How to Debug Kernel-Mode Windows Drivers Using VirtualBox and WinDBG

John Larimer
English 202C
7/31/11

Preface

This guide will step through the process of setting up a virtual machine using VirtualBox, configuring that virtual machine to communicate with WinDBG over a virtual COM port and using WinDBG to debug kernel-mode code in a Windows driver running on the virtual machine.

Table of Contents

Preface 2
Table of Contents 3
Before You Begin 4
1. Background Information 4
2. Assumptions 5
3. Requirements 5
4. Tips for Using this Guide 5
5. PART 1: Creating the Virtual Machine in VirtualBox 6
6. PART 2: Configuring VirtualBox and the Virtual Machine 11
7. PART 3: Using WinDBG to Inspect Kernel-Mode Code 14

Before You Begin

Please read sections 1 through 4 before beginning this instruction guide.

1. Background Information

Windows driver testing needs to happen in a virtual machine in order to make the development process safer and more efficient. A virtual machine allows the driver to run in a controlled environment: an instance of Windows within Windows. That way, when the driver crashes, just the virtual machine crashes and not the host computer you are developing on.

Here is some terminology that will be used throughout the guide and is useful if you understand it before you begin:

Host Machine (or physical machine, host OS) – the physical computer hardware that runs VirtualBox and the virtual machine (like a Dell, Apple Mac, etc.).

Virtual Machine (or guest OS) – the operating system running "inside of" the host machine. Think of it as a virtual world in a video game – it mimics a real operating system and executes programs like a physical machine.

The host machine and virtual machine operating systems do not have to be the same, but the CPU architecture does. For instance, an Intel-based Apple Macintosh computer can virtually run Microsoft Windows.

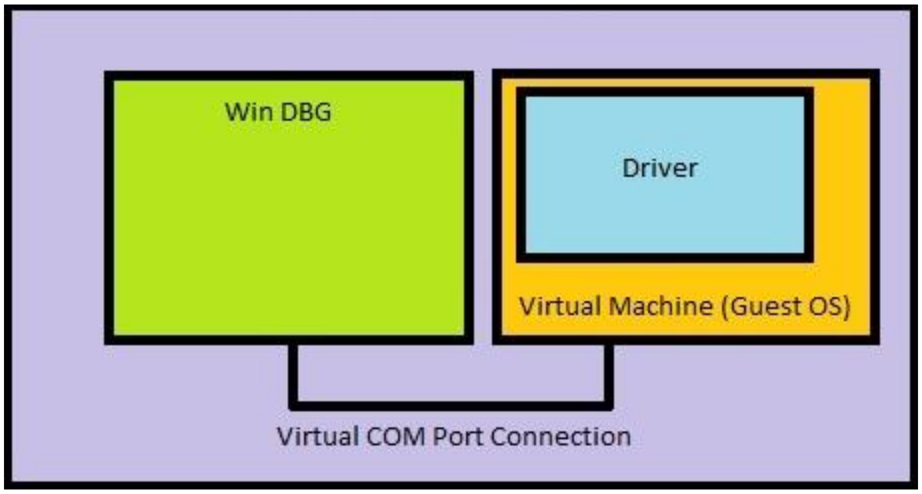


Figure 1: Debugging over virtual COM port abstract diagram

2. Assumptions

This guide assumes novice knowledge of Windows driver development and advanced Windows operating system procedures such as command line operations, BIOS configurations, Windows installation, and use of CD/DVD image files. Most developers that are new to writing kernel-mode applications, such as Windows drivers, are unaware of how to debug their code. The process is more complex than user-mode debugging in an IDE such as Visual Studio. For it to be done safely and successfully requires specialized software and hardware configurations fully explained in the following instructions.

This guide was written using Windows 7 SDK, Windows 7 WDK, VirtualBox version 4.1 and a Windows 7 64-bit host machine and Windows 7 32-bit virtual machine. The exact requirements for you to successfully complete these instructions are listed in Section 3.

3. Requirements

Here are the system requirements for the **host machine**:

- **CPU -** A recent AMD or Intel processor with multiple cores (Core 2 Duo or above).
- **Memory** – at least 3 GB. The host machine and virtual machine each need their own memory because it cannot be shared between them. The more memory in the host machine, the better debugging will go. Sometimes errors and crashes can result from insufficient memory – especially on Windows 7.
- **Hard Disk** – at least 250 GB. VirtualBox and WinDBG do not take up a lot of space but VM's can be 20-40 GB in size.
- **Internet Connection**
- **Microsoft Windows 7 (x86 or x64)**
- **Available COM port**

You will also need a bootable image of Windows 7 (x86 or x64) with a valid license to create the virtual machine.

4. Tips for Using this Instruction Guide

Follow these tips to help you use this guide more efficiently:

- Text written in `courier new and highlighted in grey` signifies a command either in a command prompt window or WinDBG window.
- Screenshots for steps will always be shown below the step.
- Complete each part in order, that is, PART 1 follows PART 2 etc. If you already know how to use VirtualBox, you can skip to PART 2.

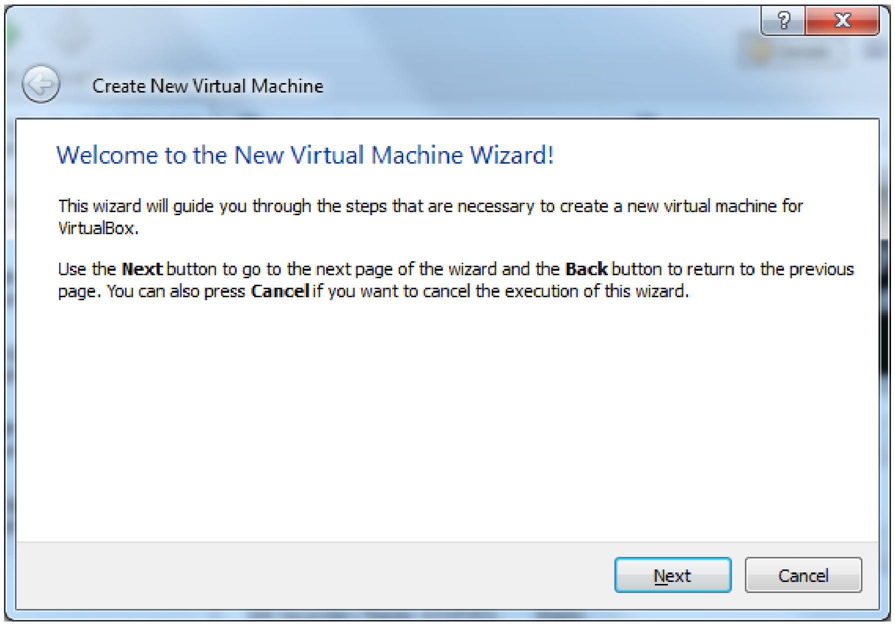
5. PART 1: Creating the Virtual Machine in VirtualBox

In this guide, we will be using a Windows 7 virtual machine running on Oracle's VirtualBox. Think of VirtualBox as the "virtual computer" on which the virtual machine runs. The following steps are a walkthrough on how to install, create and run a VirtualBox virtual machine. You can skip this step and go to PART 2 if you already know how to use VirtualBox and have a VM set up.

Follow these steps for **PART 1**:

1. Download and install VirtualBox from www.virtualbox.org.
2. Once VirtualBox is installed, go to **Machine->New** from the menu bar.

   You should get a dialog like this:



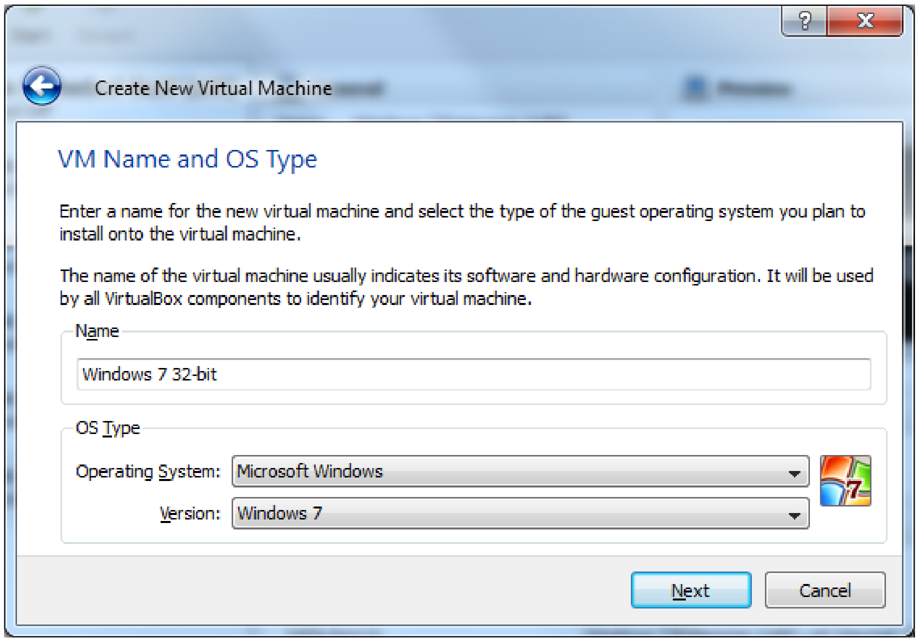
Welcome to the New Virtual Machine Wizard!

This wizard will guide you through the steps that are necessary to create a new virtual machine for VirtualBox.

Use the **Next** button to go to the next page of the wizard and the **Back** button to return to the previous page. You can also press **Cancel** if you want to cancel the execution of this wizard.

Figure 2: Create New Virtual Machine Window

3. Click "Next" and on the following screen when it asks for the "VM Name and OS type", enter a name for the VM and select the appropriate OS from the drop down box. Make sure to select your correct architecture (x86 or x64) as well and hit "Next."


Create New Virtual Machine

VM Name and OS Type

Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine.

The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.

Name: Windows 7 32-bit

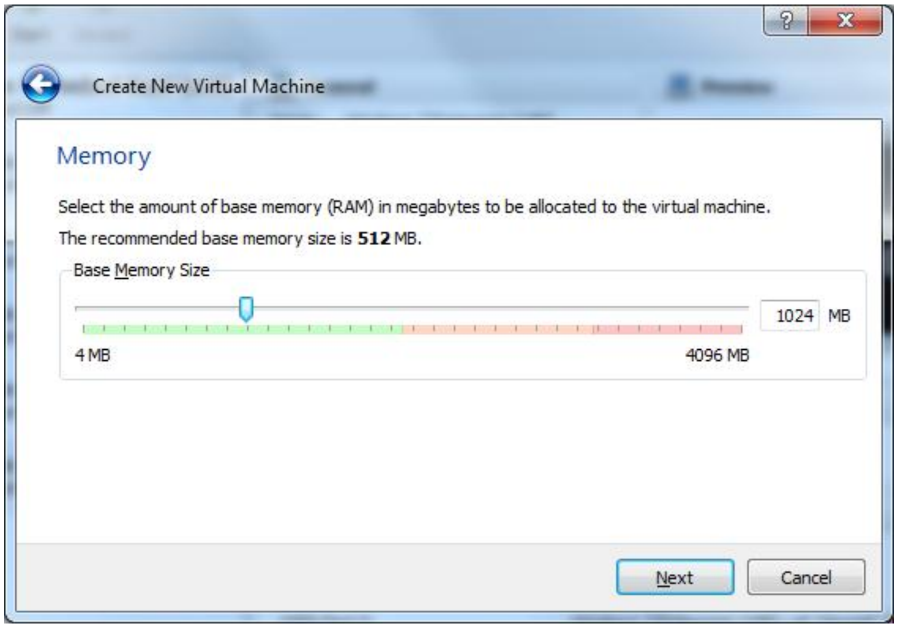
OS Type

Operating System: Microsoft Windows

Version: Windows 7

Figure 3: VM Name and OS Type

4. On the following screen select the amount of RAM you would like to allocate to the VM. It is recommended you allocate AT LEAST 1 GB for a VM running Windows 7. Do not allocate more than half the total RAM available on the host machine. Refer to section 3 if you are unsure of the requirements for this. Hit "Next" after setting the memory.


Create New Virtual Machine

Memory

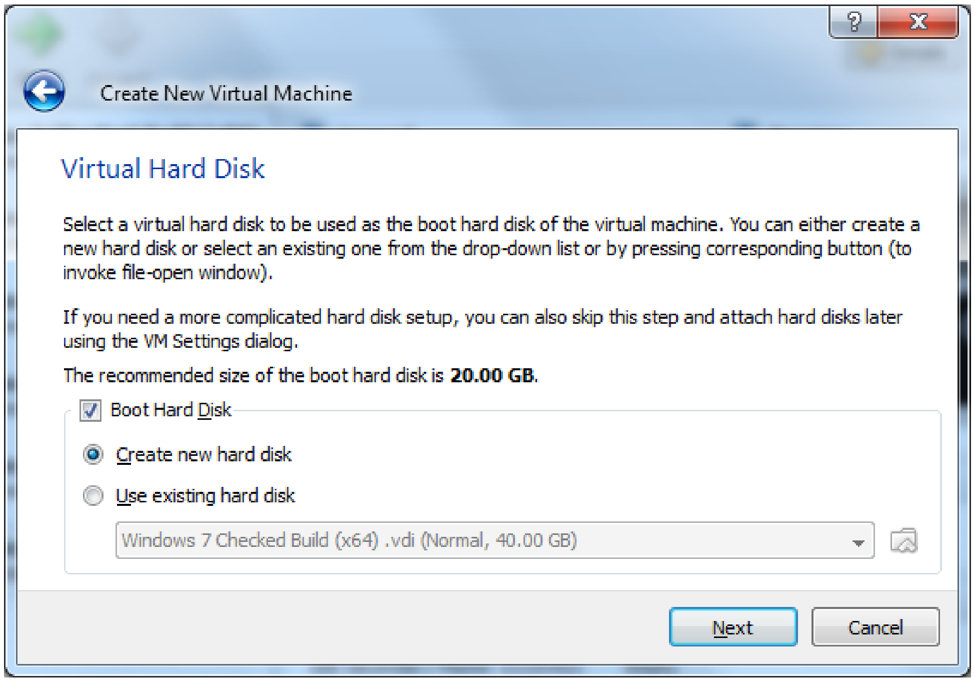
Select the amount of base memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended base memory size is **512** MB.

Base Memory Size: 1024 MB

Figure 4: Memory Allocation

5. On the following screen select "Boot Hard Disk" and "Create new hard disk" and hit "Next."


Create New Virtual Machine

Virtual Hard Disk

Select a virtual hard disk to be used as the boot hard disk of the virtual machine. You can either create a new hard disk or select an existing one from the drop-down list or by pressing corresponding button (to invoke file-open window).

If you need a more complicated hard disk setup, you can also skip this step and attach hard disks later using the VM Settings dialog.

The recommended size of the boot hard disk is **20.00 GB**.

[✓] Boot Hard Disk

[● Create new hard disk]

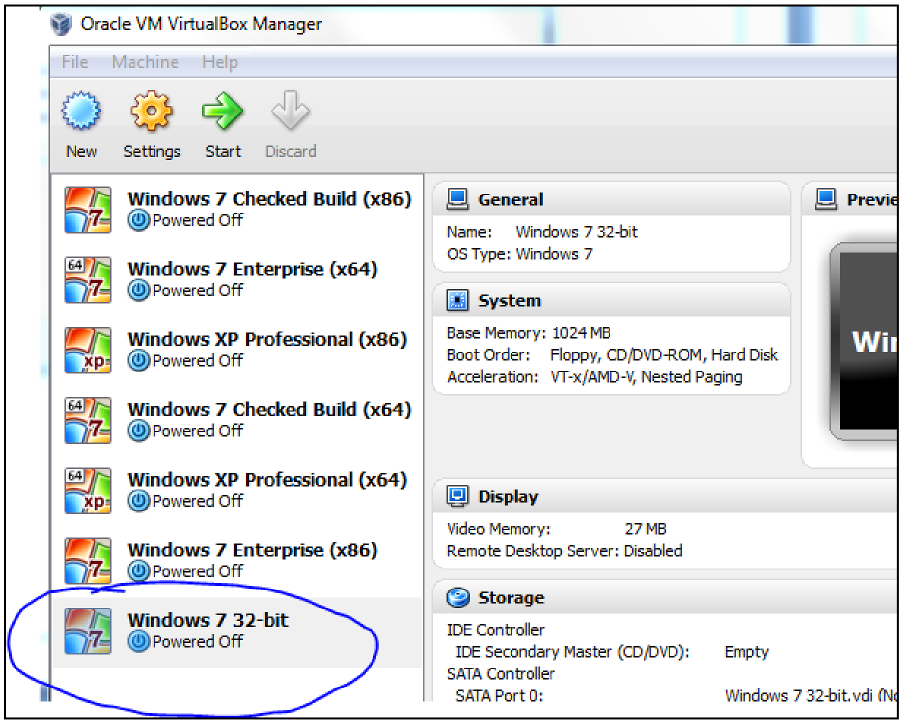
○ Use existing hard disk

Windows 7 Checked Build (x64) .vdi (Normal, 40.00 GB)

Figure 5: Virtual Hard Disk

6. The "Create New Virtual Disk" wizard will pop up. Follow the dialog making sure you set the disk drive to "Dynamically expanding storage" and make the size at least 30 GB. Since the drive will be dynamically expanding, the VM will not actually take up 30 GB but only what it uses.

7. After setting up the "Virtual Hard Disk", click "Finish" and your newly created VM will show up in VirtualBox Manager as in the figure below.



Windows 7 Checked Build (x86) — Powered Off
Windows 7 Enterprise (x64) — Powered Off
Windows XP Professional (x86) — Powered Off
Windows 7 Checked Build (x64) — Powered Off
Windows XP Professional (x64) — Powered Off
Windows 7 Enterprise (x86) — Powered Off
Windows 7 32-bit — Powered Off

General
Name: Windows 7 32-bit
OS Type: Windows 7

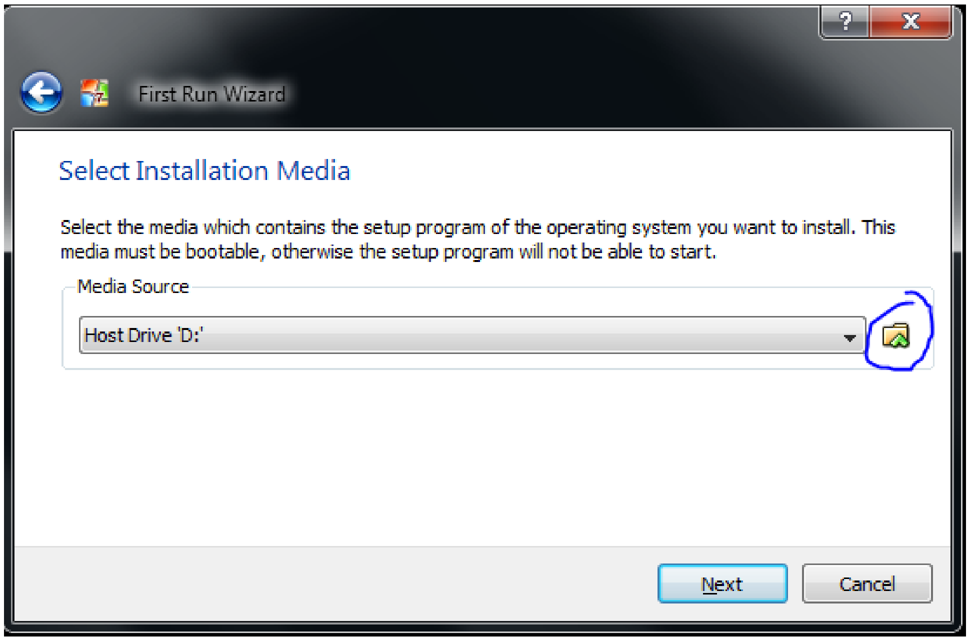
System
Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD-ROM, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Display
Video Memory: 27 MB
Remote Desktop Server: Disabled

Storage
IDE Controller
IDE Secondary Master (CD/DVD): Empty
SATA Controller
SATA Port 0: Windows 7 32-bit.vdi (N

Figure 6: VirtualBox Manager (VM Circled in blue).

8. Double click on the VM you just created to boot it up. The “First Run Wizard” will appear. Click “Next” and THEN click on the little folder with a green arrow (circled in blue in figure 7) when it asks you to select the installation media.



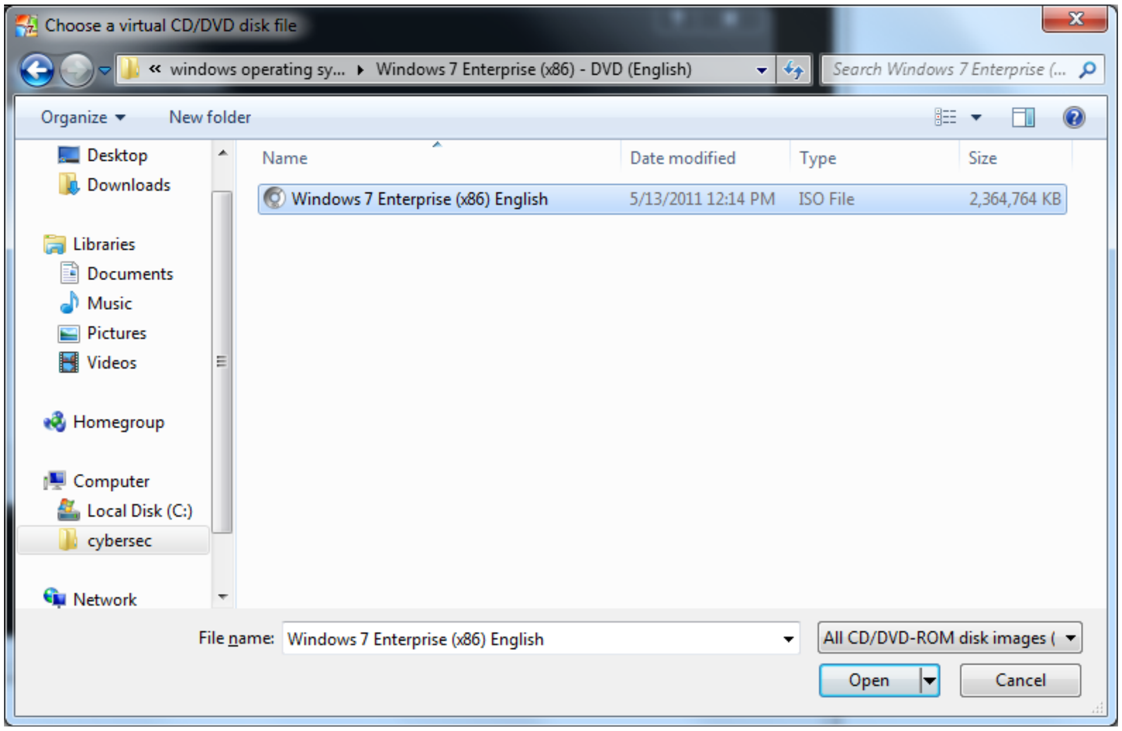
Select Installation Media

Select the media which contains the setup program of the operating system you want to install. This media must be bootable, otherwise the setup program will not be able to start.

Media Source: Host Drive 'D:'

Figure 7: First Run Wizard

9. Browse and select your Windows 7 image file to boot from. This can be an **.iso**, **.dmg**, or **.cdr**. After locating your image file, hit "Open."


Choose a virtual CD/DVD disk file
« windows operating sy... ▸ Windows 7 Enterprise (x86) - DVD (English)
Search Windows 7 Enterprise (...
Organize
New folder
Desktop
Downloads
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
cybersec
Network
File name:
All CD/DVD-ROM disk images (
Open
Cancel

Name	Date modified	Type	Size
Windows 7 Enterprise (x86) English	5/13/2011 12:14 PM	ISO File	2,364,764 KB

File name: Windows 7 Enterprise (x86) English

Figure 8: Choose virtual CD/DVD disk file

10. Hit "Next" and "Finish". The DVD/CD image file you selected will now boot and you can install Windows 7 on the VM just as you would on any normal physical computer. Install Windows and download any necessary updates, and then proceed to PART 2.

6. PART 2: Configuring VirtualBox and the Virtual Machine

Part 2 explains how to set up VirtualBox and the virtual machine (Windows 7) so that kernel-mode debugging can occur over a virtual COM port.

Follow these instructions **after PART 1** has been **completed**.

NOTE: The .vdi file must be type "normal" and NOT immutable to enable debugging on the VM. Only pay attention to this note if you deviated from the instructions in PART 1.

1. Make sure the BIOS setting on your host machine for serial ports is set to "COM1" and not "AUTO". Use the appropriate F function key to get into these settings.

2. Install WinDBG on the host machine and make sure there is an available COM port. You can download WinDBG by downloading Debugging Tools for Windows from: http://msdn.microsoft.com/en-us/windows/hardware/gg463009

3. Change boot settings on VM to enable debugging (via command prompt as admin):

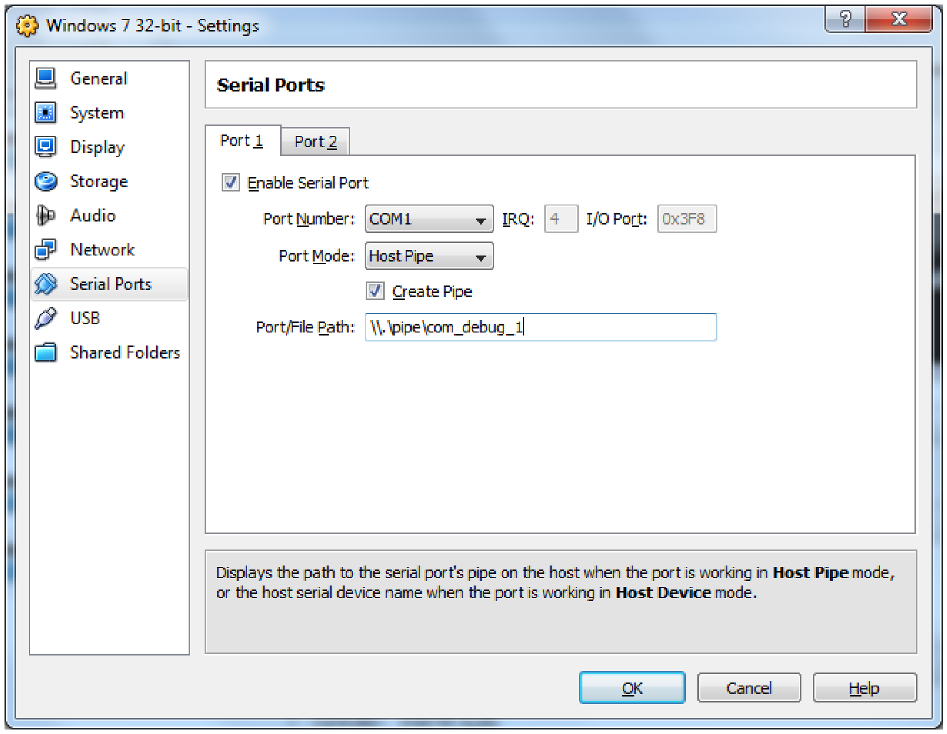
```
a. bcdedit /dbgsettings serial debugport:1
b. bcdedit /debug ON
```

4. Power down the target VM.

5. Add a serial port in VirtualBox Manager (host machine):

    1. In VirtualBox Manager on the host machine, go to **Machine->Settings**->**Serial Ports->Port 1** for the target VM.

    2. Enable Serial Port = ticked, Port Number = COM1, Port Mode = Host Pipe, Create Pipe = ticked, Port/File Path = "\\.\pipe\com_debug_1" (without quotes)



Serial Ports

Port 1 | Port 2

- [✓] Enable Serial Port
- Port Number: COM1 IRQ: 4 I/O Port: 0x3F8
- Port Mode: Host Pipe
- [✓] Create Pipe
- Port/File Path: \\.\pipe\com_debug_1

Displays the path to the serial port's pipe on the host when the port is working in **Host Pipe** mode, or the host serial device name when the port is working in **Host Device** mode.

Figure 9: Serial Port settings

3. Power up the target VM.

Run these commands on the VM to make sure the connection is set up:

```
bcdedit /enum
```

The “Debug” setting should be “ON” for your target boot entry.

```
bcdedit /dbgsettings
```

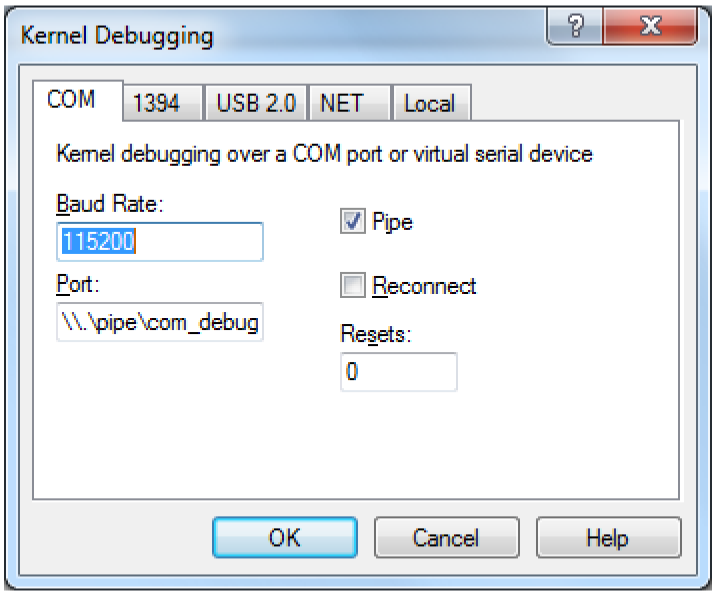
The debug settings should be set to serial and port 1.

If you don’t get these results, go back to step 3 and make sure you are in an administrative command prompt (**Right-click->Run as administrator**).

6. Open WinDbg on the host machine as an administrator and go to **File->Symbol File Path** and set this to: **SRV****your_directory****http://msdl.microsoft.com/download/symbols;*driver_symbols***

your_directory is a location to store debugging symbols. For example, ***your_directory*** could be: **C:\websymbols.** And ***driver_symbols*** is the directory to your driver symbols (the .pdb file).

7. Next, in WinDbg on the host machine, go to **File**->**Kernel Debug** and select the "COM" tab. Set this tab so it matches with the screenshot below and click "OK."



Kernel debugging over a COM port or virtual serial device

Baud Rate: 115200

Port: \\.\pipe\com_debug

Pipe [✓]

Reconnect [ ]

Resets: 0

Figure 10: Kernel Debugging Settings Window

WinDbg should print out (for Windows 7 x86):

```
Opened \\.\pipe\com_debug_1
Waiting to reconnect...
Connected to Windows 7 7601 x86 compatible target at (Tue
Jun  7 16:31:24.271 2011 (UTC - 4:00)), ptr64 FALSE
Kernel Debugger connection established.
```

NOTE: If WinDbg stays on "Waiting to reconnect…", go to **Debug**->**Break** in WinDbg

Move onto PART 3 after you see "Kernel Debugger connection established."

7. PART 3: Using WinDBG to Inspect Kernel-Mode Code

Follow these instructions **after PART 2** has been **completed**.

Once your driver is registered on the VM, you can use WinDbg to step through kernel code. To do this, just as in user-mode code debugging, break points must be set. There are two options to set break points.

Option 1: Set a hard break point:

In your source file, insert the line: `DbgBreakpoint();` where you want the break to occur.

Option 2: Set breakpoint in WinDbg

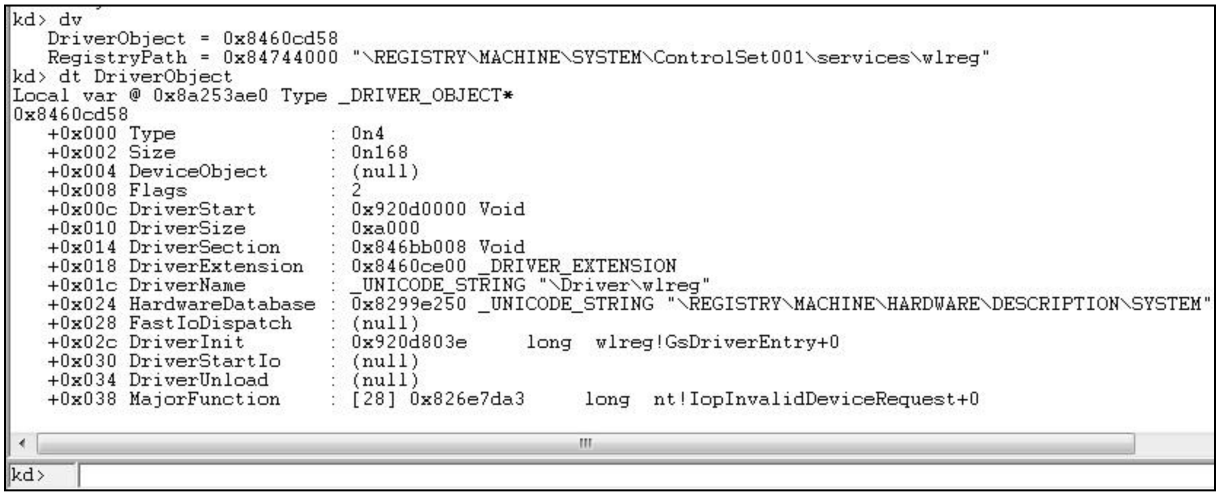
Use the command `bu your_driver!driverentry` in the WinDbg command line to set a breakpoint. `your_driver` = the name of your driver and in this specific example, a break point is set at the start of the DriverEntry() function.

To step through code:

After setting break points, run your driver and WinDbg will break at the specified break points and allow you to step through kernel-code by pressing F10. The source file where the break occurred should also automatically appear. If it does not, go to **File->Open Source File**.

To inspect local variables:

1. Use the `dv` command.
2. Then use `dt` **`variable name`** where **`variable name`** is the name of the variable.

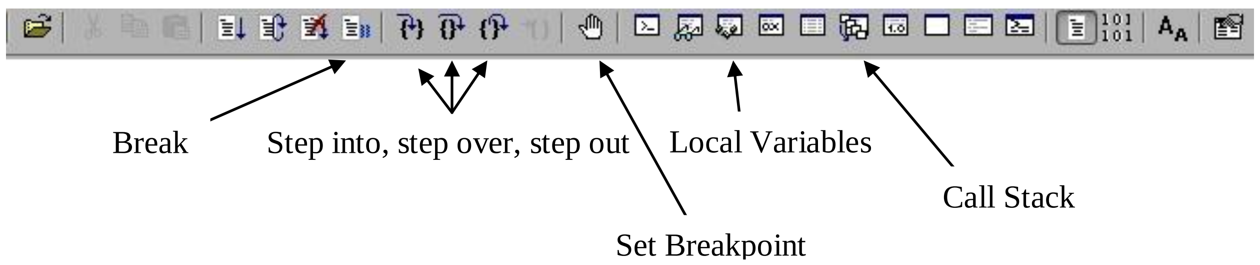


```
kd> dv
   DriverObject = 0x8460cd58
   RegistryPath = 0x84744000 "\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\wlreg"
kd> dt DriverObject
Local var @ 0x8a253ae0 Type _DRIVER_OBJECT*
0x8460cd58
   +0x000 Type             : 0n4
   +0x002 Size             : 0n168
   +0x004 DeviceObject     : (null)
   +0x008 Flags            : 2
   +0x00c DriverStart      : 0x920d0000 Void
   +0x010 DriverSize       : 0xa000
   +0x014 DriverSection    : 0x846bb008 Void
   +0x018 DriverExtension  : 0x8460ce00 _DRIVER_EXTENSION
   +0x01c DriverName       : _UNICODE_STRING "\Driver\wlreg"
   +0x024 HardwareDatabase : 0x8299e250 _UNICODE_STRING "\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM"
   +0x028 FastIoDispatch   : (null)
   +0x02c DriverInit       : 0x920d803e     long  wlreg!GsDriverEntry+0
   +0x030 DriverStartIo    : (null)
   +0x034 DriverUnload     : (null)
   +0x038 MajorFunction    : [28] 0x826e7da3     long  nt!IopInvalidDeviceRequest+0
kd>
```

Figure 11: Sample Inspection of "Driver Object"

Another way to inspect local variables is to open the "Locals" window by pressing "Alt+3."

Commonly Used Commands on WinDbg Toolbar:



Additional Reference:

The website http://windbg.info/doc/1-common-cmds.html lists many additional commands for WinDbg.