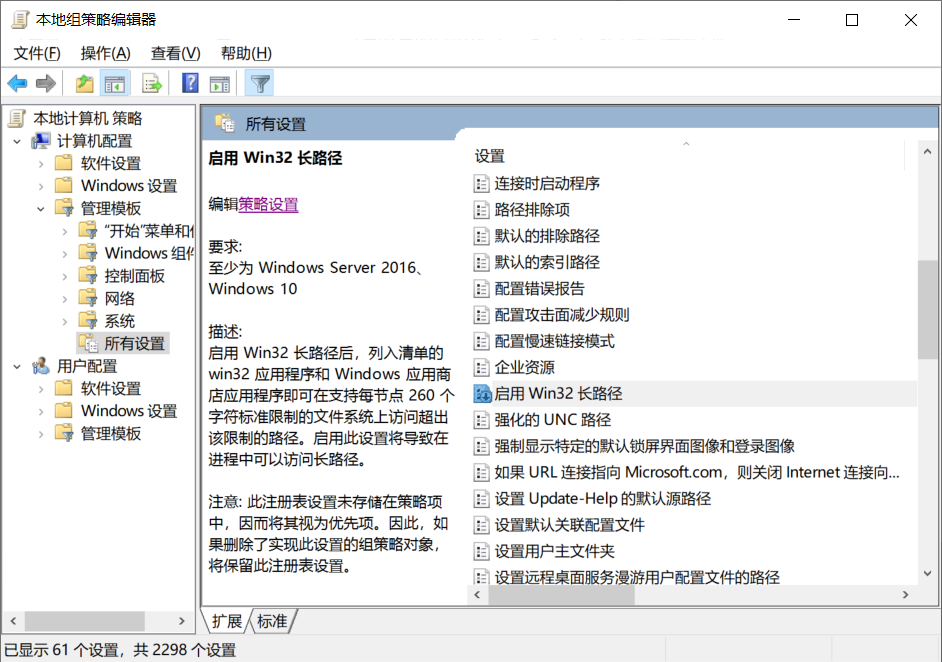
在选择debugger probe时，只选择第一个，就好了。

1. 执行protobuf编译时报错，但是vs console里面编译没有问题，这个错误是vs 没有找到可执行文件，我们编译的是库，没有可执行程序所以报错，这个不用理会。





4,拷贝caffe model时有些模型路径太长会拷贝失败，以下为win10下解决路径太长无法复制问题解决办法：<https://blogs.msdn.microsoft.com/jeremykuhne/2016/07/30/net-4-6-2-and-long-paths-on-windows-10/>)



5, 如果报错：cl不是内部或外部命令也不是可运行的程序。说明VS2013的环境变量没有设置好，在cmd中执行以下脚本：E:\"Program Files (x86)"\"Microsoft Visual Studio 12.0"\VC\vcvarsall.bat。环境变量只在当前cmd环境中有效。