**1、     概述**

微软在”WDK7600“以后就不再提供独立的内核驱动开发包了，而是必须首先安装微软集成开发环境VisualStudio，然后再从微软官网下载集成的WDK开发包、或者离线安装包，但是安装后Visual Studio就集驱动程序开发，编译，安装，部署和调试于一身，使得Windows驱动程序开发会变得更容易。对于Windows 10和Visual Studio2015驱动开发WDK环境变量的配置，与其它版本的Windows和WDK环境的配置有很大不同。本人也是苦苦探索了近一周的时间才配好。

**2、     WDF驱动模型**

设备驱动程序是硬件设备连接到计算机系统的软件接口，任何设备都必须有相应的驱动程序才能在计算机系统上正常工作。设备驱动程序的优劣直接关系到整个系统的性能和稳定性，因此，设计和开发稳定高效的驱动程序具有重要意义。

早期的Windows 95/98的设备驱动是VxD(Virtual DeviceDriver)，其中x表示某一类设备。从Windows 2000开始，开发驱动[程序](http://www.xuebuyuan.com/" \o "程序" \t "_blank)必以WDM（Windows Driver Model）为基础的，但是，如果使用DDK来开发WDM，其开发难度之大，根本不能奢望像用户模式应用程序开发那样容易，因此，一般用户都是使用WinDriver、DriverStudio之类的第三方工具。为改善这种局面，从Vista开始，微软推出了新的驱动程序开发环境WDF（Windows Driver Foundation ）。WDF(WindowsDriver Foundation)是微软提出的下一代全新的驱动程序模型，它是在WDM(windowsDriver Model)的基础上发展而来的，支持面向对象、事件驱动的驱动程序开发，提供了比WDM更高层次抽象的高度灵活、可扩展、可诊断的驱动程序框架。WDF框架管理了大多数与操作系统相关的交互，实现了公共的驱动程序功能（如电源管理、PnP支持），隔离了设备驱动程序与操作系统内核，降低了驱动程序对内核的影响。

WDF提供了两个框架：KMDF(内核模式驱动程序框架)和UMDF（用户模式驱动程序框架）。

1、内核模式驱动程序 KMDF（Kernel-Mode DriverFramework）：

这类驱动程序作为内核模式操作系统组件的一部分执行，它们管理I/O、即插即用、内存、进程和线程、安全等。内核模式驱动程序通常为分层结构。KMDF是Windows系统底层驱动，文件名为：\*.SYS。关于KMDF更多的内容，可参阅 MSDN中“Getting Started with Kernel-ModeDriver Framework ”。

2、用户模式驱动程序 UMDF（User-Mode DriverFramework）：

这类驱动程序通常提供 Win32 应用程序与内核模式驱动程序或其他操作系统组件之间的接口。用户模式驱动程序支持基于协议或基于串行总线（如摄像机和便携音乐播放器）的设备。UMDF是用户层驱动，文件名为：\*.DLL。关于KMDF更多的内容，可参阅 MSDN中“Introduction to UMDF“。

无论内核模式的驱动程序或者用户模式的驱动程序，都使用同一环境进行构建，这一环境称为WDK；都采用同一套对象模型构建，采用同一个基础承载，这个基础就是WDF。由于WDF驱动模型提供了面向对象和事件驱动的驱动程序开发框架，大大降低了开发难度。从现在开始，掌握Windows设备驱动程序的开发人员，由过去的“专业”人士，将变为“普通”大众。因此，像WinDriver、DriverStudio之类的第三方工具也随之退出历史舞台。更重要的，也是微软反复炫耀的是封装了[驱动程序](http://baike.baidu.com/view/1048.htm" \t "_blank)中的某些共同行为：例如即插即用和电源管理就属于这种共同行为。因为大多数驱动程序中都需要处理即插即用和电源管理问题，据说这大概要上千行的代码，况且，没有相当水平还不一定能处理好。为了一劳永逸，WDF干脆将即插即用和电源管理封装了进了对象之内，一举成了对象的缺省（默认）行为。WDF将驱动程序与操作系统内核之间进行了分离，驱动程序与操作系统交互工作交给框架内封装的方法（函数）完成，这样驱动开发者只需专注处理硬件的行为即可。这不仅避免了顾此失彼两面不周的弊端，也由于双方的分离，对操作系统内的某些改动，硬件制造商配套[驱动程序](http://baike.baidu.com/view/1048.htm" \t "_blank)的开发都有莫大的好处。

**3、     用户模式和内核模式**

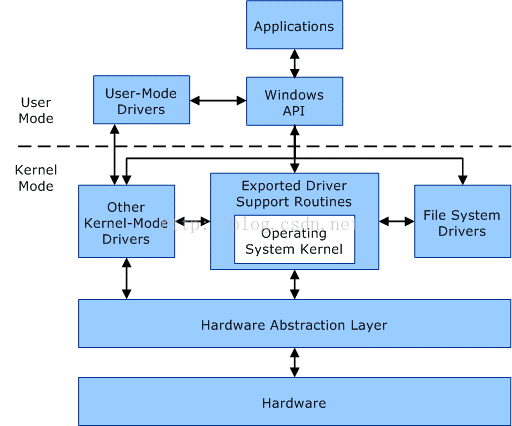
运行 Windows 的计算机中的处理器有两个不同模式：“用户模式”和“内核模式”。根据处理器上运行的代码的类型，处理器在两个模式之间切换。应用程序在用户模式下运行，核心操作系统组件在内核模式下运行。多个驱动程序在内核模式下运行时，某些驱动程序可能在用户模式下运行。

当启动用户模式的应用程序时，Windows 会为该应用程序创建“进程”。进程为应用程序提供专用的“虚拟地址空间”和专用的“句柄表格”。由于应用程序的虚拟地址空间为专用空间，一个应用程序无法更改属于其他应用程序的数据。每个应用程序都孤立运行，如果一个应用程序损坏，则损坏会限制到该应用程序。其他应用程序和操作系统不会受该损坏的影响。

用户模式应用程序的虚拟地址空间除了为专用空间以外，还会受到限制。在用户模式下运行的处理器无法访问为该操作系统保留的虚拟地址。限制用户模式应用程序的虚拟地址空间可防止应用程序更改并且可能损坏关键的操作系统数据。

在内核模式下运行的所有代码都共享单个虚拟地址空间。这表示内核模式驱动程序未从其他驱动程序和操作系统自身独立开来。如果内核模式驱动程序意外写入错误的虚拟地址，则属于操作系统或其他驱动程序的数据可能会受到损坏。如果内核模式驱动程序损坏，则整个操作系统会损坏。

此图说明了用户模式组件与内核模式组件之间的通信。



**4、     驱动模型特点**

UMDF和传统的驱动程序差异非常大，简要说来，UMDF是这样的：

* UMDF是基于COM思想的，运行于用户模式（RING3）的驱动程序模块。

那么，这种驱动模型带来什么变化呢？

首先基于COM思想，引入接口机制，可以把相关联的函数分门别类进行组织，使得驱动代码清晰明了；其次，运行在RING3的驱动，大幅度降低了驱动程序在稳 定性和安全性上面的风险，UMDF驱动崩溃不会导致bugcheck（蓝屏），并且UMDF驱动的宿主进程是在受限的用户身份下运行的，不是受信任的系统内核模块。可以在UMDF里面使用Win32 API。

运行于RING3的UMDF对于程序员开说至少带来两个额外好处：

* 驱动程序不需要强制数字签名，因为UMDF驱动不是系统信任模块，所以在x64下面的部署更加方便。特别是个人开发者可能无法承受WHQL的费用，或者其他原因暂时无法WQHL的情况下，使用UMDF是一个较好的选择。
* 调试难度大幅降低，不再需要SoftICE和Syser之类的单机内核调试器或者WinDBG之类的双机调试，我们可以用WinDBG或者VS调试器attach到UMDF宿主进程即可进行调试。可以参考[调试UMDF驱动](http://hyperiris.wordpress.com/2010/03/06/%E8%B0%83%E8%AF%95umdf%E9%A9%B1%E5%8A%A8/" \t "_blank)。

而KMDF内核模式驱动程序中的错误会损坏系统地址空间，并可能会导致计算机失败。由于这些驱动模型的特点，在构建驱动程序的开发调试环境是也会有所区别，一般在做KMDF驱动的开发调试时都选择双机或搭建虚拟机环境进行，而UMDF驱动可以进行单机调试，下面的环境搭建主要以更为复杂的KMDF模式为例。

**5、     开发调试环境配置**

**5.1     工具及平台环境**

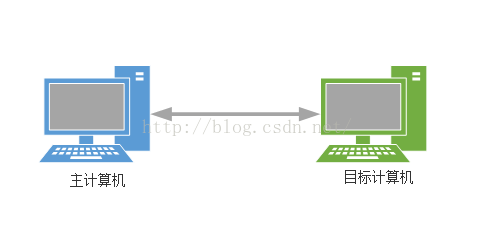
平台：Windows 10操作系统

开发环境：Visual Studio 2015 Enterprise 和Windows WDK、Windows SDK

调试环境：Oracle VM VirtualBox 5.0.14虚拟机

**5.2     方法/步骤**

**5.2.1     设置主计算机和测试计算机**

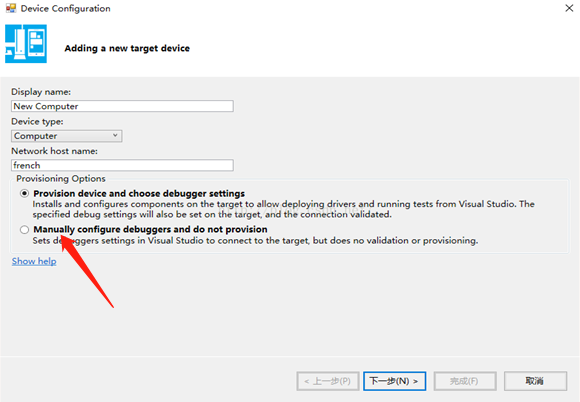


设置目标或测试计算机是配置计算机以便进行自动驱动程序部署、测试和调试的过程。一个测试和调试环境具有两台计算机：主计算机和目标计算机。目标计算机也称为“测试计算机”。使用主机上的 Vsual Studio开发和构建驱动程序。调试程序在主计算机上运行（可以使用 Visual Studio 用户界面或WinDbg调试工具），当测试和调试驱动程序时，驱动程序在目标计算机上运行。

虚拟打印机会占用一个com，设置可以windbg联通虚拟机，然后设置的时候选择手动模式，

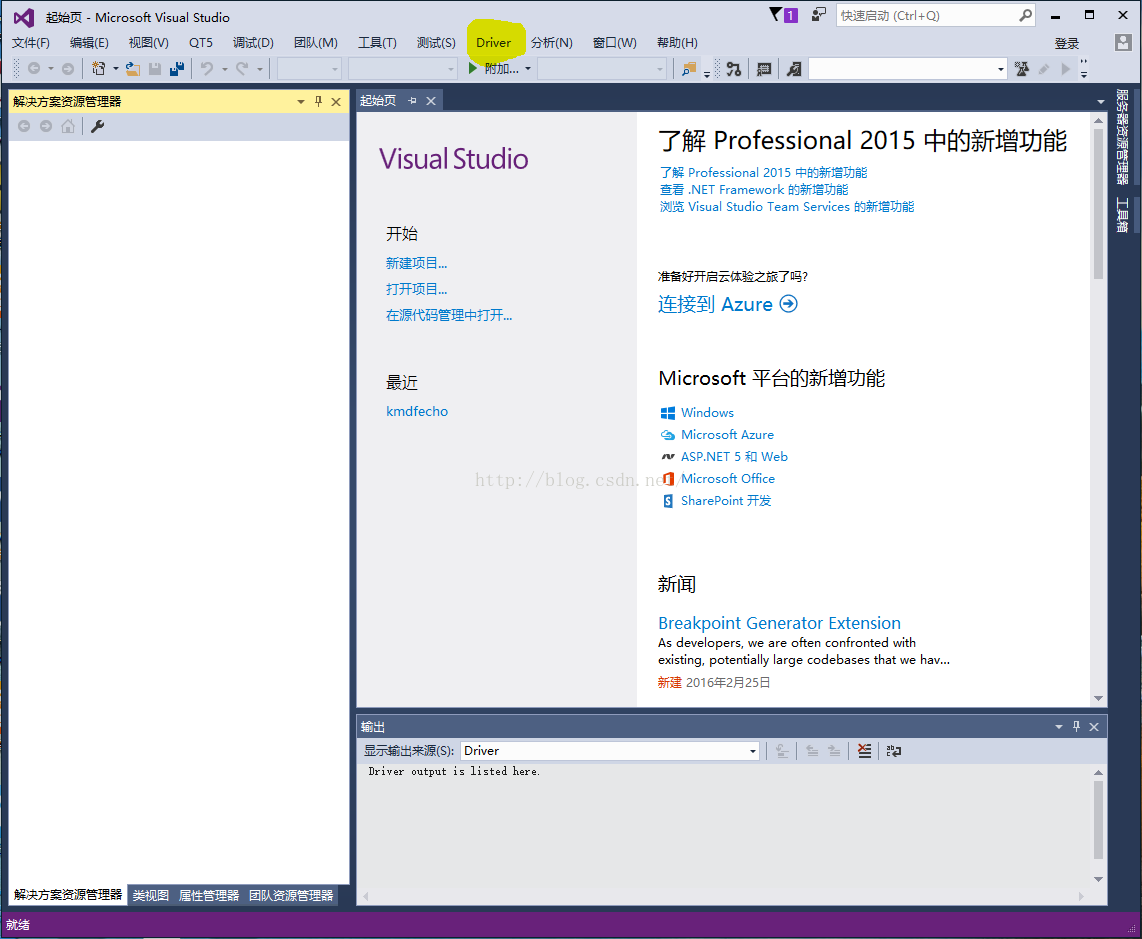
管道两个方向都要选对

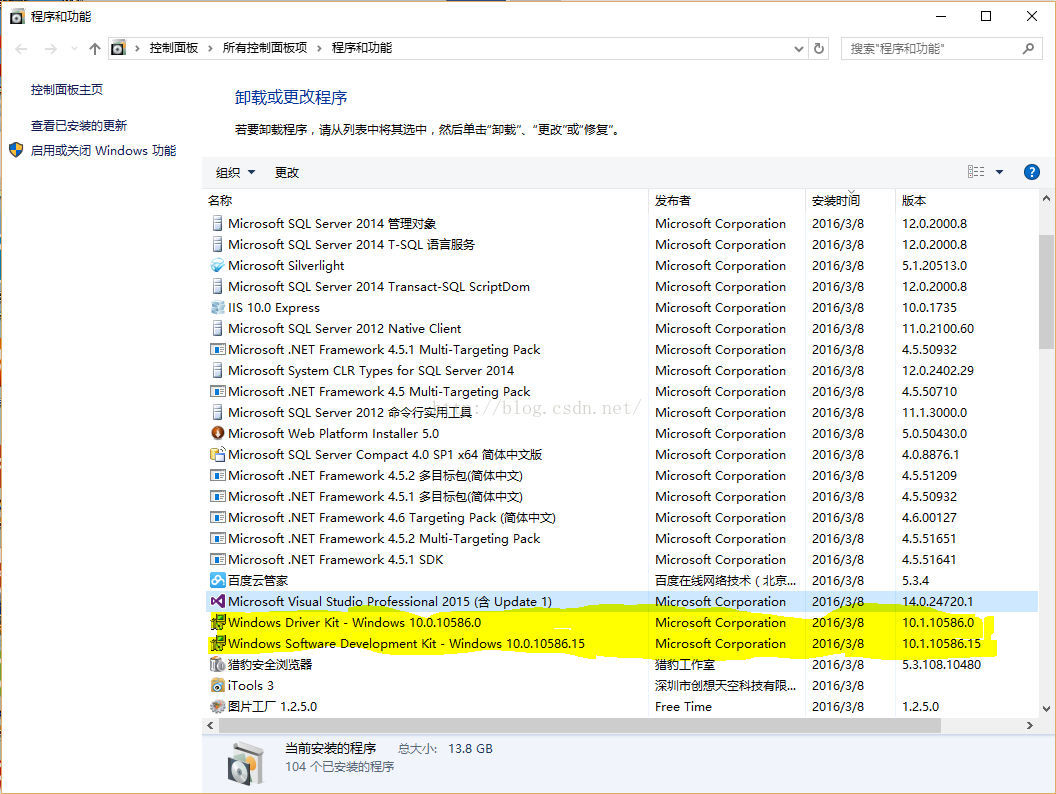




**1      安装设置主计算机**

本次搭建环境主计算机操作系统使用win10，首先安装Visual Studio 2015 Enterprise，然后安装Wdk 10。注意：WDK10一定要在vs2015后面安装，安装完后，启动vs2015在界面菜单上会出现driver菜单，如下图。另外，一般VisualStudio 2015 Enterprise是已经内带windows SDK，但是如果不是和安装的WDK一致的版本，建议在单独安装windows SDK 10，使得WDK和SDK版本一致。





**2      安装设置目标计算机**

本次测试目标计算机采用Oracle VM VirtualBox 5.0.14虚拟机。首先在主计算机上安装Oracle VM VirtualBox5.0.14虚拟机，然后在虚拟机上安装win10操作系统。

**3      联通主计算机和目标计算机**

主计算机和目标计算机必须能够彼此按名称执行 ping 操作。如果两台计算机已加入到同一工作组或同一网络域，则此操作可能更容易。

验证主机和目标计算机可以彼此执行 ping 操作。打开命令提示符窗口，并输入 ping 192.168.X.X(ip\_adress)。

如果主计算机和目标计算机已加入到一个工作组，但它们位于不同的子网上，你可能必须调整某些防火墙设置，以便主计算机和目标计算机可以通信。请按照下列步骤进行操作：

两种方法（选一即可）：

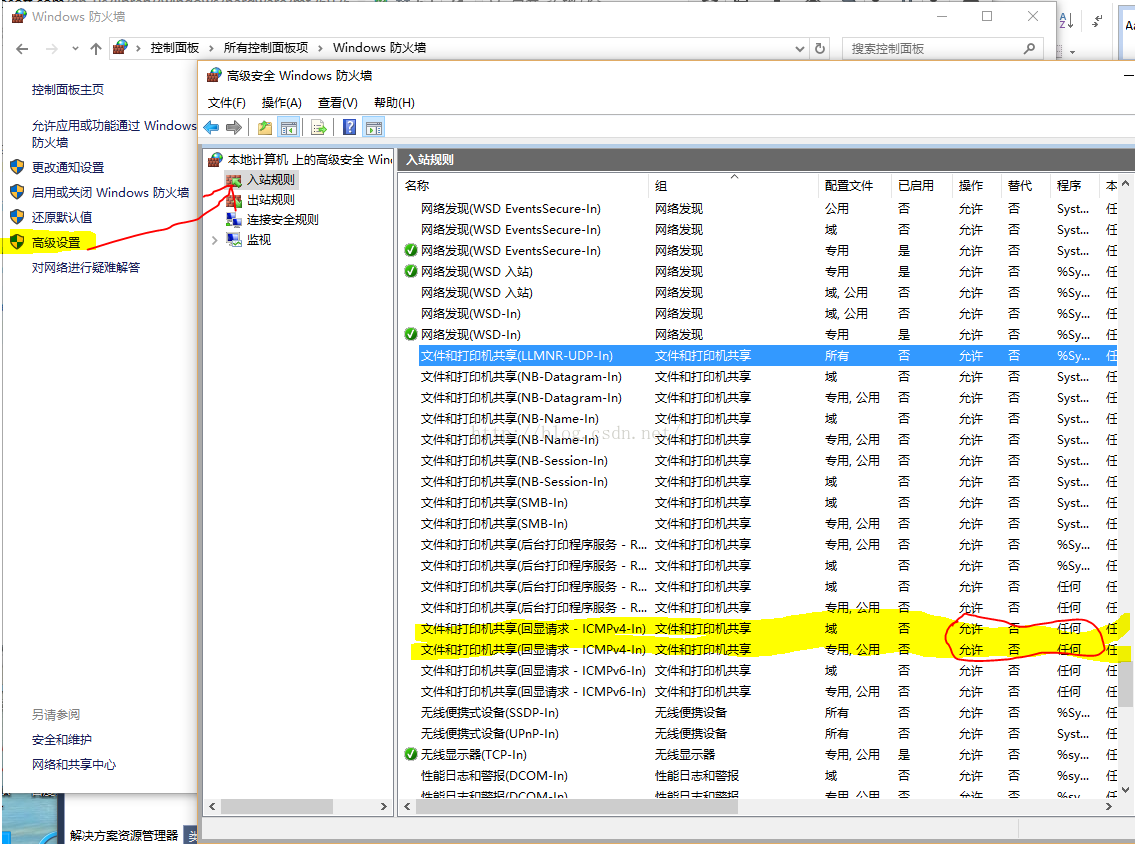
方法一：

在目标计算机上的“控制面板”中，转到“网络和 Internet”>“网络共享中心”。注意你的活动网络。它可能是“公共网络”、“私有网络”或“域”。

在目标计算机上的“控制面板”中，转到“系统和安全”>“Windows 防火墙”>“高级设置”>“入站规则”。

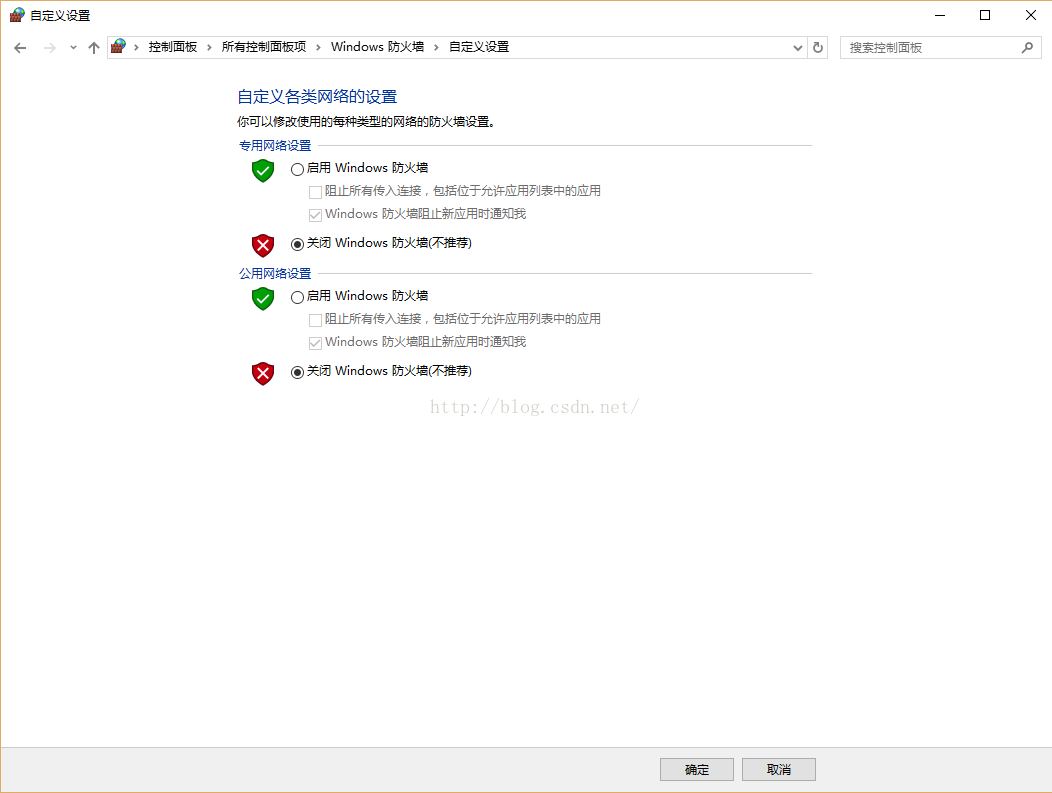
在入站规则列表中，查找用于活动网络的所有网络发现规则。（例如，查找所有“配置文件”为“专用”的网络发现规则。）双击每个规则，然后打开“作用域”选项卡。在“远程 IP 地址”下，选择“任何 IP 地址”。

在入站规则列表中，查找用于活动网络的所有“文件和打印机共享”规则。对于每个规则，双击该规则，将打开“作用域”选项卡。在“远程 IP 地址”下，选择“任何 IP 地址”。



方法二：

“控制面板”->“系统和安全”->“Windows防火墙”->"启动或关闭Windows防火墙"->"关闭Windows防火墙"



**4      目标计算机允许内核调试**

1)    在测试目标计算机上用系统管理员账户打开CMD命令窗口，输入下面命令：

C:\> bcdedit /set {default} DEBUG YES

C:\> bcdedit /set TESTSIGNING ON

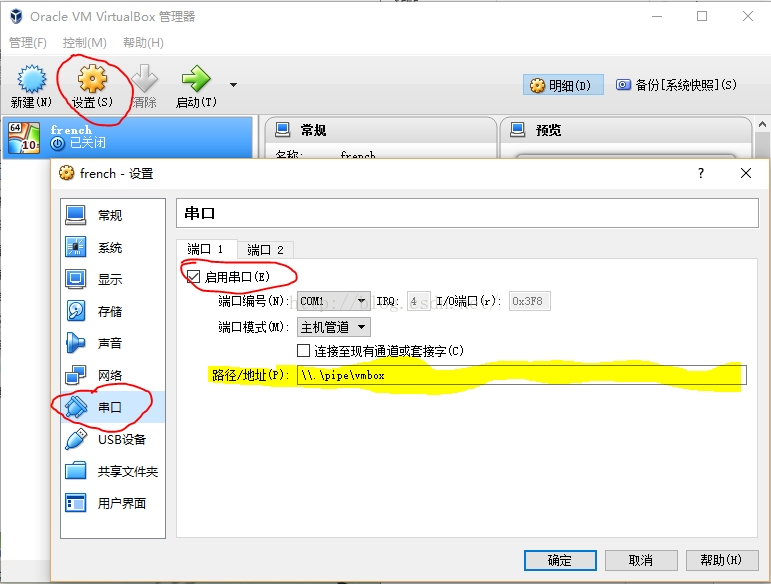
**5.2.2     设置WinDbg调试方式**

由于调试方式有好多种，我们选择常用的串口调试和网络调试两种方式进行讨论。首先我们进行串口方式的设置。

**5.2.2.1       设置测试目标计算机为串口调试方式**

**A.    在虚拟机上设置串口**

首先关闭虚拟机上的操作系统，在虚拟机上设置。对于普通的VirtualBox中的虚拟机，默认配置情况下，是没有开启对应的串口的，其配置对应的是这样的：



把串口设置成主机上的一个管道pipe（pipe是个软件上的概念）

如何设置：

* Windows系统中：名字必须为

|  |
| --- |
| ***\\.\pipe\<name>*** |

* 其中***name***自己随便起一个（当然，应该起一个，和你的虚拟机所对应的，有意义的名字）

**B.   在测试目标计算机上设置串口调试**

在测试目标计算机上用系统管理员账户打开CMD命令窗口，输入下面命令：

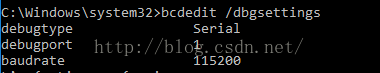
C:\> bcdedit /debug on

C:\> bcdedit /dbgsettings serial debugport:1 baudrate:115200

其中debugport:1，表示选择串口com1。

然后重启操作系统，在测试目标计算机上打开CMD命令窗口，输入下面命令，就可以看到刚才配置的参数：

C:\> bcdedit / bcdedit /dbgsettings



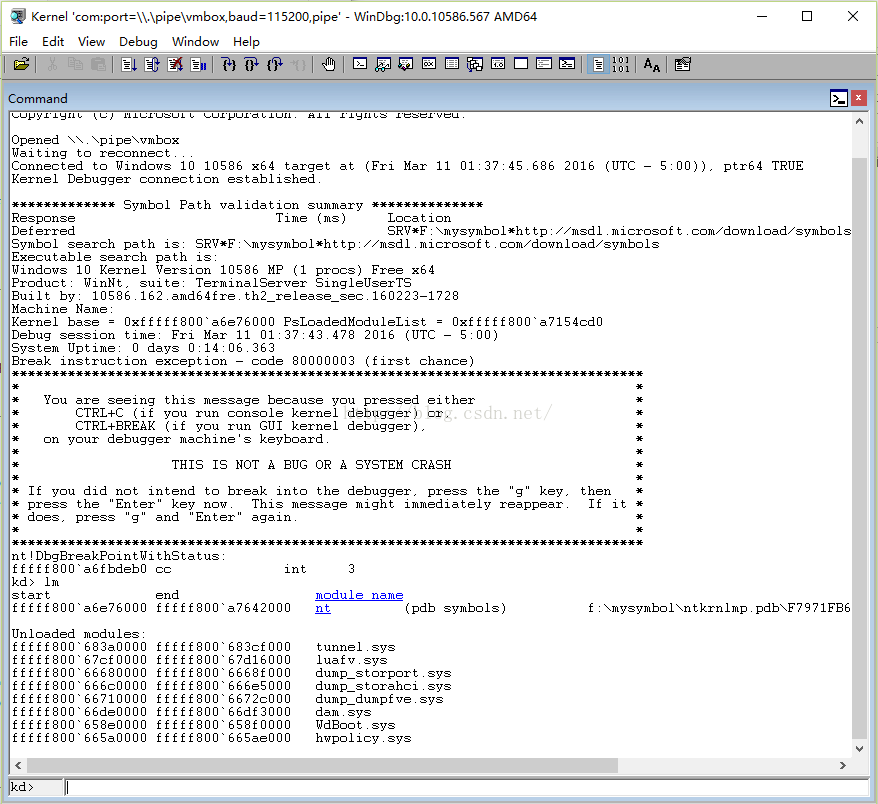
**C.   在主计算机启动WinDbg调试程序**

在主计算机上打开CMD命令窗口，进入到WinDbg程序的文件夹，一般在如下的路径：

C:\>cd C:\Program Files (x86)\Windows Kits\10\Debuggers\x64

C:\> **windbg -k com:pipe,port=\\.\pipe\vmbox,resets=0,reconnect**

Windbg程序正常运行，效果如下，说明配置正确，可以通过串口调试驱动程序了。



**5.2.2.2       设置测试目标计算机为网络调试方式**

**A.    在测试目标计算机上设置网络调试**

在测试目标计算机上用系统管理员账户打开CMD命令窗口，输入下面命令：

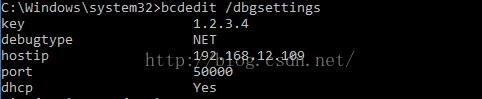
C:\> bcdedit /debug on

C:\> bcdedit /dbgsettings net hostip:192.168.12.109 port:50000 key:1.2.3.4

其中hostip:192.168.12.109，表示主计算机ip地址，port:50000制定通讯端口，key参数指定通讯使用的加密密钥key。

然后重启操作系统，在测试目标计算机上打开CMD命令窗口，输入下面命令，就可以看到刚才配置的参数：

C:\> bcdedit / bcdedit /dbgsettings



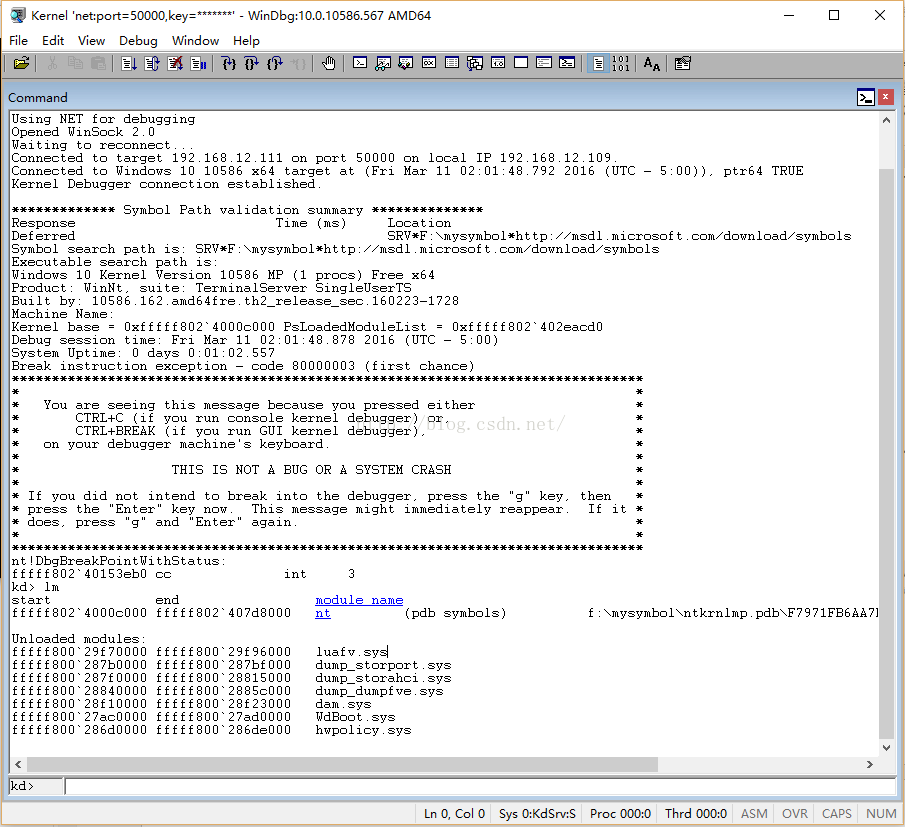
**B.   在主计算机启动WinDbg调试程序**

在主计算机上打开CMD命令窗口，进入到WinDbg程序的文件夹，一般在如下的路径：

C:\>cd C:\Program Files (x86)\Windows Kits\10\Debuggers\x64

C:\> **WinDbg –k net:port=50000,key=1.2.3.4**

Windbg程序正常运行，效果如下，说明配置正确，可以通过串口调试驱动程序了。



**5.2.3     设置Visual Studio 2015调试方式**

Visual Studio 2015本身集驱动程序开发，编译，安装，部署和调试于一身，使得Windows驱动程序开发会变得更容易。Visual Studio 2015作为调试、部署界面和使用WinDbg作为调试界面在配置上有一些不同。 

**5.2.3.1       设置测试目标计算机为串口调试方式**

**A.    在虚拟机上设置串口**

首先关闭虚拟机上的操作系统，在虚拟机上设置。对于普通的VirtualBox中的虚拟机，默认配置情况下，是没有开启对应的串口的，其配置参看上一节的串口设置。

**B.   在测试目标计算机上设置串口调试**

在测试目标计算机上用系统管理员账户打开CMD命令窗口，输入下面命令：

C:\> bcdedit /debug on

C:\> bcdedit /dbgsettings serial debugport:1 baudrate:115200

其中debugport:1，表示选择串口com1。

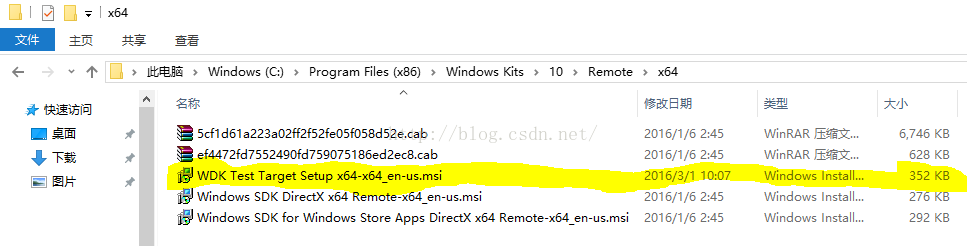
然后重启操作系统，在测试目标计算机上打开CMD命令窗口，输入下面命令，就可以看到刚才配置的参数：

C:\> bcdedit / bcdedit /dbgsettings

**C.   设置测试目标计算机为WDKRemoteUser**

在目标计算机上，运行与目标计算机平台匹配的 WDK 测试目标安装程序 MSI。可以在安装过WDK10的主计算机的WDK10安装目录下找到该 MSI。

例如：C:\Program Files (x86)\WindowsKits\10\Remote\x64\WDK Test Target Setup x64-x64\_en-us.msi

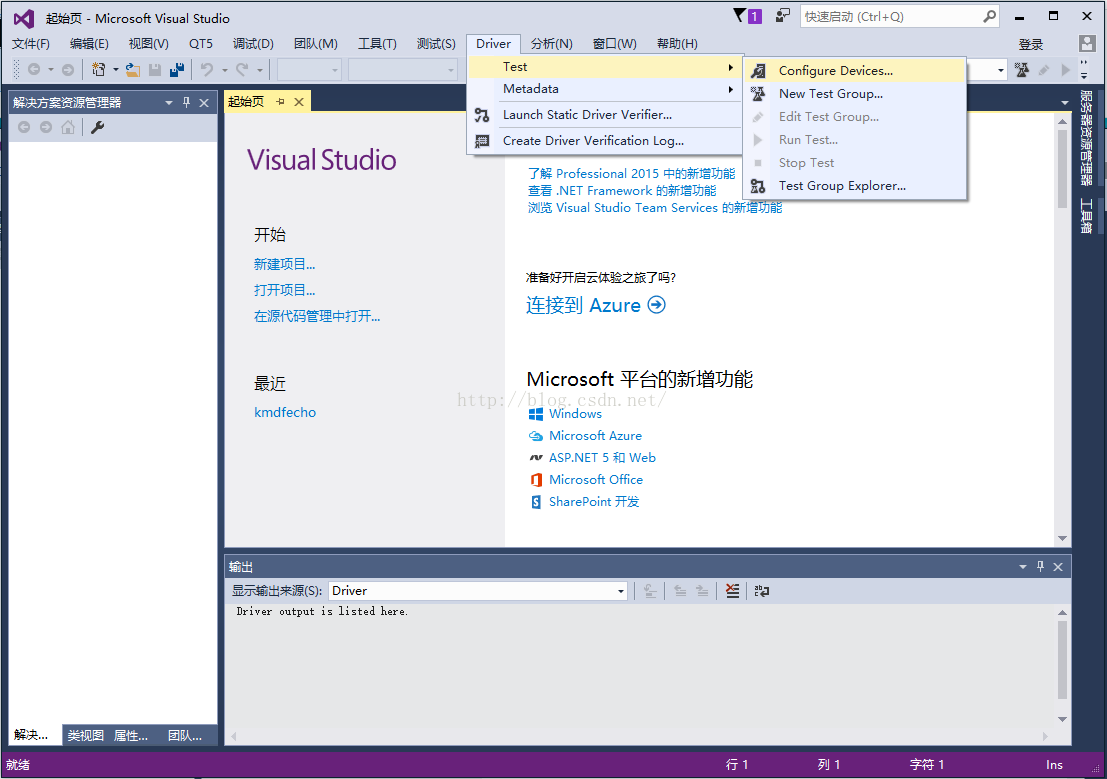


注意：如果该程序没有安装，则下一步再vs2015上面做的配置就不能成功，我就在这一步上被卡住很久，一直找不到原因。

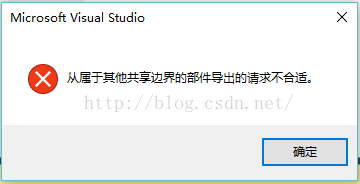
**D.   在主计算机设置**Visual Studio 2015**程序**

在主计算机上，在 Visual Studio 的“Driver”菜单中，选择“Test”>“Configure Deivce”。

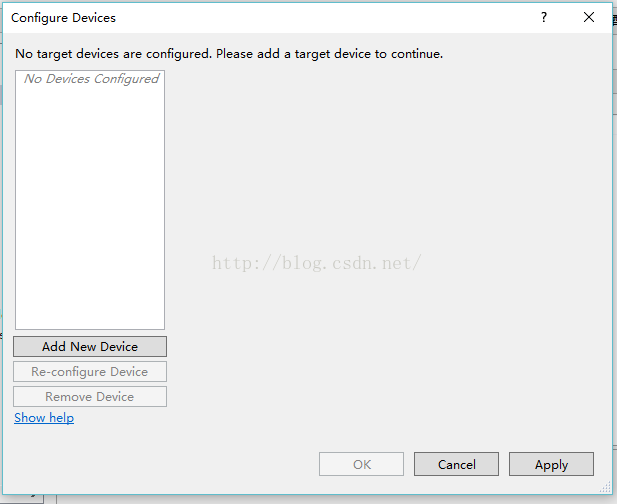
单击“Add New Device”。



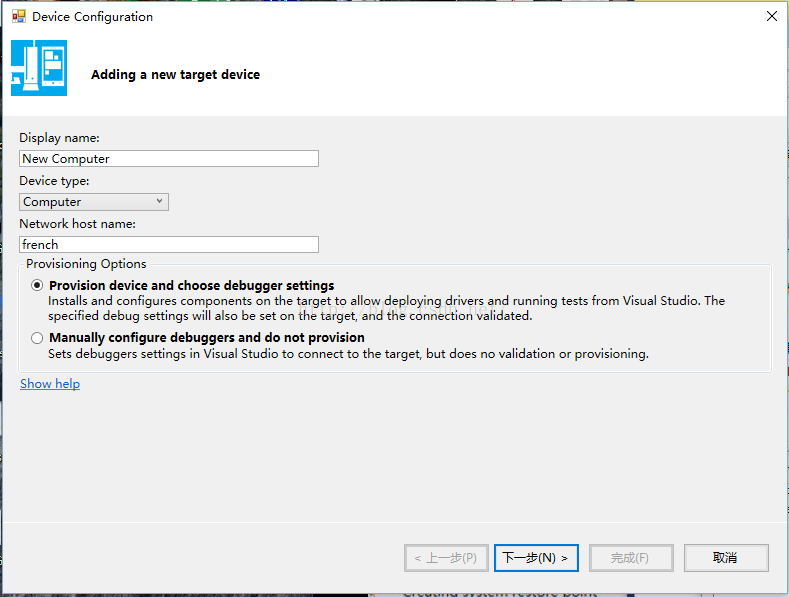
这里需要注意：如果vs2015的版本是vs2015 Update 1版本，这时候会出现一个错误提示窗口，不能进行相应的设置，提示如下：



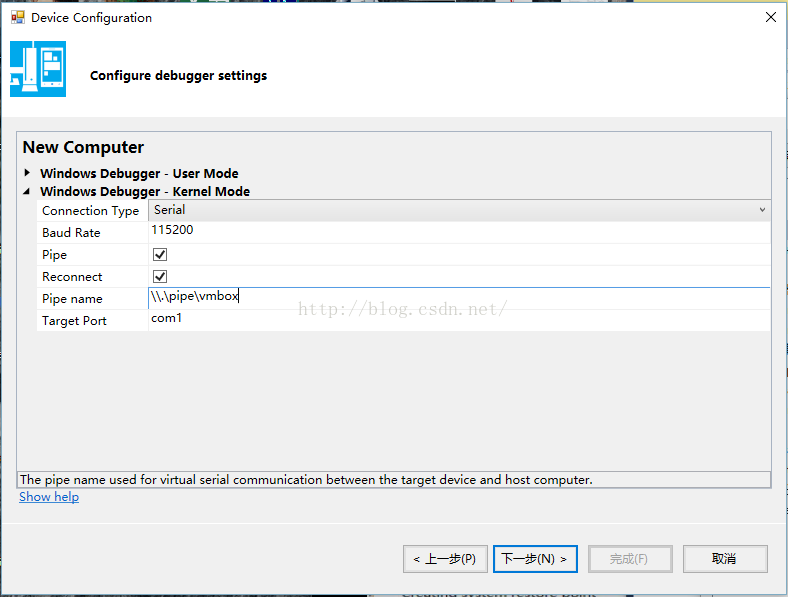
这是可以从驱动程序的项目点右键—》选择属性菜单—》选择调试选项—》在右边要启动的调试器上选择Debugging Tools for windows – Kernel Debugger—》在Remote Computer Name上点击配置，就会出现配置窗口，界面如下：



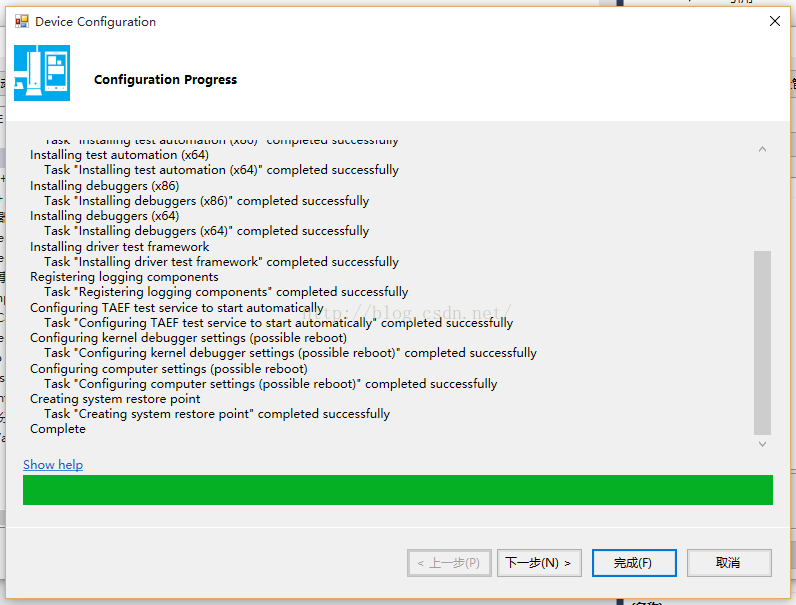
点击Add New Devide添加提个测试主机，在出现的界面上填写测试目标主机的显示名称，设备类型（计算机设备或移动设备），测试目标主机的主机名（我测试填IP地址也可以），在Provision Options选项中建议选择第一项Provision Deviceand choose debugger setting”，这样vs2015就能就行自动部署测试的驱动程序，但是设置的时候测试目标主机会自动多次重启以完成设置。选择第二项的话，测试目标主机不会重启，但是需要手工部署测试的驱动程序。



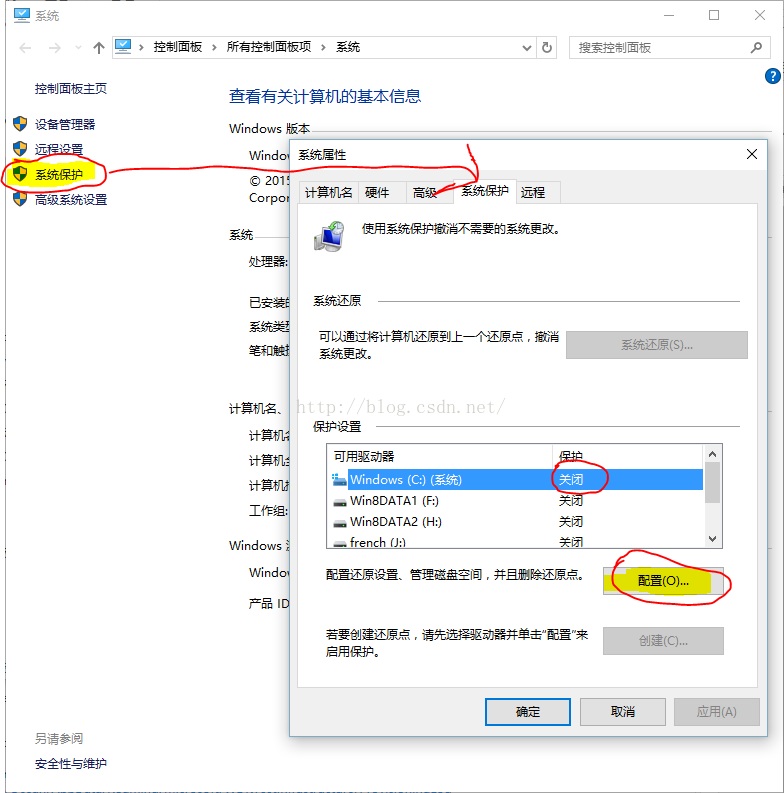
点击下一步，出现连接类型（可以选择串口/网络等），本次选择串口，然后填写在测试目标主机上设置的参数，设置界面如下：



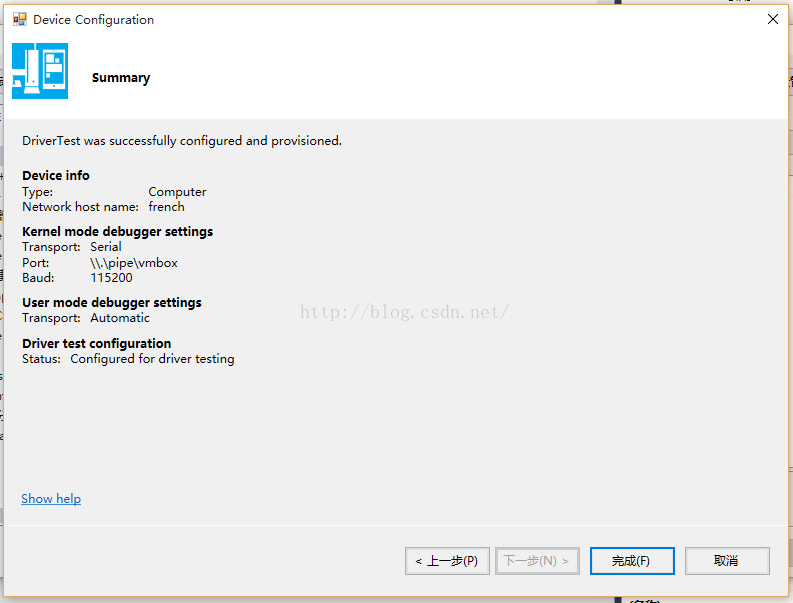
设置完成，点击下一步，则会在测试目标主机上进行相应的环境设置，期间测试目标主机会有多次重启。显示界面如下：



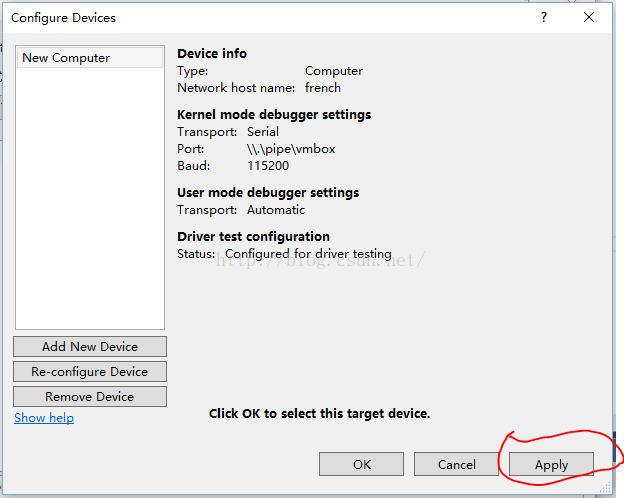
注意：如果在设置过程中在Creaing system restorepoint出现错误的话，需要在测试目标主机的C盘上打开系统还原点。设置的步骤为：开始右键—》系统—》系统保护—》配置—》启动并设置C盘还原点。



设置完成后显示如下界面，提示配置成功：



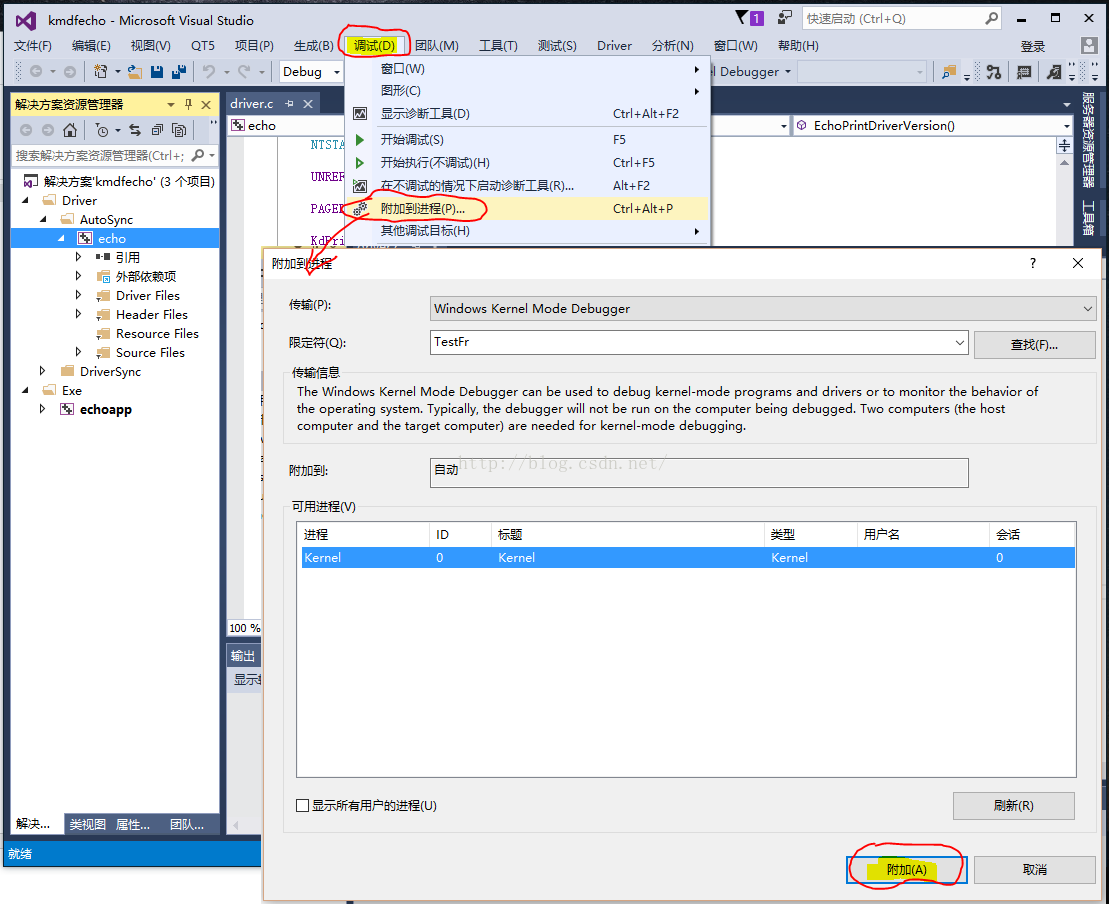
点击完成，显示如下界面：



点击Apply应用，然后选择OK。

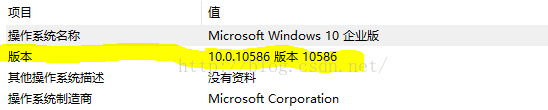
**E.    在主计算机Visual Studio 2015程序测试配置**

最后在主计算机的VS2015的调试菜单中—》选择附加到进程--》在传输（P）下拉框中选择Windows Kernel ModelDebugger，在限定符（Q）下拉框中选择刚才配置的测试目标主机名称—》在可用进程中选择Kernel—》最后点击附加按钮。



点击附加按钮后出现调试界面表示配置成功，可以进行驱动调试了。

注意：这里如果点击附加按钮后Vs2015出现崩溃现象请检查Win10的版本，如果win10的版本还是10240版本的话请升级到10586版本。可以点击VS2015的帮助菜单—》关于Microsoft Visual Studio—》系统信息可以查看到系统的版本。本人在这里折腾了将近两天的时间，最后才找到是由于版本不匹配的问题。

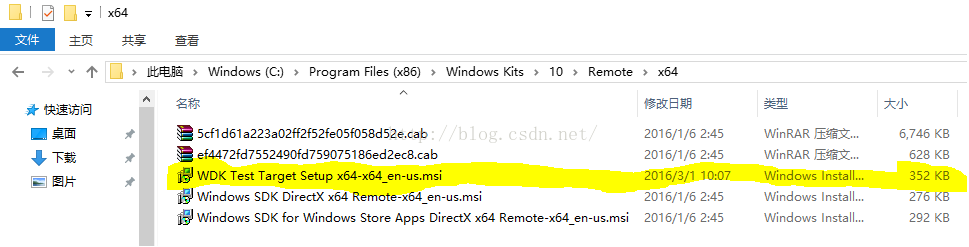


**5.2.3.2       设置测试目标计算机为网络调试方式**

**A.    设置测试目标计算机为WDKRemoteUser**

在目标计算机上，运行与目标计算机平台匹配的 WDK 测试目标安装程序 MSI。可以在安装过WDK10的主计算机的WDK10安装目录下找到该 MSI。

例如：C:\Program Files (x86)\WindowsKits\10\Remote\x64\WDK Test Target Setup x64-x64\_en-us.msi

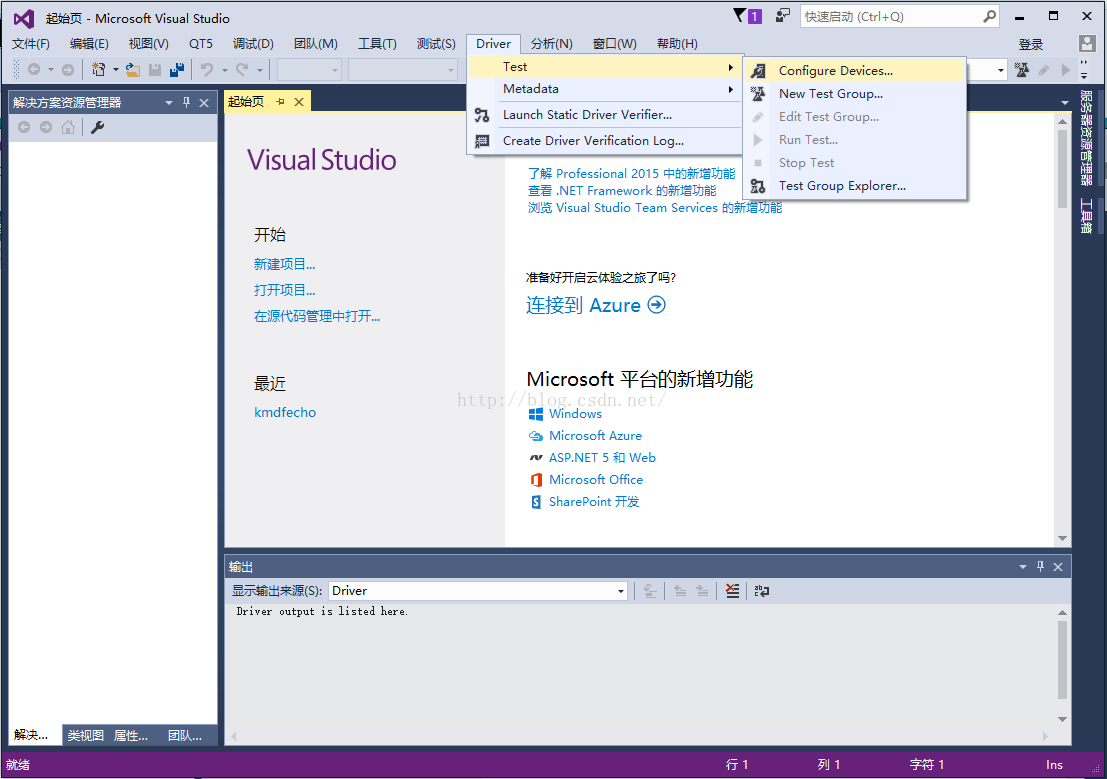


注意：如果该程序没有安装，则下一步再vs2015上面做的配置就不能成功，我就在这一步上被卡住很久，一直找不到原因。

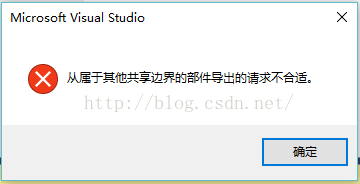
**B.   在主计算机设置**Visual Studio 2015**程序**

在主计算机上，在 Visual Studio 的“Driver”菜单中，选择“Test”>“Configure Deivce”。

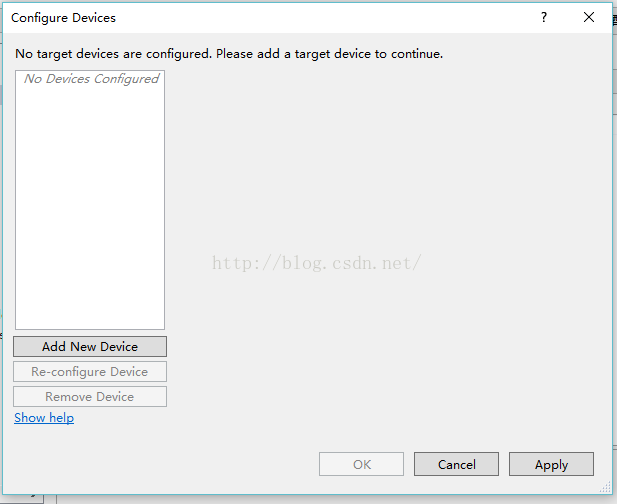
单击“Add New Device”。



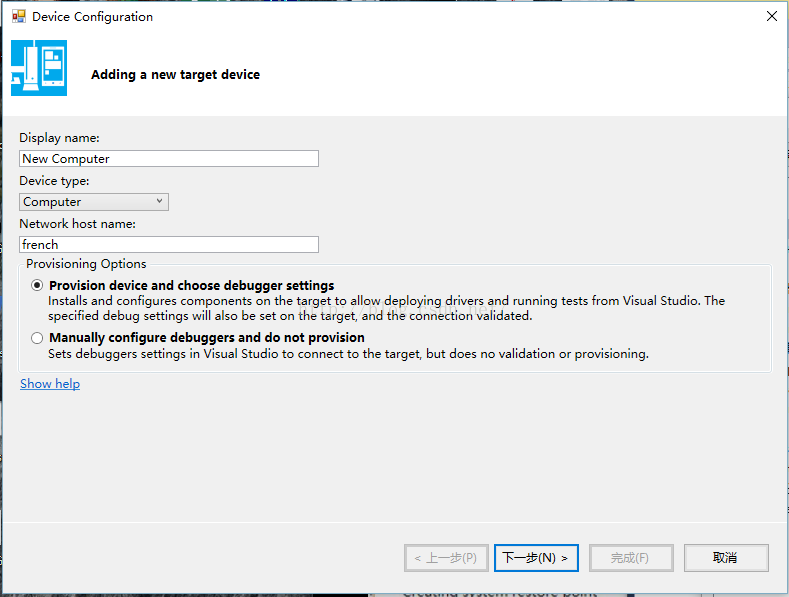
这里需要注意：如果vs2015的版本是vs2015 Update 1版本，这时候会出现一个错误提示窗口，不能进行相应的设置，提示如下：



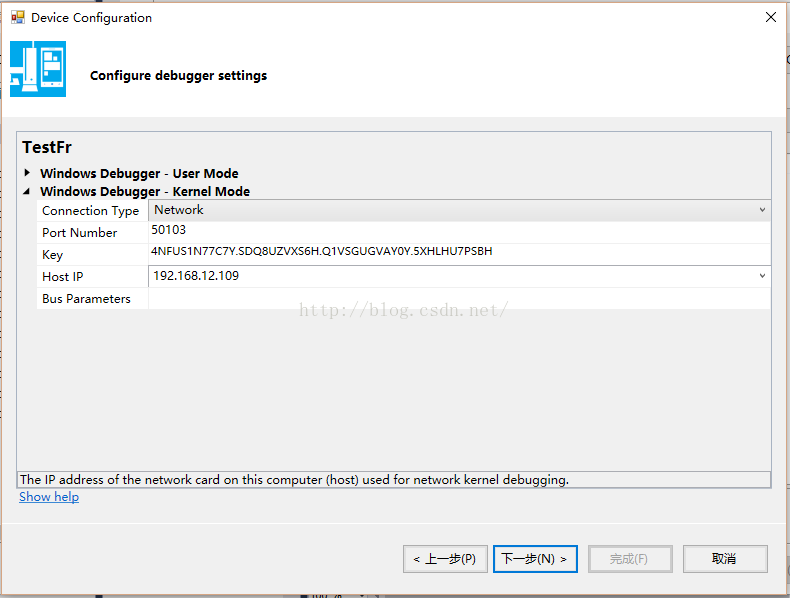
这是可以从驱动程序的项目点右键—》选择属性菜单—》选择调试选项—》在右边要启动的调试器上选择Debugging Tools for windows – Kernel Debugger—》在Remote Computer Name上点击配置，就会出现配置窗口，界面如下：



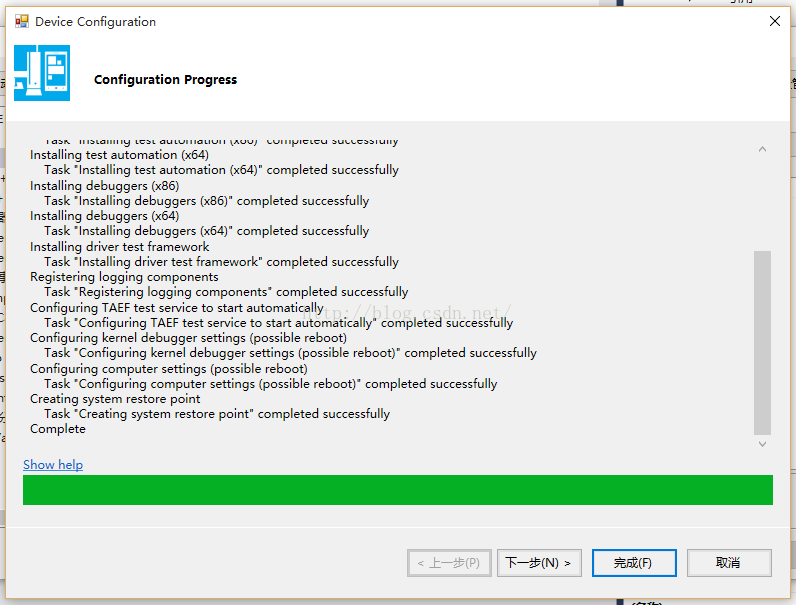
点击Add New Devide添加提个测试主机，在出现的界面上填写测试目标主机的显示名称，设备类型（计算机设备或移动设备），测试目标主机的主机名（我测试填IP地址也可以），在Provision Options选项中建议选择第一项Provision Deviceand choose debugger setting”，这样vs2015就能就行自动部署测试的驱动程序，但是设置的时候测试目标主机会自动多次重启以完成设置。选择第二项的话，测试目标主机不会重启，但是需要手工部署测试的驱动程序。



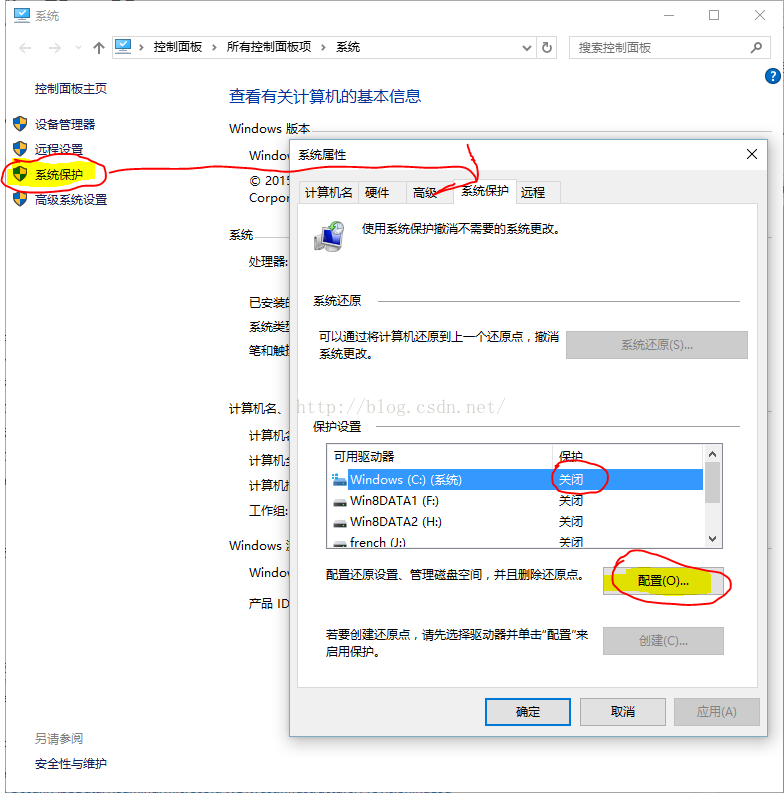
点击下一步，出现连接类型（可以选择串口/网络等），本次选择网络，然后填写在测试目标主机上设置的参数，设置界面如下：



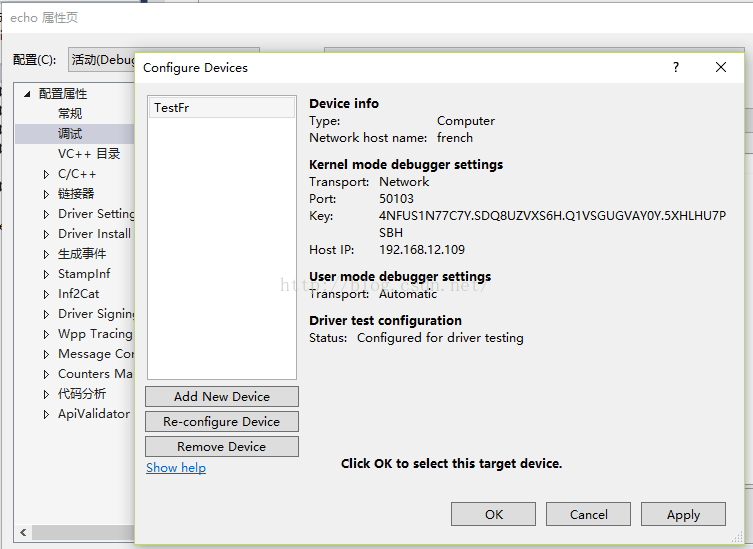
设置完成，点击下一步，则会在测试目标主机上进行相应的环境设置，期间测试目标主机会有多次重启。显示界面如下：



注意：如果在设置过程中在Creaing system restorepoint出现错误的话，需要在测试目标主机的C盘上打开系统还原点。设置的步骤为：开始右键—》系统—》系统保护—》配置—》启动并设置C盘还原点。



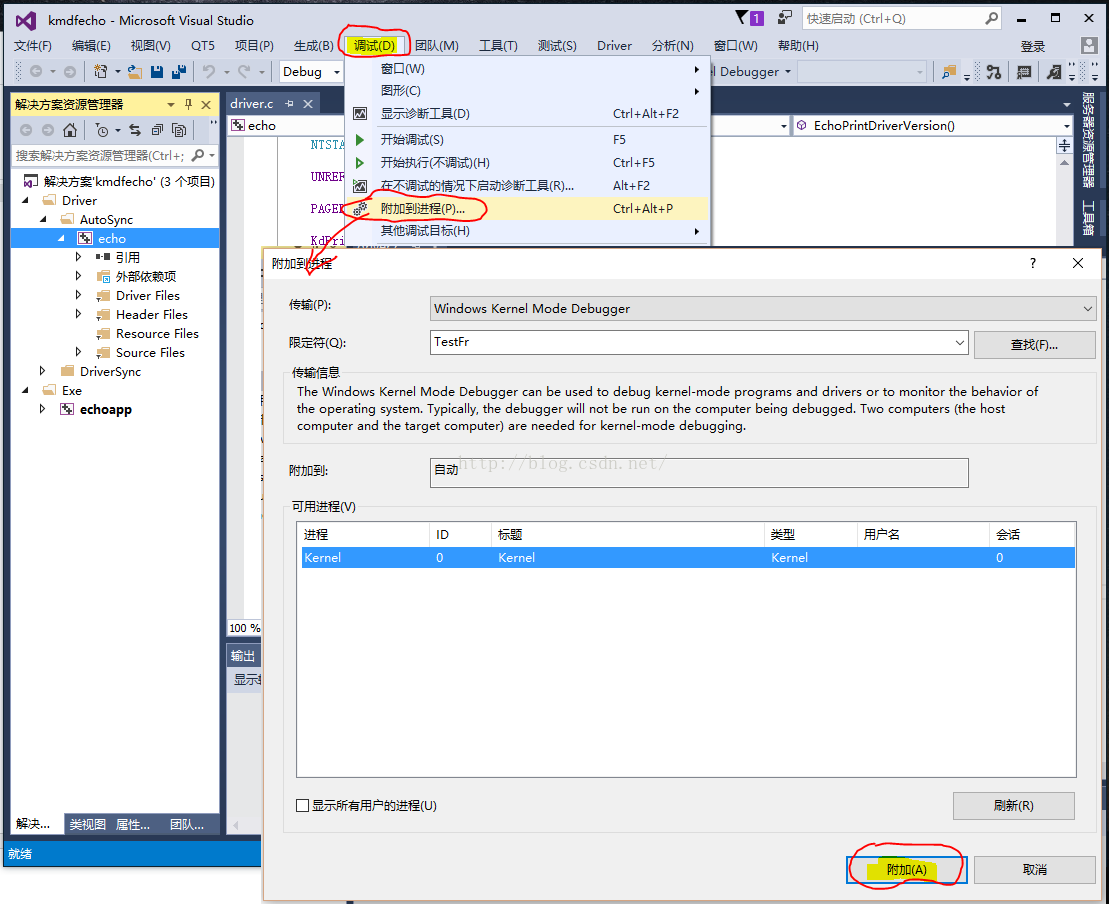
设置完成后显示如下界面，提示配置成功：



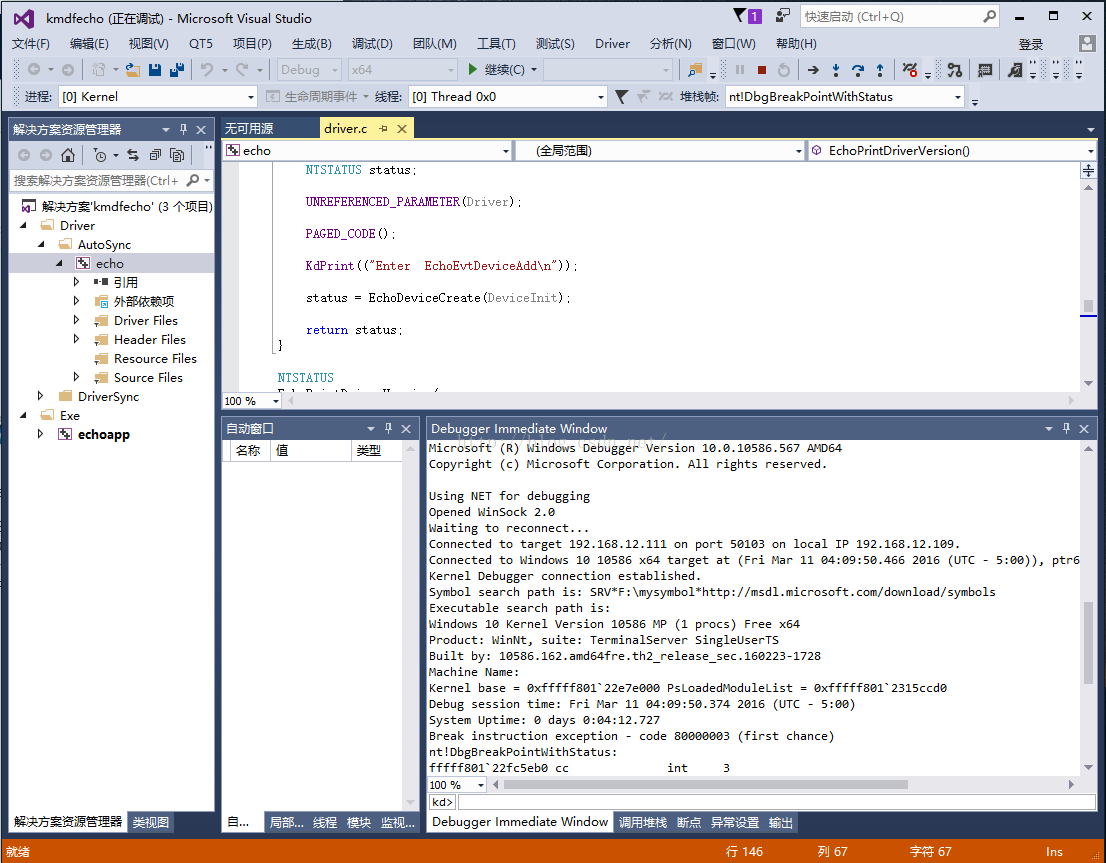
点击Apply应用，然后选择OK。

**C.   在主计算机Visual Studio 2015程序测试配置**

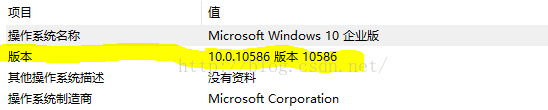
最后在主计算机的VS2015的调试菜单中—》选择附加到进程--》在传输（P）下拉框中选择Windows Kernel ModelDebugger，在限定符（Q）下拉框中选择刚才配置的测试目标主机名称—》在可用进程中选择Kernel—》最后点击附加按钮。



点击附加按钮后出现调试界面表示配置成功，可以进行驱动调试了。



注意：这里如果点击附加按钮后Vs2015出现崩溃现象请检查Win10的版本，如果win10的版本还是10240版本的话请升级到10586版本。可以点击VS2015的帮助菜单—》关于Microsoft Visual Studio—》系统信息可以查看到系统的版本。本人在这里折腾了将近两天的时间，最后才找到是由于版本不匹配的问题。

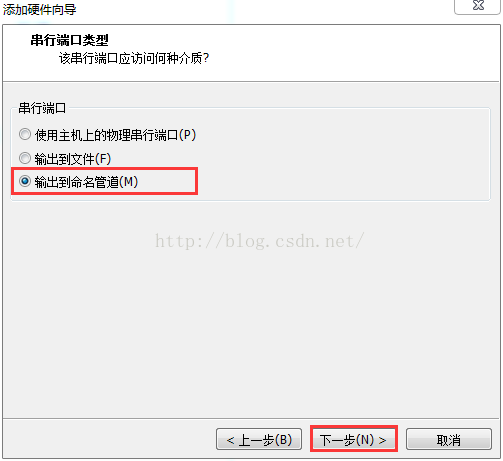


**二、其它方式参考**

工具安装就不说了，不会的百度一下。只说一下一些关键的地方。

**1、安装好虚拟机后，给虚拟机增加一个串口设备。具体操作见图：**







上面的\\.\pipe\com\_1可以改\\.\pipe\com\_2，我就不改了后面名字可以随便取的，完成配置后虚拟机设备配置就多了一个串行端口的设备。因为打印机用了端口1。所以设备里面显示的是串行端口2，重新启动虚拟机系统后，会发现设备管理里面多了一个COM2的串口设备（如果想设置成端口1，把打印机设备删了在添加串口设备）。

**2、配置虚拟机系统为调试模式**

     以管理员身份，在虚拟机命令行模式下运行如下2个命令开启。

      bcdedit /debug on

      bcdedit /dbgsettings serial debugport:*n* baudrate:115200

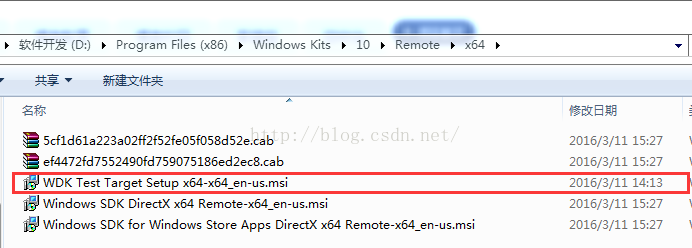
      where *n* is the number of a COM port on the virtual machine.（n时虚拟机的一个COM端口号。就是设备显示的串行端口2，n就是端口2）。

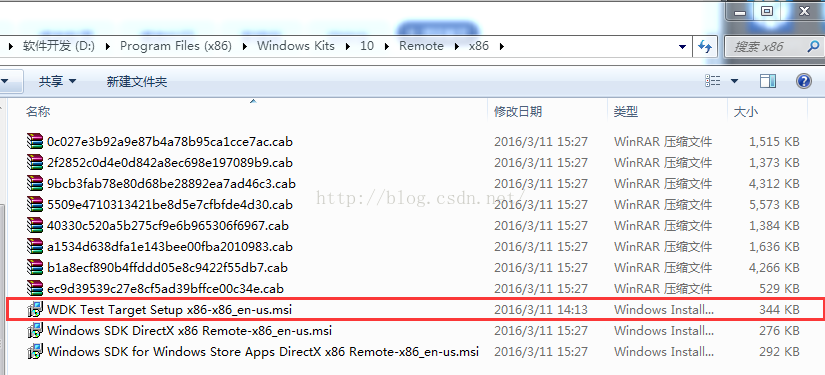
       我们使用下面的语句即可：

**bcdedit /debug on**

**bcdedit /dbgsettings serial debugport:***2***baudrate:115200**

**3、把WDK10的虚拟机调试驱动拷贝到虚拟机中安装。**

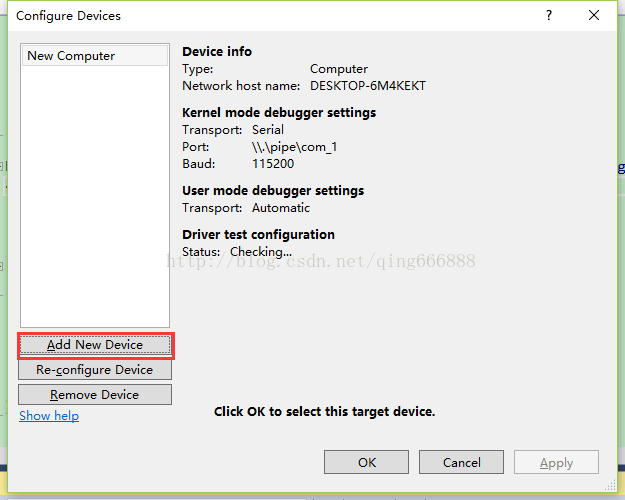
****

****

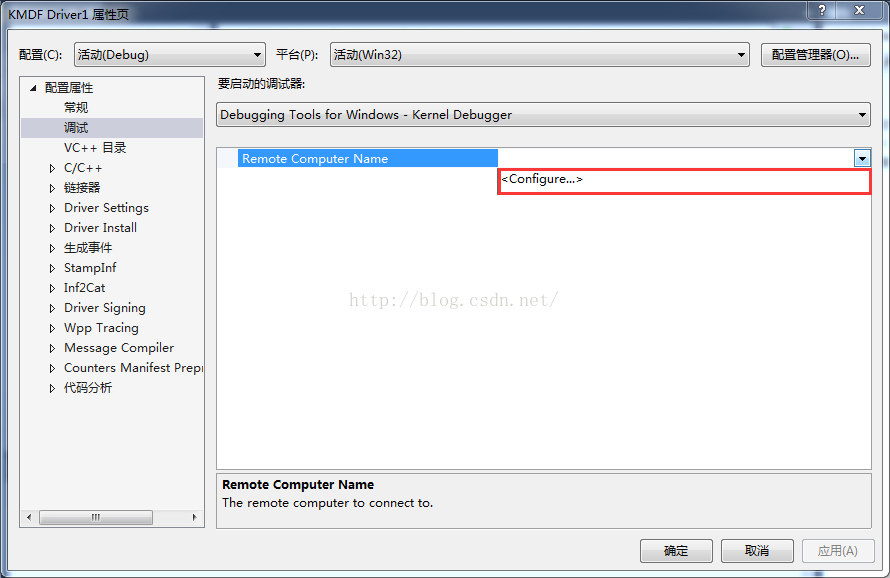
因为VS2015可以调试x86 x64两种驱动，根据被调试系统选择不同的文件拷贝到虚拟机安装，我的是64位的选择x64

**4、VS2015开发工具配置**

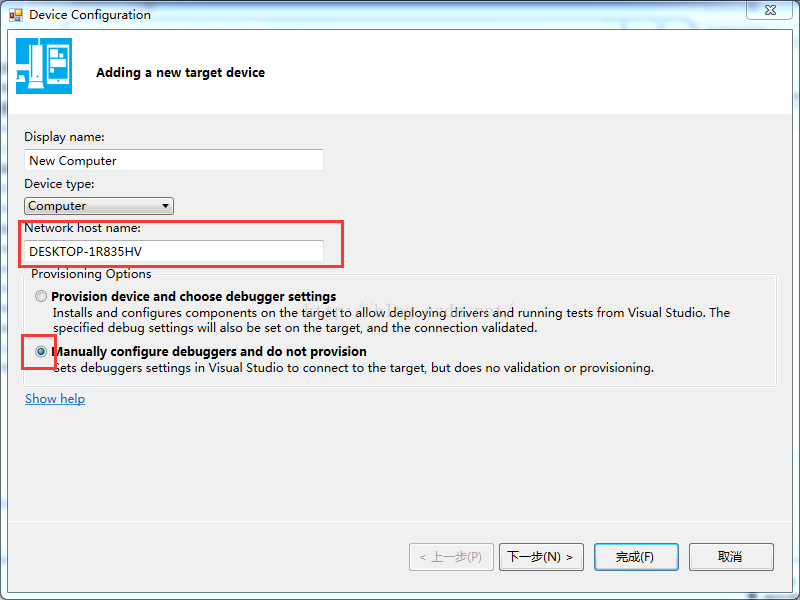
在VS2015的菜单栏点击DRIVER--TEST--CONFIGURE DEVICES,选择增加新设备，如下图：



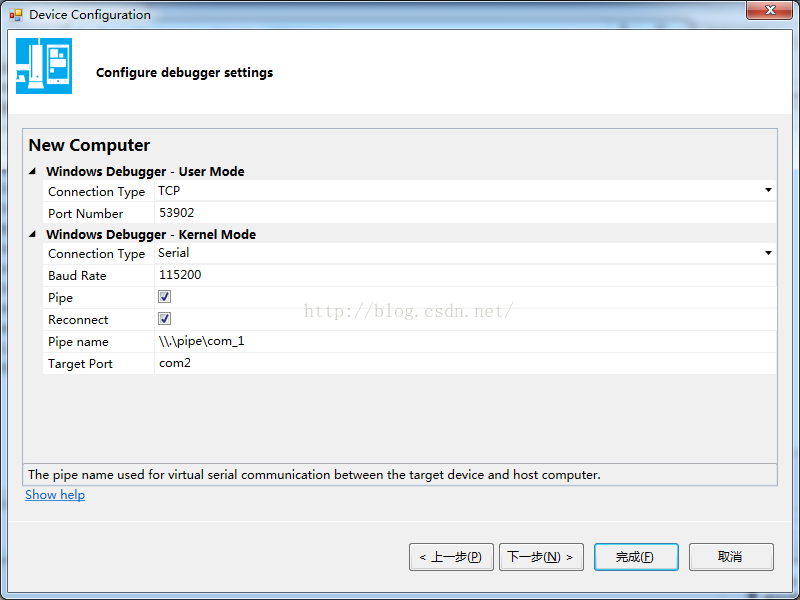
或者换个地方配置，从工程属性里面进行配置。右键工程属性配置从这也可以进行测试设备的配置：

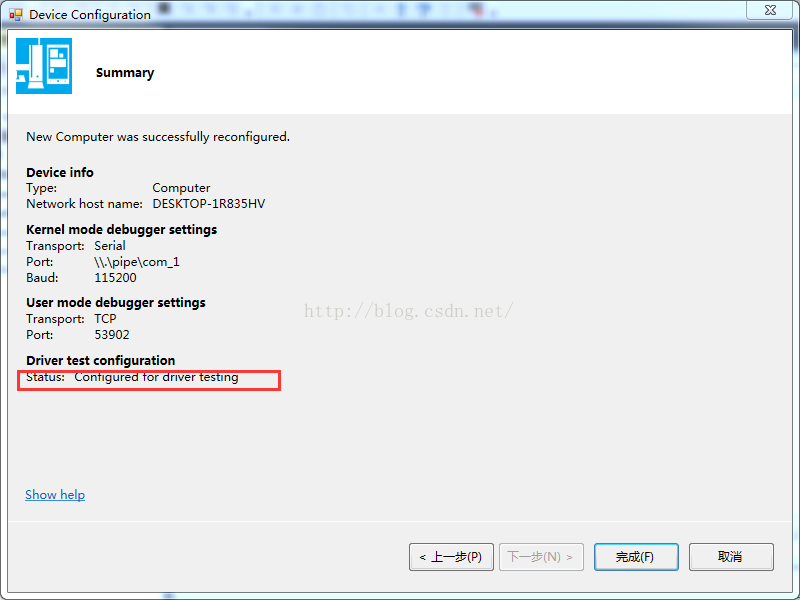


配置界面，下面两个选项第一项由VS2015帮我们创建一个WDKuser测试账户，自动打开测试模式等。第二个是使用我们自己配置的参数。由于上面我们已经配置了串口参数 、设置了调试模式所以我们选择第二个就行了。



如下配置在内核模式中，选择串口、配置好波特率、勾上2个选项、管道名称、端口号（我们上面配置的COM2）。

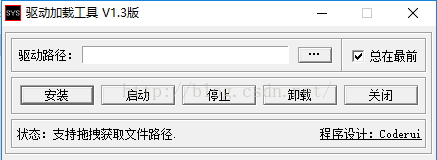




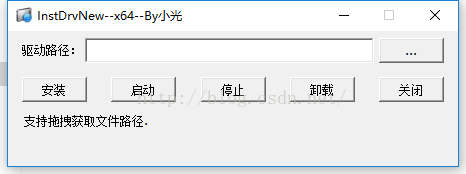
出现上述红色框字样则说明配置已经成功了。（不要开启全局代理软件，否则会出现连接不上的问题）。

**5、使用工具安装调试64位驱动**

网上下的下面的驱动安装软件貌似64位的都用不了。



自己动手丰衣足食大笑，自己在网上弄了个源码改了下弄成可以加载64位的驱动的版本了。

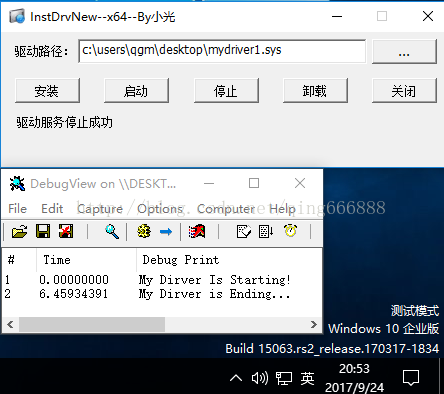


1）使用DebugView打印内核调试信息是开发驱动的非常重要的手段，但DebugView默认在WINDOWS 10下却无法获取内核的调试日志，驱动调用KdPint/DbgPrint等的打印结果是无法被DebugView给捕捉到的。设置方法如下（复制下面的代码到txt文件。改后缀为.reg后执行（看清楚了是.reg不是.bat敲打），然后重启电脑）：

**[cpp]** [view plain](https://blog.csdn.net/qing666888/article/details/50858272) [copy](https://blog.csdn.net/qing666888/article/details/50858272)

1. Windows Registry Editor Version 5.00
3. [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Debug Print Filter]
4. "DEFAULT"=dword:0000000f

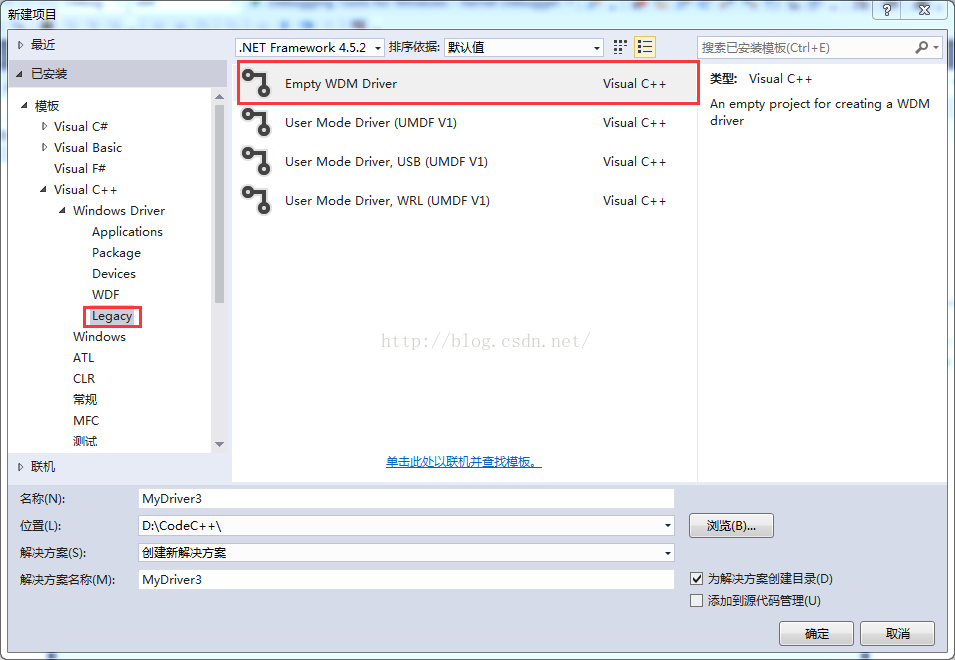
2）还有自己编译出来的驱动要使用编译的Debug版本来测试，使用KdPrint打印需要显示的调试信息。下面是在Win10 64位的虚拟机中用此安装程序，加载64位的驱动以及用DebugView查看驱动打印消息的截图结果。DebugView中需要勾选Capture Kernel选项。



**6、在VS2015中下断点进行调试**

如果你第五部已经完成了，并且在DebugView中已经能够输出调试的打印信息了，辣么骚年你已经离成功只有一部之遥了大笑。

创建一个驱动工程，WDK10提供的模板中根本没有提供NT驱动模板，我们如何创建NT驱动呢?  
解决：其实虽然没有提供NT模板，但是我们可以建立WDM空模板工程，然后再自己添加文件，编译，得到的也就是NT驱动了。



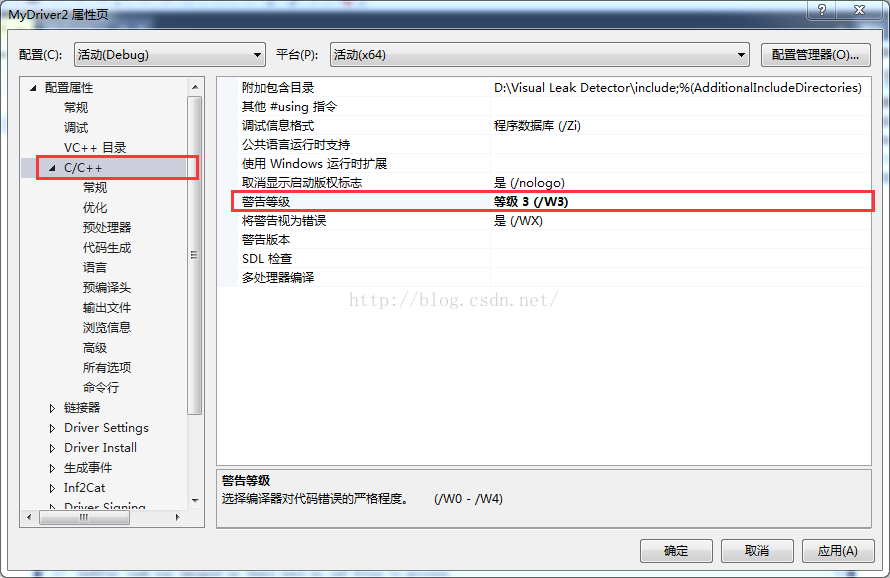
驱动工程中会帮你建立一个inf文件，NT是使用不到的，可以直接删除。我们直接添加一个MyDriver.c

在文件中添加测试代码。

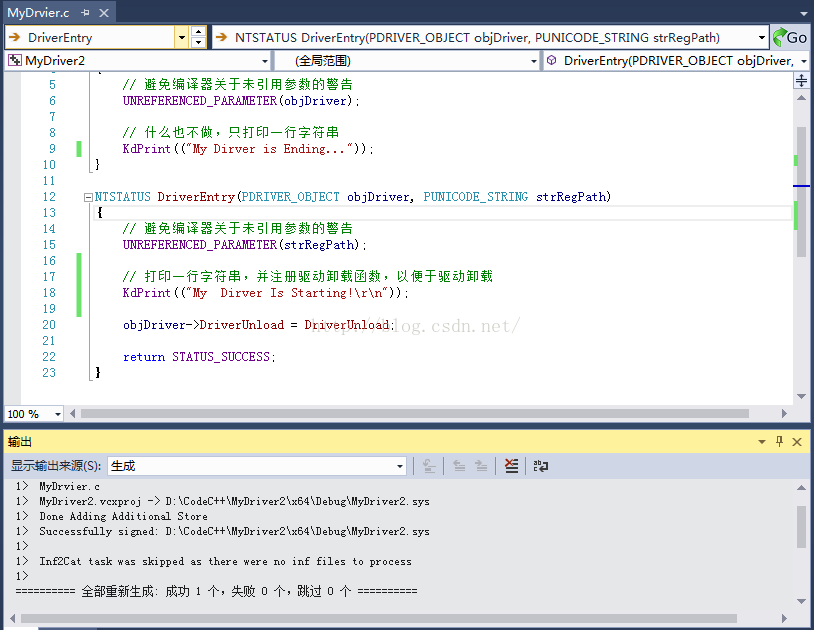
**[cpp]** [view plain](https://blog.csdn.net/qing666888/article/details/50858272) [copy](https://blog.csdn.net/qing666888/article/details/50858272)

1. #include <ntifs.h>
3. **VOID** DriverUnload(PDRIVER\_OBJECT objDriver)
4. {
5. // 避免编译器关于未引用参数的警告
6. UNREFERENCED\_PARAMETER(objDriver);
8. // 什么也不做，只打印一行字符串
9. KdPrint(("My Dirver is Ending..."));
10. }
12. NTSTATUS DriverEntry(PDRIVER\_OBJECT objDriver, PUNICODE\_STRING strRegPath)
13. {
14. // 避免编译器关于未引用参数的警告
15. UNREFERENCED\_PARAMETER(strRegPath);
17. // 打印一行字符串，并注册驱动卸载函数，以便于驱动卸载
18. KdPrint(("My  Dirver Is Starting!\r\n"));
20. objDriver->DriverUnload = DriverUnload;
22. **return** STATUS\_SUCCESS;
23. }

编译，报错,没有关系，这些都是因为安全警告等级太高了，我们可以**降低编译器警告等级的**方式解决：



执行编译，编译的时候选择Debug模式，选择X64的选项（因为我们的虚拟机使用的是64位的系统，不能够运行x86的驱动，如果你编译出来的是x86的驱动，用上的工具安装是可以，但是启动失败提示阻止运行之类的）。编译成功后，按照上面的第五步进行安装测试。

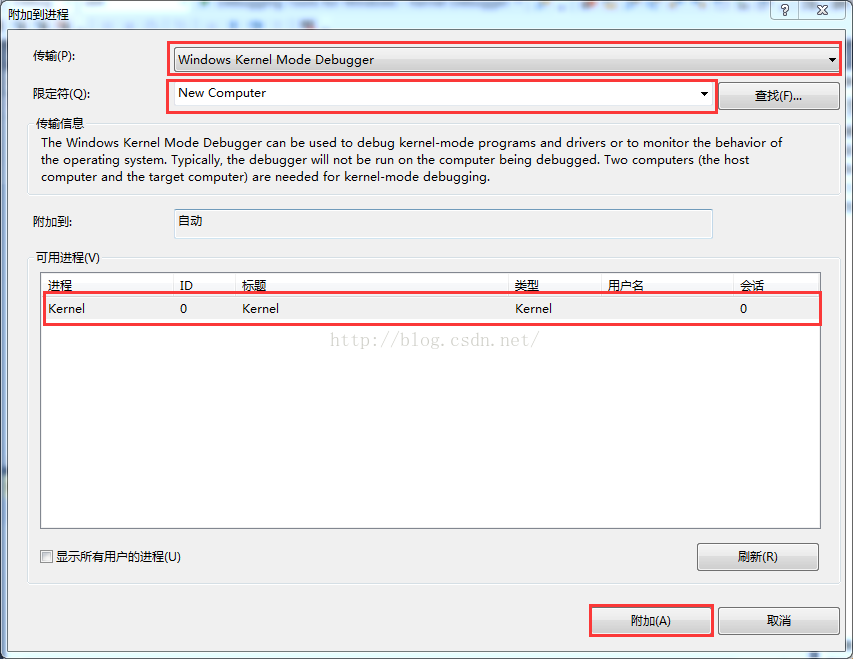


不知不觉感觉已经写了好长了。。。。奋斗抽口烟压压精。好吧我不抽烟的。。。。。继续。如果上述的驱动成功编译出来了。辣么现在我们开始附加虚拟机的内核进行调试了。

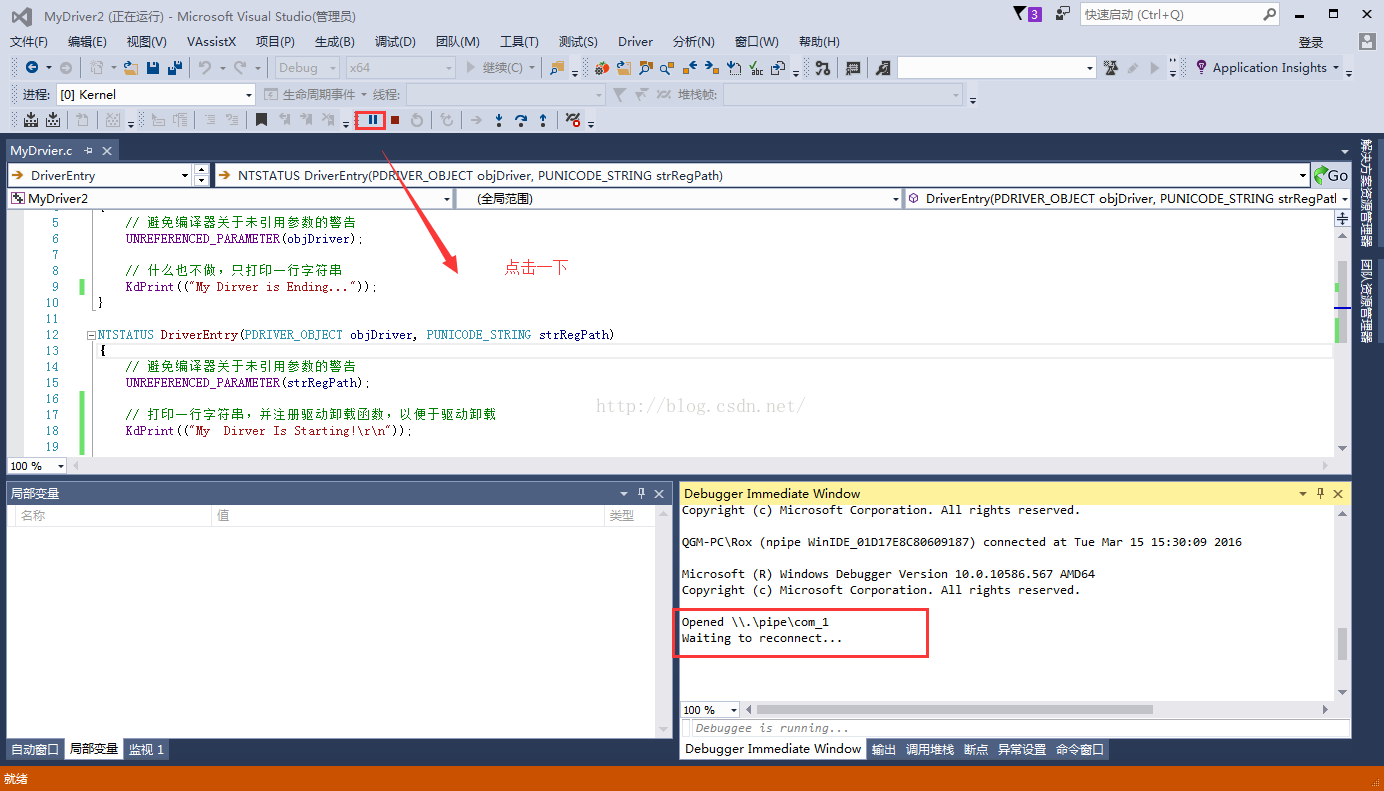
首先把虚拟机的系统内核附加到VS2015，具体步骤见下图：



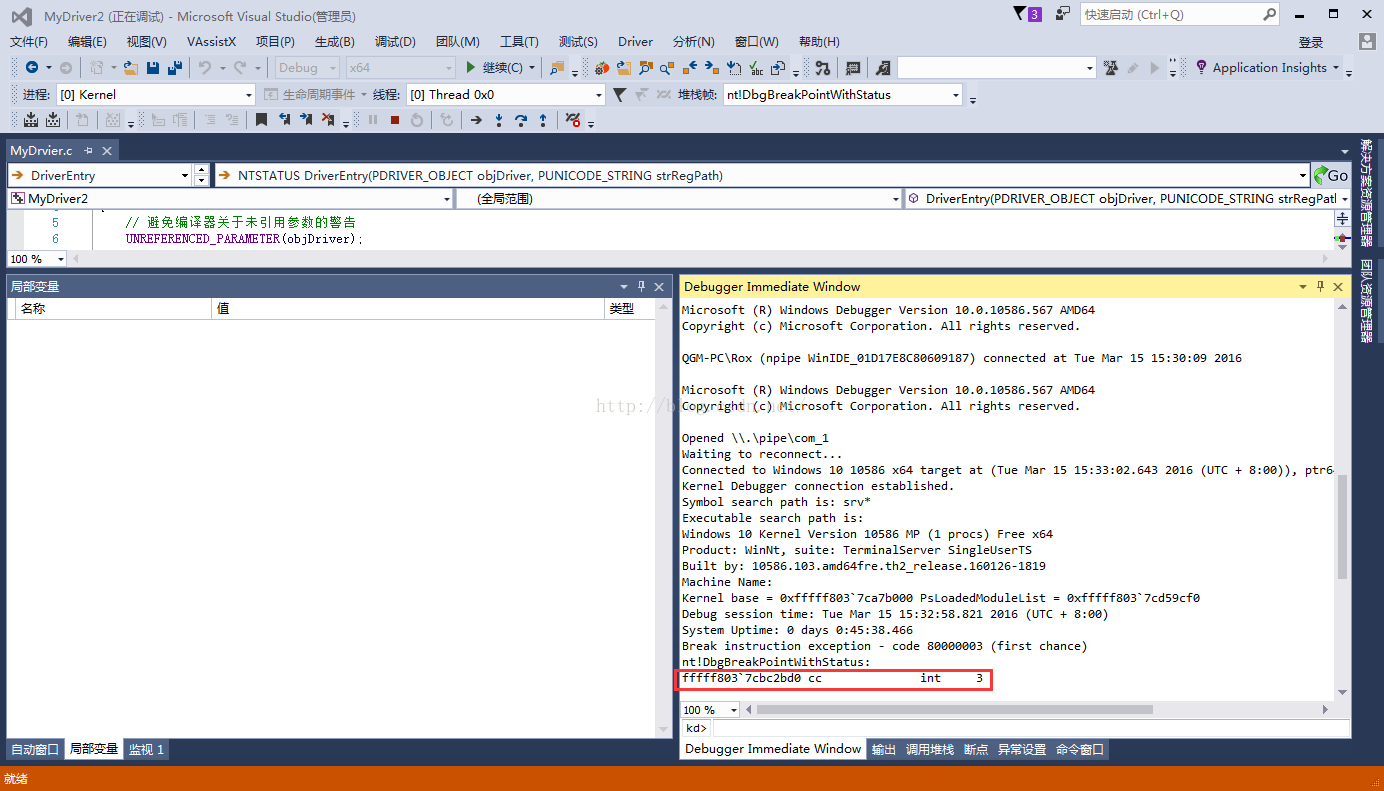
选中内核调试模式、选自己配置的那个电脑、选中下面列表中内核。见下图



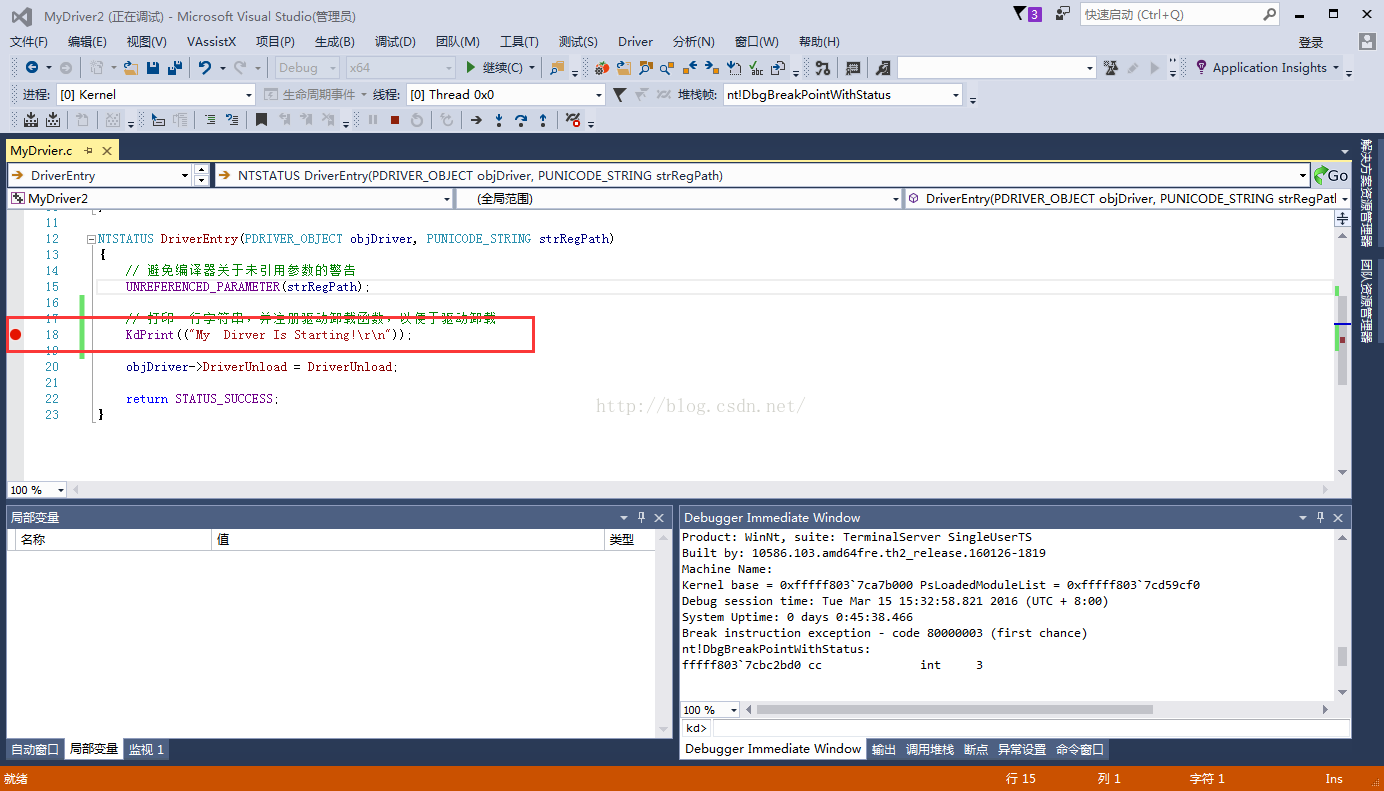
点击附加，然后出现下图界面,点击一下全部中断。



然后系统就断下了，虚拟机的系统我们鼠标已经点不了，现在系统处于挂起状态了。见下图



现在我们可以到源代码中下断点了。我就随便下个断点了，见图：



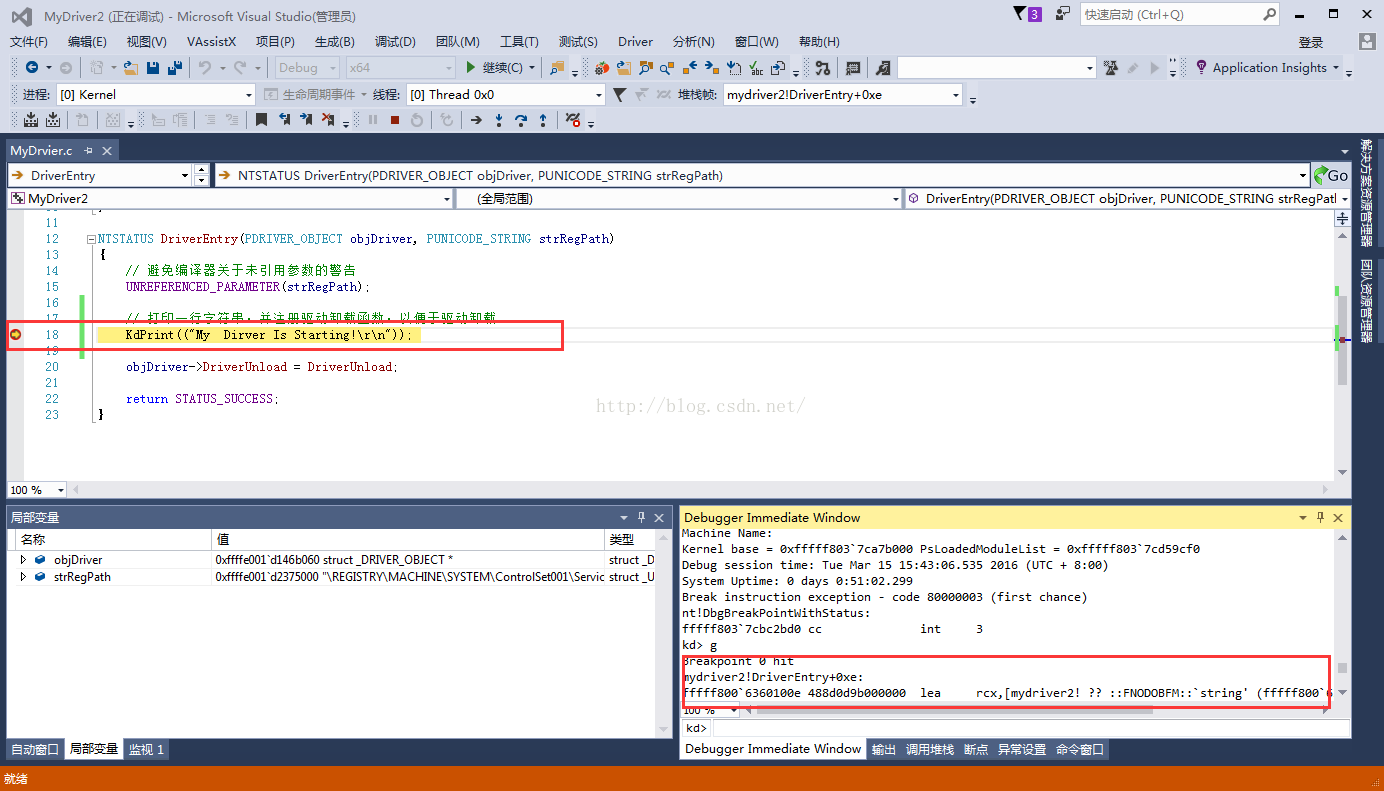
然后我们在下面的Debugger Immediate Winddow窗口的 Kd>  一行输入命令   g     让系统运行起来。这样虚拟机中的系统又可以点击了。

最后一步了。使用我们刚刚建立的驱动工程编译出来的.sys驱动文件拷贝到虚拟机中。用第五步的工具进行安装、启动。启动的时候自动触发VS2015中的下的断点。至此可以像平时我们调试程序一样进行单步调试了。

注意事项：1、一定要拷贝Debug版本的进行调试。

                  2、拷贝到虚拟机中的.sys文件一定要是你源码编译出来的。

                  3、如果改动了源码后一定要重新拷贝一份新的驱动文件到虚拟机中调试，否则无法触发断点的。



**关于驱动签名**

增加一点关于驱动签名的吧，有很多小伙伴编译成功后，驱动安装的时候一直弹出下面的这个图片，驱动没有签名被系统阻止安装，解决方案百度上面说的基本是每次开机都进入启动设置中----禁用驱动签名强制、bcdedit  nointegritychecks on 或者 loadoptions DDISABLE\_INTEGRITY\_CHECKS等。这些方法都是以前的方法基本都没什么用了。

下面提供两个可用方法：

1）使用过期签名（下载个过期签名给驱动签上，把系统时间改到签名有效期内）。签名有版权问题东西就不提供了，自己到看雪能搜到。

2）使用UPGDSED（可以同时关闭PathGuard跟驱动强制签名）使用说明见下：

下载地址：<https://github.com/hfiref0x/UPGDSED>

管理员权限运行patch.exe，按照提示输入大写的CONTINUE按回车确认，patch完成后输入Enter退出，重启系统用另外一个启动项启动系统。本人在windows10 企业版（15063）测试成功，可以不开启测试签名模式下加载64位驱动。

