

# Sitara Linux SDK Getting Started Guide

## Contents

### Articles

Sitara Linux SDK Getting Started Guide	1
Sitara Linux SDK Supported Platforms and Versions	5
Sitara Linux SDK Software Stack	6
Sitara Linux SDK Creating a SD Card with Windows	7
How to Build a Ubuntu Linux host under VMware	13
Sitara SDK Installer	27
Sitara Linux SDK Setup Script	29
Sitara Linux SDK create SD card script	33

### References

Article Sources and Contributors	39
Image Sources, Licenses and Contributors	40

# Sitara Linux SDK Getting Started Guide

---



**For the Getting Started Guide (GSG) specific to your SDK release, please refer to Archived GSGs under [Archived Versions](#)**

## Welcome to the Sitara Linux Getting Started Guide

Thanks for your interest in learning more about the Sitara Linux Software Development Kit (SDK). The SDK as we affectionately call it is our attempt to provide a great starting point to develop an embedded system on a Sitara Processor running Linux. Given this goal, we wanted to provide something that is more than just a typical Board Support Package (BSP) containing a bootloader, Linux kernel, and filesystem. While these are certainly necessary elements, we feel they are just a starting point, especially for those that aren't experts in developing with Linux. So, the SDK also contains tools for developing on Sitara Processors (a validated cross-compiling toolchain, for example), pre-built libraries that you can use without having to rebuild them yourself, and some documentation to help explain how all of these pieces work together. We package all of this together with a working Linux Embedded System that has been built with all of the things mentioned above, and it contains a featured application called "Matrix" (derived from the fact that it is basically a simple Graphical User's Interface (GUI) of Icon's arranged in a "matrix"). Matrix is a fairly simple embedded Linux system that highlights some of the key features of the Sitara offering (LCD display, graphics, networking, etc.).

What it really serves as is a "known good" starting point. One of the big challenges with starting development on a new platform (not to mention, a new Operating System (OS) for many), is getting an environment set up where you can build and debug code on hardware. The SDK attacks this problem with providing everything you need to do development, and it is validated on standard [TI hardware platforms \(EVMs\)](#). It wraps all of this up into one simple installer that helps get everything you need in the right place to do development. For example, you can start off with simply re-building the Linux Embedded System that we provide to validate that everything is working on your system. This simple step gives you confidence that you can move forward from a good baseline.

As you go along your development journey and have questions, there is documentation and support available to you. Make sure to save a pointer to the [Sitara Linux SDK Software Developer's Guide \(SDG\)](#). It is the best place to start to find the answers that you need. As part of the SDG, you'll find a link to our [Sitara Boot Camp](#) <sup>[1]</sup> training material that will help answer a lot of questions as well. If you don't find what you need, take a look at the active [Sitara Forum](#) <sup>[2]</sup> on the E2E Community and see if the topic has been covered before. If not, post a new thread and we'll do our best to provide some guidance.

## What would you like to do with the SDK?

As described above, the SDK has a lot to it. Let's break it down to two pieces to simplify things a bit:

- The example [embedded Linux system](#) starring Matrix. Essentially, a working bootloader (U-Boot), Linux kernel, and filesystem that can be put on an SD card and ran on a TI EVM, or even one of the very popular Beaglebones (either the original "white" or the newer "black").
- Everything needed to create the above embedded system from "scratch":
  - U-Boot sources and configuration files
  - Kernel sources and configuration files
  - A Linaro cross-compiling toolchain as well as other host binaries and components
  - A Yocto/OE compliant filesystem and sources for example applications in Matrix
  - A variety of scripts and Makefiles to automate certain tasks
  - Other components needed to build an embedded Linux system that don't fit neatly into one of the above buckets

With these two pieces more clearly defined, we can now get back to that all important question, "What would you like to do with the SDK?". If the answer is clearly "I want to build something and I'm ready to start developing now!", then go ahead and skip down to the "I want to Develop!" (or, [Developing with the SDK](#) section below for instructions on installing the SDK on a Linux Host System. This is a somewhat involved process focusing on the second of the two parts of the SDK listed above and may be more than some people want to start with. However, it provides access to the full spectrum of development from rebuilding the SDK from sources to fully adapting it with new device drivers and applications.

So, if you're not quite there yet, let's discuss some other options. Maybe you'd like to evaluate the SDK a bit to see if it is how you'd like to get started. There is a Sitara Boot Camp module entitled "[Introduction to the Sitara SDK](#)"<sup>[3]</sup> that you can view to learn a lot more. This is just one module of the [Sitara Linux SDK Training](#), which we refer to as our Boot Camp material. You can also view all of the documentation online from links on the [Sitara Linux SDK Software Developer's Guide \(SDG\)](#) mentioned previously.

If this is not good enough and you really want to get your hands on something, check out the next section which shares how to play with the embedded Linux system featuring Matrix, the first piece of the SDK mentioned earlier. All you'll need is access to a Windows computer, a SD card, a SD card reader, and some free, open-source software, and a supported [Hardware platform](#).

## Evaluating the SDK Embedded Linux System and Matrix

If you're a hands on person, reading documentation and looking at presentations gets old fast. So, if you want to see an example of what you can build with the SDK and actually hold it in your hands and play with it (or show it to someone else that needs help understanding what you want to do with it), with minimal effort, you can simply run the SDK Embedded Linux System with Matrix on a [supported Hardware platform](#). This will allow you to poke and prod and interact. It's a powerful way to get the imagination active and engaged.

If you've recently purchased a Sitara EVM or Starterkit, they should have come with a SD card with the SDK on them. If that is the case, simply plug the card in, boot it up, and let your imagination run wild. However, if you're like us and the boards you are given never have all of the stuff they came with, or if you purchased a Beaglebone<sup>[4]</sup> or Beaglebone Black<sup>[5]</sup>, you might not have a SD card with the SDK on it. Or, maybe, the SDK on your SD card is simply a few revisions old and you want the latest and greatest. If that is the case, check out the [Creating a SD Card with Windows](#) page. Just remember, you won't be able to build or change anything, simply evaluate the SDK Embedded Linux System with Matrix as delivered. But, even this is enough to get the imagination going and all some folks want to do.

## Start your Linux Development

OK, you're all in. Either you've known this is what you wanted to do, or you've gone through the above steps and you want to do more. It's time to develop! Here's a high level overview:

- Get a Linux host up and running if you don't already have one
- Install the SDK and run some scripts to get everything set up
- Put the SDK Embedded Linux System on a SD card to play with
- Build something to validate set up – the SDK for example
- Add something to the SDK, like a simple Hello World app

After completing these steps, you'll have a known good baseline from which you can start development.

### 1. **Configure a Linux Host** - If you already have a Linux host machine, go to Step 2.

To do Linux development with the SDK, you'll need a host PC running Linux. The Linux host is generally much faster and has a lot more memory (both RAM and hard disk space) than the typical embedded system. While it is certainly possible to do all development natively, we feel the advantages of using a host provide a better way to go and what is supported out of the box with the SDK.

There are many, many ways to get access to a Linux host. We simply can't validate all possibilities and iterations, therefore we focus on validating using Ubuntu <sup>[6]</sup> as the host Linux distribution, running either natively or in a Virtual Machine. We validate the Long-term Support (LTS) versions of Ubuntu at the time of a SDK release (for example, at the time of this writing, Ubuntu 12.04 is the only LTS version).

Can you use other versions of Ubuntu or even other distributions? Theoretically, yes, as long as you can get it to work and there may be more "assembly" required. If you can use the Ubuntu version validated against the SDK, it will be the smoothest path and we will be able to help you more if you do run into trouble.

Likewise, we would strongly recommend getting a **native** Ubuntu LTS machine set up for development. For the cost of a little bit of hard drive space, Ubuntu can have direct access to the host's hardware. Virtual Machines (VMs) have come a long way over the years, and many people use them daily without problems. However, when you are working with a target embedded system (that may be a prototype board), whether it be a TI board or eventually your own, removing the complexity of a VM from the get go can avoid a lot of frustration (i.e. wasted time). When using a VM while connecting and disconnecting hardware components, you have to be very diligent about making sure what is connected to what. You might prefer using an hour to get more work done than debugging a perceived problem caused by the fact the virtual host grabbed a USB port when you weren't watching.

When you're ready to proceed, Ubuntu <sup>[7]</sup> provides a great overview for how to install natively. If you choose to use a VM, here is some information to help you out:

- **Build a Ubuntu 12.04 LTS Linux host with VMware on Win7 (preferred)**
- **Build a Ubuntu 12.04 LTS Linux host with VirtualBox on Win7**

### 2. **Install the SDK** - Within your Linux host machine, **Install the Sitara SDK**

### 3. **Create a SD Card using the SDK Create SD Card Script****NOTE**

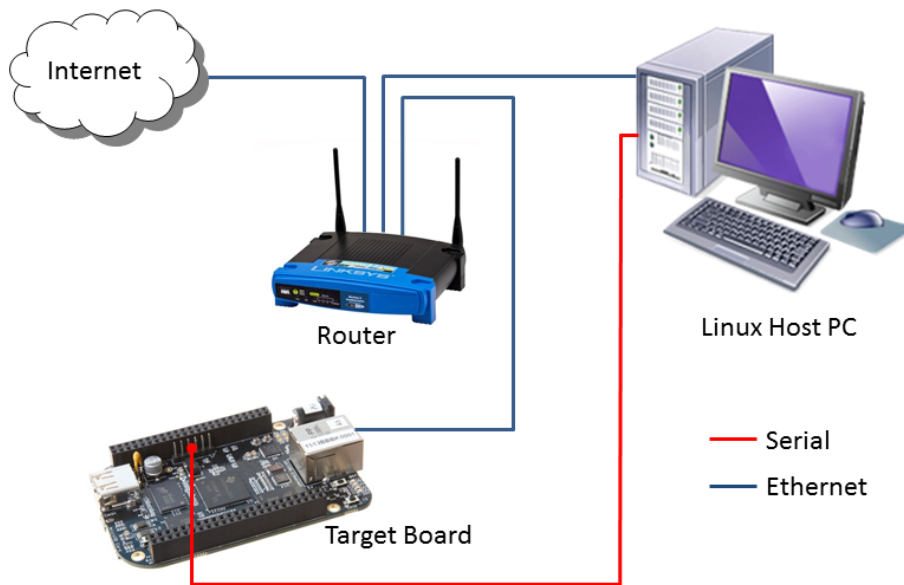
You will need a >2GB SD Card and the capability to connect that card to your Linux Host machine (using a USB SD Card reader, for example).

#### **NOTE**

If using a virtual machine as your Linux host, you may need to import the SD Card reader into your virtual machine (disconnect it from the host and connect it to the VM so that the Linux VM can see it). Please see [here](#) for more information.

### 4. **Configure your development environment**

There are many ways to connect the host development platform and the target board. These connections will vary depending on how you like to develop and what you are trying to do. Here is an example of a common set up with a serial connection for console and ethernet for networking (TFTP, NFS, etc.):



5. **Use the SD Card to boot the target board** properly connected for your development environment.
6. **Run the Setup Script** - Once the SDK has been installed, **Run the Setup.sh Script** to guide you through the remaining development environment configuration.**NOTE**  
If using a virtual machine as your Linux host, you will likely need to import the target board into the virtual machine as a mass storage device using the instructions [here](#)

## What Would You Like to do Next?

Now that you have a solid baseline set up, you can choose what you'd like to do next based on what you need to do. Here are some of the many possibilities:

Link	Summary
<a href="#">Sitara Linux Software Developer's Guide</a>	The SDK's Homepage, a must have link for SDK users.
<a href="#">Sitara Linux SDK Training</a>	The SDK Training page. Lots of great information. The Hands On with the SDK has some great information for developing your first Linux application.
<a href="#">Sitara Linux SDK Documentation</a>	Contains links to all relevant Sitara SDK Documentation.
<a href="#">Sitara Linux SDK</a>	More information on the Linux Kernel provided with the SDK (how to build it, for example).
<a href="#">Sitara Linux U-Boot</a>	Everything you want to know about U-Boot, the bootloader provided with the SDK.
<a href="#">Sitara Linux SDK Filesystem</a>	Details about the various Filesystems delivered with the SDK, and their contents.
<a href="#">Sitara Linux SDK Tools</a>	Documentation for all of the various tools included with the SDK.

## Archived Versions

The Getting Started Guide was first included in SDK 7.0.

## References

- [1] <http://www.ti.com/sitarabootcamp>
- [2] [http://e2e.ti.com/support/arm/sitara\\_arm/f/791.aspx](http://e2e.ti.com/support/arm/sitara_arm/f/791.aspx)
- [3] <http://www.ti.com/llds/ti/arm/training/OLT312001.page>
- [4] <http://beagleboard.org/Products/BeagleBone>
- [5] <http://beagleboard.org/Products/BeagleBone%20Black>
- [6] <http://www.ubuntu.com>
- [7] <http://www.ubuntu.com/download/desktop/install-desktop-long-term-support>

# Sitara Linux SDK Supported Platforms and Versions

The following Sitara ARM microprocessors are supported with this SDK version.

Platforms	SDK	Kernel	U-Boot	Toolchain	Release Date
BeagleBone	7.0	3.12	2013.10	Linaro GCC 4.7	March 2014
BeagleBone Black	7.0	3.12	2013.10	Linaro GCC 4.7	March 2014
AM335xEVM	7.0	3.12	2013.10	Linaro GCC 4.7	March 2014
AM335x StarterKit (SK)	7.0	3.12	2013.10	Linaro GCC 4.7	March 2014

## EVM Hardware Overview

Details for various hardware platforms supported by this Sitara Linux SDK are provided before.

Platform	Document	EVM Provider
AM335xEVM	<a href="#"><u>Hardware User's Guide</u></a> <sup>[1]</sup>	<a href="#"><u>AM335x EVM TI Home Page</u></a> <sup>[2]</sup>
AM335x StarterKit (SK)	<a href="#"><u>Hardware User's Guide</u></a> <sup>[3]</sup>	<a href="#"><u>AM335x Starter Kit EVM TI Home Page</u></a> <sup>[4]</sup>
BeagleBone	<a href="#"><u>Hardware User's Guide</u></a> <sup>[5]</sup>	<a href="#"><u>Beagleboard.org</u></a> <sup>[4]</sup>
BeagleBone Black	<a href="#"><u>Hardware User's Guide</u></a> <sup>[6]</sup>	<a href="#"><u>Beagleboard.org</u></a> <sup>[5]</sup>

## Archived Versions

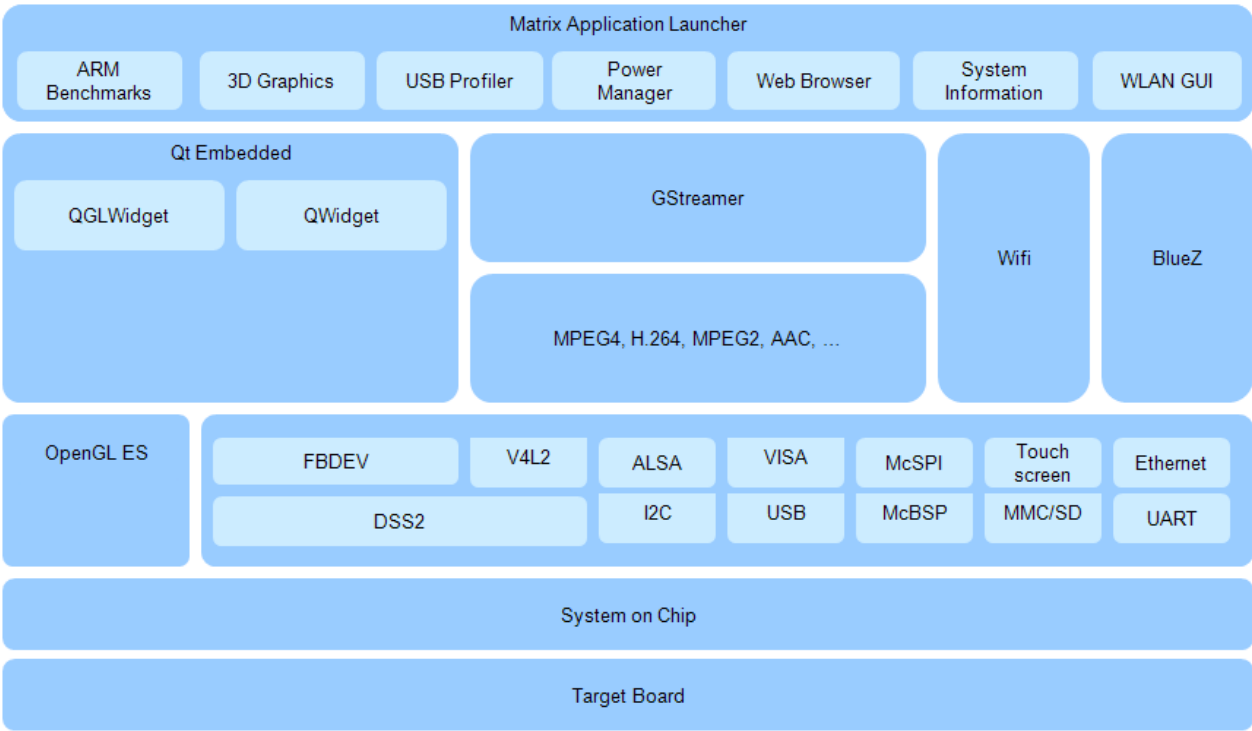
SDK 6.0 <sup>[7]</sup>

## References

[1] [http://processors.wiki.ti.com/index.php/AM335x\\_General\\_Purpose\\_EVM\\_HW\\_User\\_Guide](http://processors.wiki.ti.com/index.php/AM335x_General_Purpose_EVM_HW_User_Guide)  
[2] <http://www.ti.com/tool/tmdxevm3358>  
[3] <http://processors.wiki.ti.com/index.php/AM335xStarterKitHardwareUsersGuide>  
[4] <http://www.ti.com/tool/tmdssk3358>  
[5] [https://github.com/CircuitCo/BeagleBone-RevA6/blob/master/BeagleBone\\_SRM\\_A6\\_0\\_1.pdf?raw=true](https://github.com/CircuitCo/BeagleBone-RevA6/blob/master/BeagleBone_SRM_A6_0_1.pdf?raw=true)  
[6] [https://github.com/CircuitCo/BeagleBone-Black/blob/master/BBB\\_SRM.pdf?raw=true](https://github.com/CircuitCo/BeagleBone-Black/blob/master/BBB_SRM.pdf?raw=true)  
[7] [http://processors.wiki.ti.com/index.php?title=Sitara\\_Linux\\_SDK\\_Supported\\_Platforms\\_and\\_Versions&oldid=171452](http://processors.wiki.ti.com/index.php?title=Sitara_Linux_SDK_Supported_Platforms_and_Versions&oldid=171452)

# Sitara Linux SDK Software Stack

The following software stack illustrates at a high level the various components provided with the Sitara Linux SDK.  
**NOTE - Availability of certain applications are platform dependent and clarified in the associated User Guides below.**



# Sitara Linux SDK Creating a SD Card with Windows

## Introduction

This page details how to use an image file to create a SD Card containing the embedded Linux system provided with the Sitara Linux SDK. This allows a user to evaluate the embedded system on a supported hardware platform.

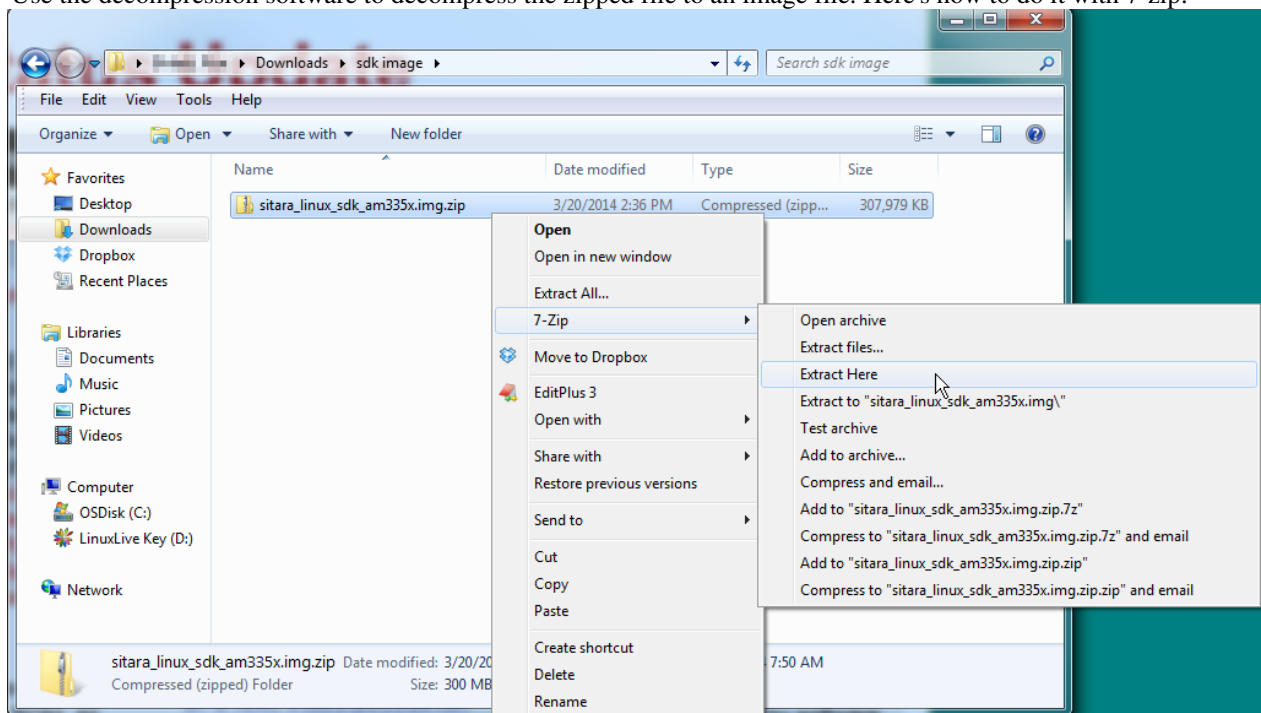
## What is Needed

- Access to a Windows PC
- A valid Sitara Linux SDK image for the appropriate processor (AM335x, for example)
- Software to decompress a zip file (ex. 7-zip)
- Software to write an image file to a SD card
- A SD card appropriate for the required hardware platform, must be 2GB or larger
- A SD card reader/writer

## Steps to Follow

Here is the process to follow to create the SD card.

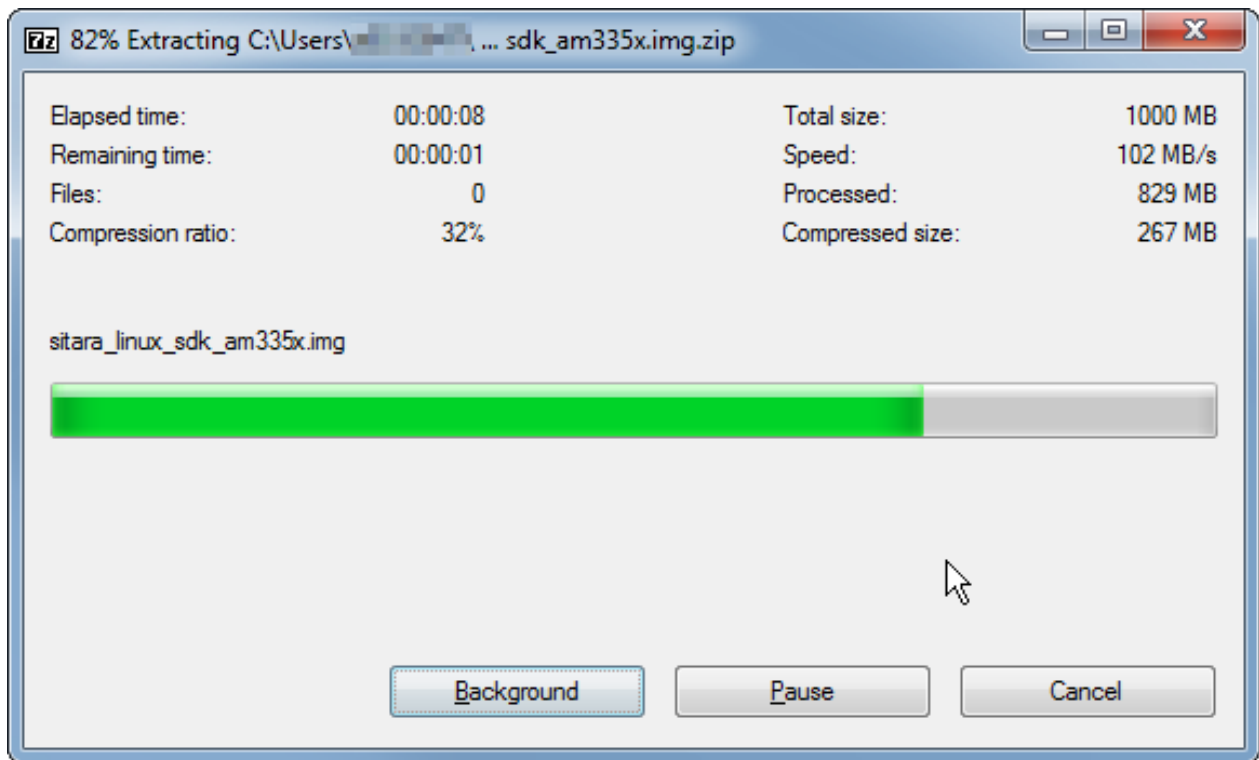
1. Download the Sitara Linux SDK image file that you want to use.
2. On a Windows PC, you'll need software to decompress a zip file. Windows 7 can do this natively. If you don't already have something that works, the open source software 7-zip<sup>[1]</sup> is a great choice. Since this image is created with lots of empty space, this step saves about 700 MB of download time.
3. Use the decompression software to decompress the zipped file to an image file. Here's how to do it with 7-zip:



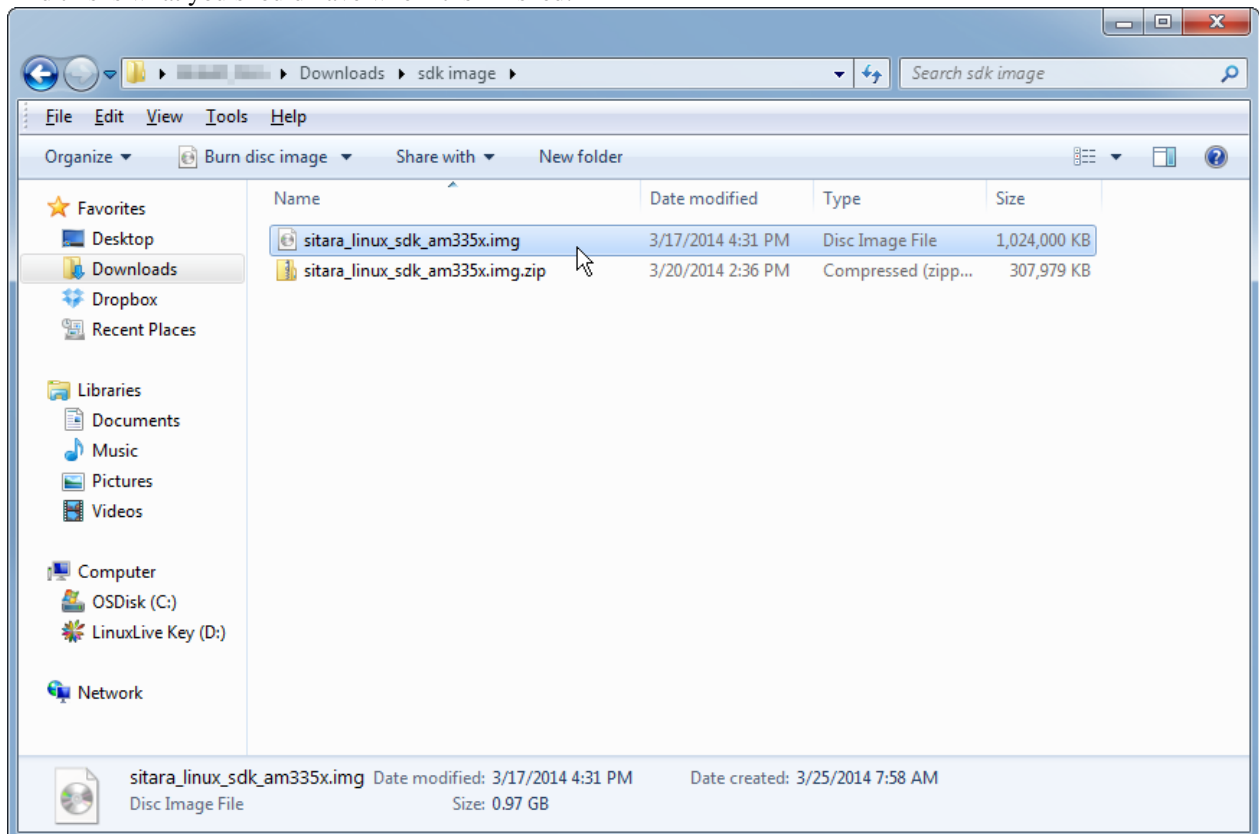
should see a status bar as the image is decompressed:

You



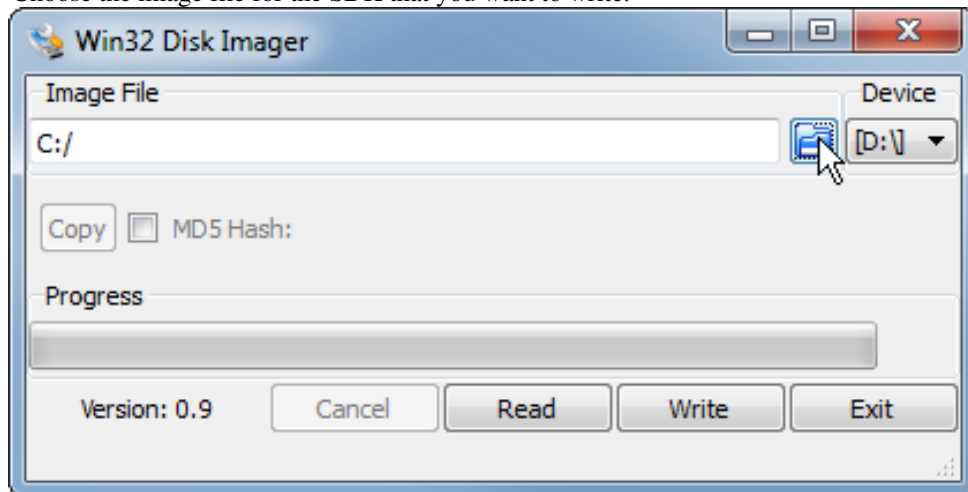


And this is what you should have when it is finished:



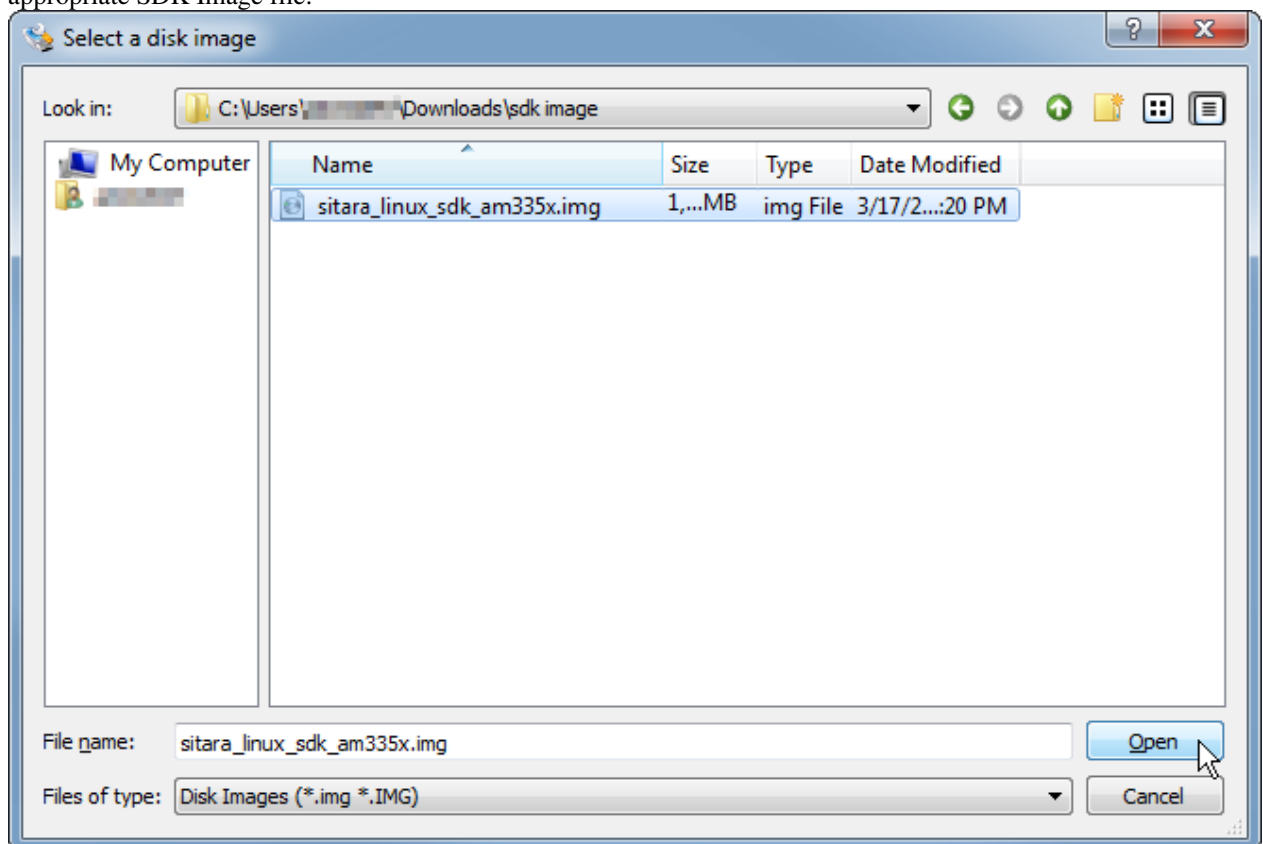
4. If you don't have it already, download a program to write the image file to the SD card. The open source Win32 Disk Imager <sup>[2]</sup> is a good option.
5. Use the software for writing an image to disk to write the decompressed .img file to the SD card.
  1. Plug the SD card into the SD card reader/writer.
  2. Insert the SD card reader/writer into the PC.
  3. Launch the disk writer software, if needed.

4. Choose the image file for the SDK that you want to write.

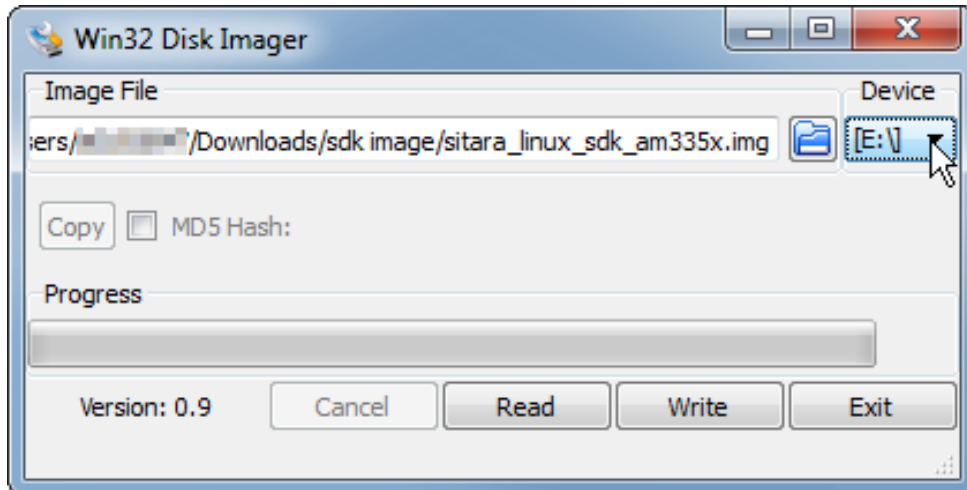


And select the

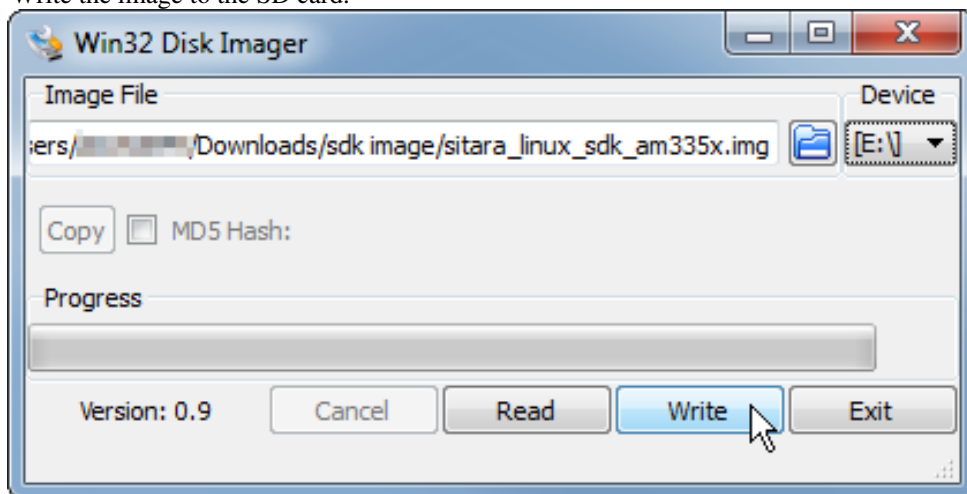
appropriate SDK Image file:




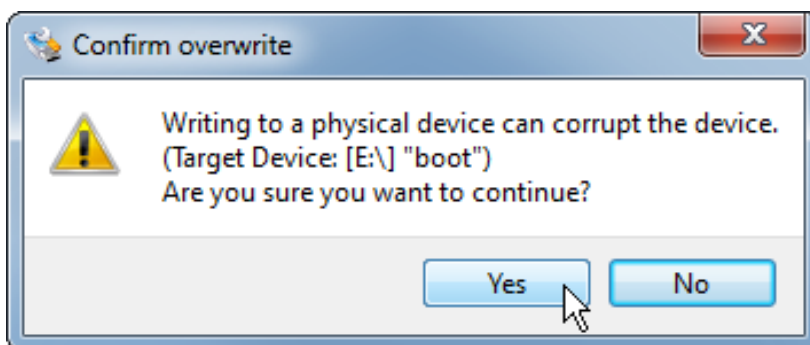
5. Choose the SD card as the destination.



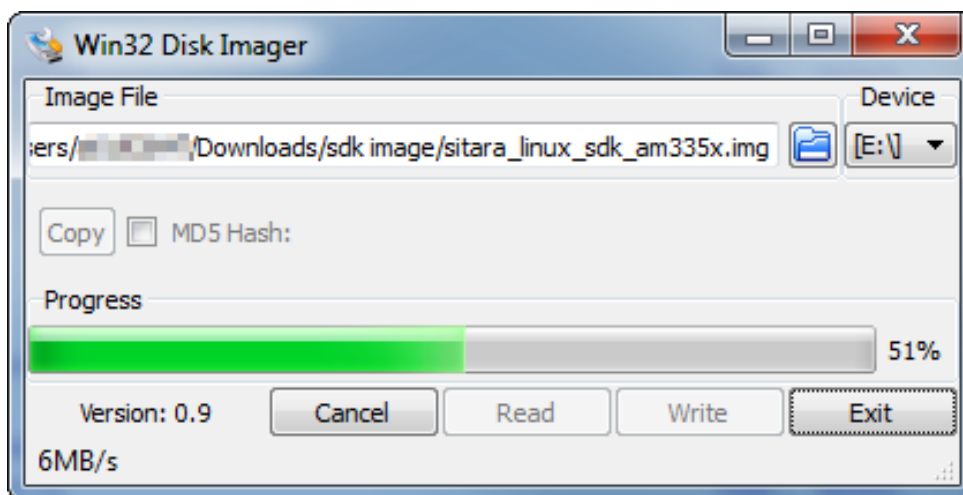
6. Write the image to the SD card.



 **Note:** You'll likely get the below confirmation box. This command will overwrite whatever disk you point it to, please make sure and choose the correct disk:

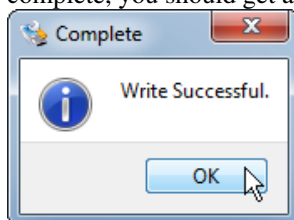


You should see the following status bar as the image is being written to the disk:

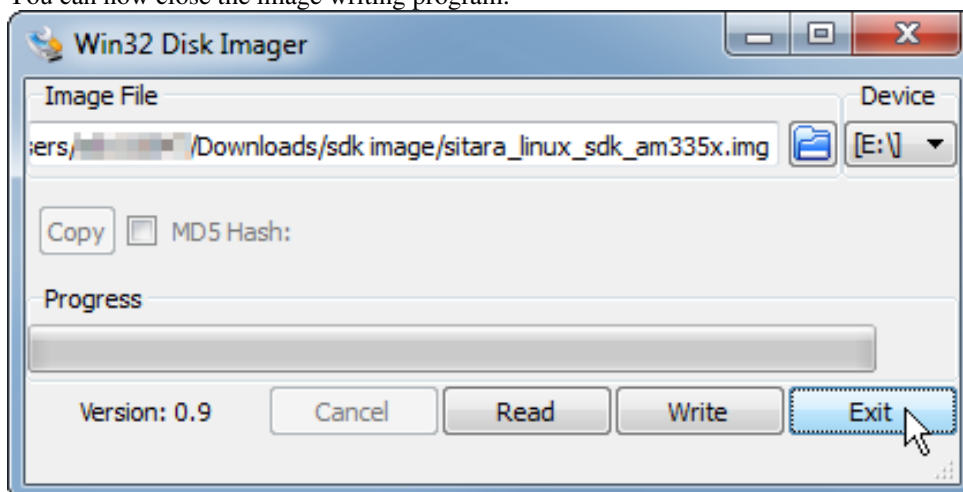


And when the write is

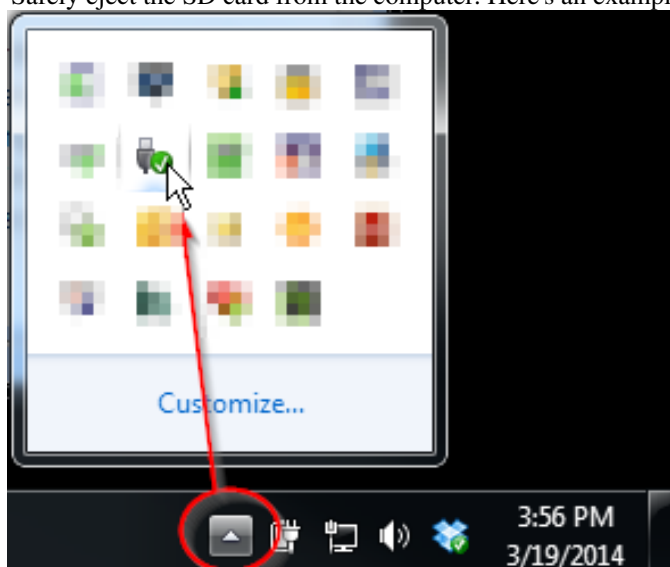
complete, you should get a notification:

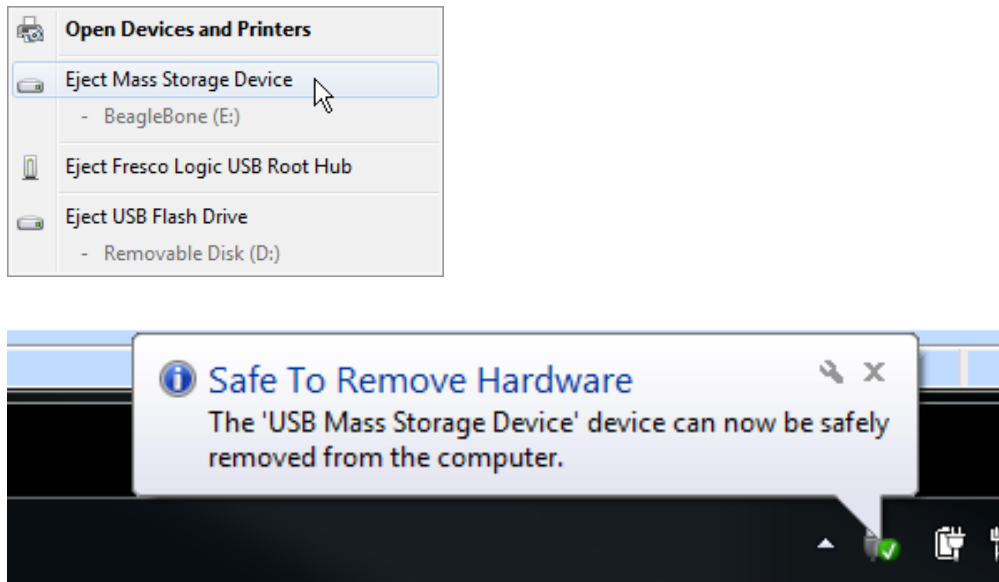


You can now close the image writing program:



6. Safely eject the SD card from the computer. Here's an example using Windows 7:





7. Plug it into a supported hardware platform and boot the platform from the SD card.
8. If the platform has a display (Starterkit, for example), you should see the Matrix application from the SDK. If the hardware does not have a display, you should be able to access Matrix remotely through a web browser if the PC and the board are on a common network. You can also connect to the board using a terminal emulator (ex. Tera Term) in order to view the serial console and interact with the embedded Linux system (ex. run `ifconfig` to get the IP address of the target board in order to connect to it to view remote matrix).

## Useful Links

Sitara Linux Software Developer's Guide

## References

- [1] <http://www.7-zip.org>
- [2] <http://sourceforge.net/projects/win32diskimager>

# How to Build a Ubuntu Linux host under VMware

---



Return to the [Sitara Linux Software Developer's Guide](#)

## Introduction

This guide demonstrates how to get a virtual Ubuntu Linux machine running with VMware under Windows 7. Please use only the 32-bit Ubuntu 12.04 release as this is what is called an LTS (Long Term Support). There are SDK scripts that will be checking for this release identity.

### Requirements:

- Windows 7 host with internet connection, at least 1G of RAM and 40G of free hard drive space.

The instructions here are for setting up a 40G virtual machine. The entire 40G is not taken at once, but as the machine is used and software is installed, the machine can grow and take up as much as 40G.

## Download the Ubuntu 12.04 LTS ISO image

*UNDER CONSTRUCTION !!!*

Get the Ubuntu 12.04 LTS CD ISO image from: <http://releases.ubuntu.com/precise/> [1]. Select PC (Intel x86) desktop CD under the Desktop CD section.

Click download and the follow instructions to download and save the ISO image somewhere. Remember where you save this - you will need the ISO soon!(CD image).

## Download VMware and install

Get VMware from: <http://www.vmware.com> [2]

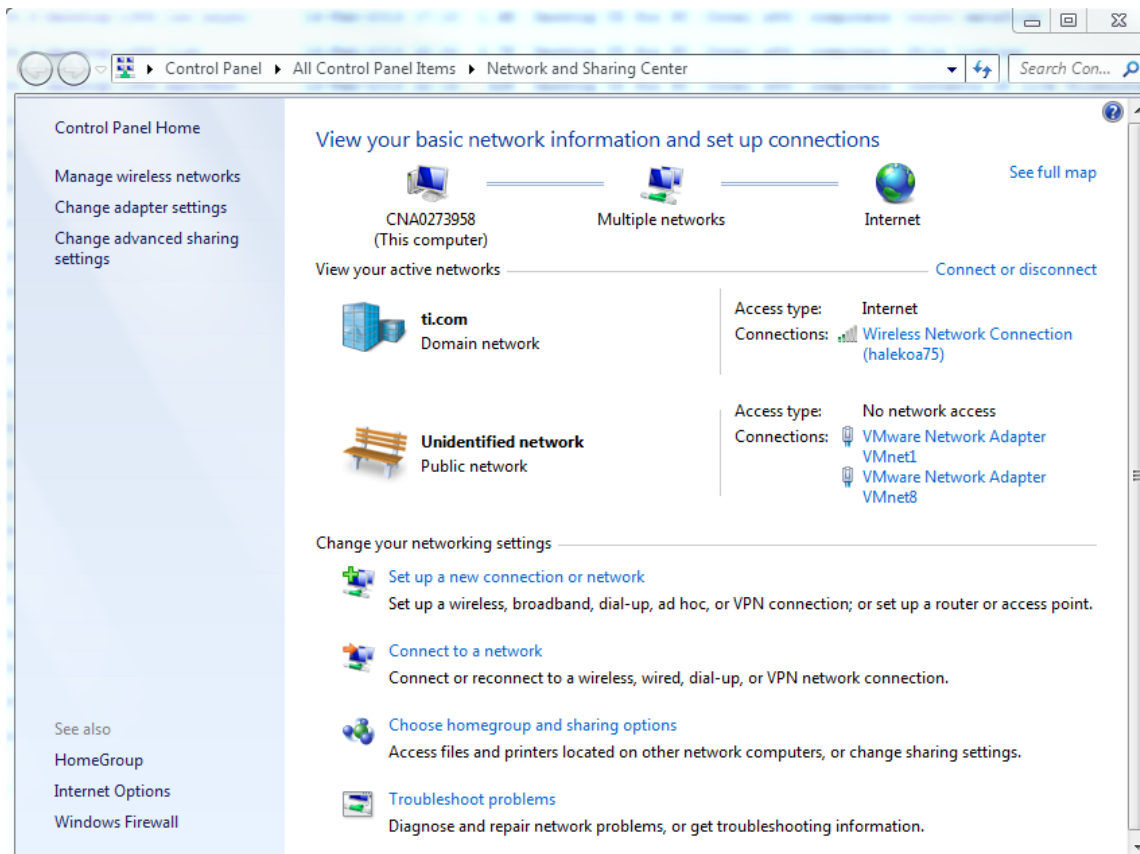
Vmware Player is a free download from the website and enables the user to create an entire virtual machine from scratch using just the ISO image downloaded from Ubuntu. It is necessary to sign up for an account at VMware in order to get to the download areas. The general steps to getting VMware are as follows:

- Login to the vmware website
- Select VMware Player from the products menu
- Follow the steps to download VMware Player

**NOTE - We have tested with v5.0.1 with no known issues. As of June 4, 2013, v5.0.2 is the latest version.**

- Run the executable to install VMware
- Accept license and all default settings.

After VMware is installed the Windows host will have two new (virtual) network adapters. These can be seen in the Windows host by looking under Control Panel --> Network Connections. If the virtual machine will use a bridged connection to an existing network, these virtual adapters should be disabled.



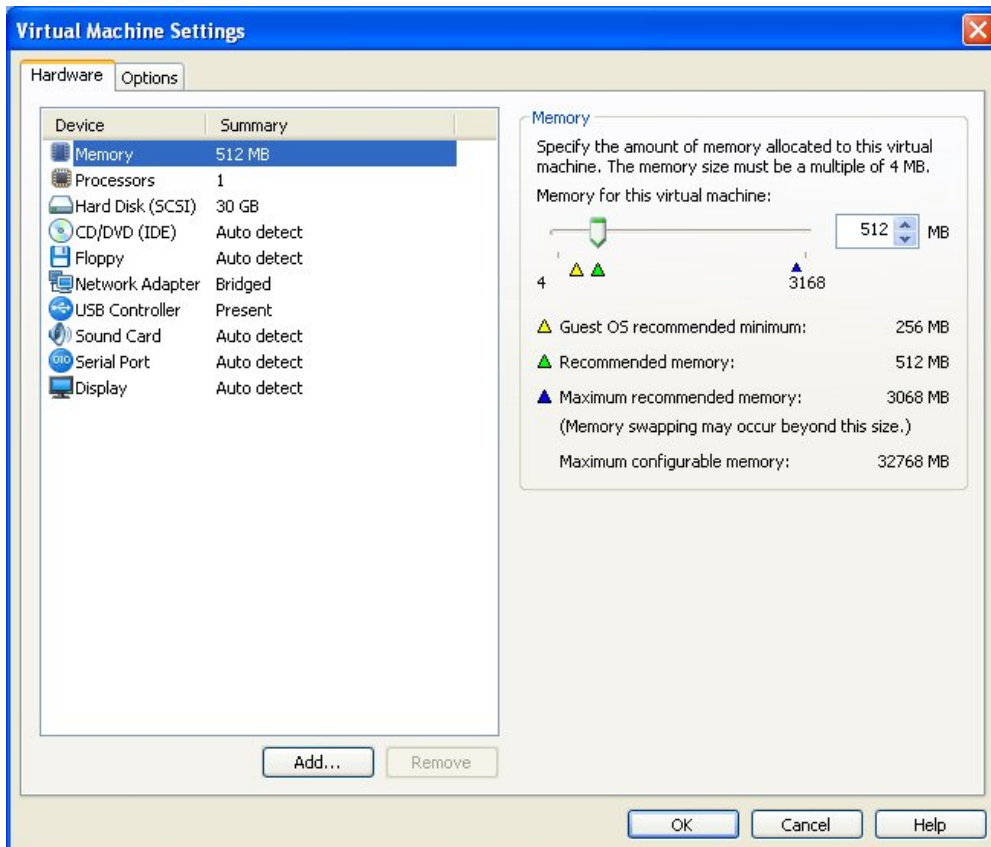
## Create a New Virtual Machine with VMware

Before starting a new installation it is assumed that the Windows host has a proper internet connection to a DHCP server and that the Windows host has enough hard drive space for the new virtual machine.

The following steps are performed with VMware 5.0.1. The exact steps with other versions may vary slightly

- Start VMware.
- From the File menu select "Create a New Virtual Machine..."
- Choose to install the operating system later. Click "Next".
- Select Linux as the "Guest Operating System" and then choose Ubuntu as the "Version". Click "next".
- Provide a "Virtual machine name" and "Location" where the machine will be stored on the Windows host. The defaults are fine here. Click "Next".
- For "Maximum disk size (GB)" it is good to start with 40G if possible. This means that it will take up 40G on the Windows host. Make sure that the Windows host has at least this much before proceeding. It is also a good practice to tell VMware to split the virtual disk into 2G files. This will makes the image easier to copy and transport if necessary. Click "Next".
- Click "Finish" to complete the creation of the virtual machine.

The machine name will now be listed under the home page of VMware. It is necessary to modify some machine settings before playing the machine for the first time. Select the machine in the home page. Under the "VM" menu select "Settings..."



Click on CD/DVD and change the connection to "Use ISO image file". Click on "Browse..." and select the Ubuntu ISO image file that was previously downloaded. Click on Network Adapter and change the Network connection to "Bridged" and then check the box to "Replicate physical network connection state".

### Adding a serial port to the virtual machine

If you plan to use a serial terminal application, a serial port must be added to the virtual machine. This port must be a physical serial port which exists on the host PC. Click on "Add..." and select "Serial Port". Click "Next". Choose "Use physical serial port on host". Click "Next". Click Finish. Click "Ok".

Since this is a physical port on the host PC, it cannot be used by the host PC and the virtual machine at the same time. When the virtual machine is started, the serial port will be unavailable for use by the host PC. If the serial port is being used at the time that the virtual machine is started, the virtual machine will not be able to access the serial port after it is booted up. So if you want the virtual machine to gain control of the physical serial port of the host PC, there can not be any application like hyperterminal or teraterm running on the host PC at the time that the virtual machine is started.

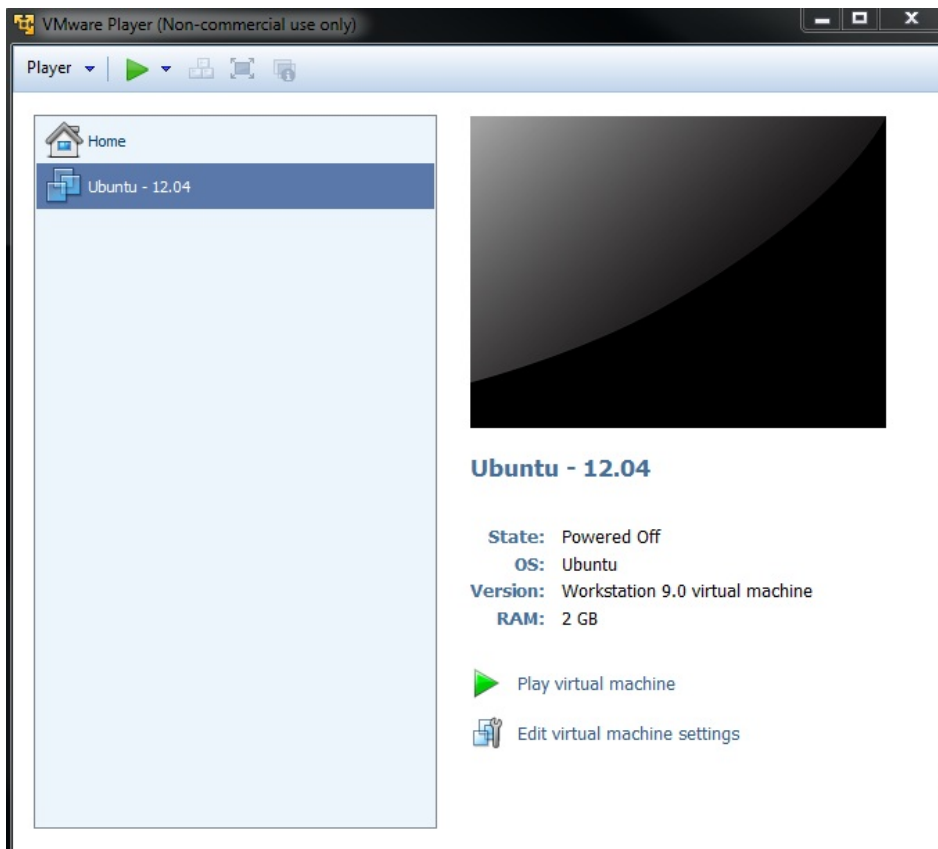
Further instructions for using the serial port with minicom inside of Ubuntu are here <sup>[3]</sup>.

Minicom is the preferred application for use with the Sitara SDK. And the installation and setup of minicom is done automatically by the Sitara SDK installer.

Now click on "Play virtual machine". Since this is the first time starting the machine and the Ubuntu ISO image is in the virtual CD drive, the Ubuntu OS will install itself in the virtual machine.

Click through the Ubuntu installation, making the appropriate choices as you go. When prompted for a login name make the login name **user**. This will help with SDK installation scripts.





The full installation will take 20-30 minutes. When it completes the machine will reboot. The machine will now prompt for the login (**user**) and password.

After the machine reboots into Ubuntu it is helpful to take the Ubuntu ISO out of the virtual CD drive. Click on the VM menu and select "Settings...". Click on CD/DVD and change the connection from "Use ISO image file" to "Use physical drive". The actual drive letter can be selected from the drop down list. If there is no physical drive on the host machine, the CD/DVD device can be simply removed from the machine.

## Install VMware Tools

VMware tools is a very useful addition to VMware. It allows you to resize the VMware screen and also allows cut-and-paste of text from the Ubuntu machine to and from the Windows host.

Later versions of VMware, such as VMware Workstation 7.x, include VMware Tools by default.

Click on the VM menu. Select "Install VMware Tools". The VMware tools are contained in an ISO image that VMware will automatically mount. This drive will show up on the Ubuntu desktop as if it were a disk in a DVD drive. There will be a single tarball on the drive. Copy this tarball to a location in the user filesystem. Extract the tarball and execute the Perl script from the tarball to install VMware Tools. The Perl script must be executed as a super-user. This is done in Ubuntu by pre-pending the command with "sudo". When prompted for a password, enter the password for the user account. In Ubuntu, there is no "root" account. However, the first user account created when Ubuntu is installed can become a super-user with the "sudo" command.

An example is shown below. This assumes that the tarball has already been copied to a directory **/home/user/VMwareTools** and extracted in place. Select all of defaults during installation of VMware Tools.

```
user@Ubuntu1004:~$ pwd
/home/user
user@Ubuntu1004:~$ cd VMwareTools/
user@Ubuntu1004:~/VMwareTools$ ls -l
```

```
total 104400
-rw-r--r-- 1 user user 106900818 2010-11-11 12:27 VMwareTools-8.4.5-324285.tar.gz
drwxr-xr-x 7 user user 4096 2010-11-11 12:26 vmware-tools-distrib
user@Ubuntu1004:~/VMwareTools$ cd vmware-tools-distrib/
user@Ubuntu1004:~/VMwareTools/vmware-tools-distrib$ ls -l
total 560
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 bin
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 doc
drwxr-xr-x 3 user user 4096 2010-11-11 12:26 etc
-r--r--r-- 1 user user 552155 2010-11-11 12:26 FILES
lrwxrwxrwx 1 user user 13 2011-02-14 14:27 INSTALL -> ./doc/INSTALL
drwxr-xr-x 2 user user 4096 2010-11-11 12:26 installer
drwxr-xr-x 17 user user 4096 2010-11-11 12:26 lib
lrwxrwxrwx 1 user user 31 2011-02-14 14:27 vmware-install.pl -> ./bin/vmware-uninstall-tools.pl
user@Ubuntu1004:~/VMwareTools/vmware-tools-distrib$ sudo ./vmware-install.pl
```

## Confirming a Valid Network Connection

After logging into the machine for the first time, bring up a terminal window. This can be found under the Applications menu in Ubuntu. Applications --> Accessories --> Terminal. Type **pwd** in this terminal. This should return **/home/user**. Now type **ifconfig**. This should return information about the network connection. Under **eth0** the IP address should be similar (but not the same) as the IP address owned by the Windows host.

```
user@Ubuntu1004:~$ pwd
/home/user
user@Ubuntu1004:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:da:a8:6e
inet addr:128.247.107.65 Bcast:128.247.107.255 Mask:255.255.254.0
inet6 addr: fe80::20c:29ff:feda:a86e/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:759 errors:0 dropped:0 overruns:0 frame:0
TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:62873 (62.8 KB) TX bytes:4937 (4.9 KB)
Interrupt:19 Base address:0x2024

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:720 (720.0 B) TX bytes:720 (720.0 B)
user@Ubuntu1004:~$
```

## How to Read a USB SD Card Reader in VMware

Sometimes there will be a need to connect a SD Card to the Virtual Machine (for example, to run the create SD card script provided with the SDK that partitions, formats, and populates a SD card that can be used with a target board). When a USB card reader with an SD card is inserted into the USB slot of the host machine, the virtual machine will automatically detect the drive and mount partitions from the SD card, making the card available to Linux running in the virtual machine. If this does not happen automatically (i.e. the SD card cannot be accessed from the Linux VM), it can be done manually by clicking the VM menu and selecting Removable Devices and then selecting the card reader from the sub-menu under Removable Devices. From this sub-menu it is possible to connect or disconnect the USB card reader.

## Using the Target as a USB Mass Storage Device

Some target boards such as the BeagleBone and BeagleBone Black also act as the USB SD card reader when they are booted (using the Linux Mass Storage Gadget running on the target board). This capability allows the host (in this case, the Linux VM), to read and write files on the SD Card mounted to the target board over the USB interface between the Linux Host VM and the Linux system running on the target board. This is only possible if the target board is powered and successfully booted from the SD Card into Linux. For these devices you can import the device as USB SD card reader into the VMWare using the following steps:

1. Once the board is powered on with the Sitara Linux SDK SD card installed you will receive messages like the following letting you know that the device can be connected to the VMWare. Instructions for how to create a SD Card with the SDK are covered here.



2. Click **OK**
3. To connect the target's Mass Storage device for the Beaglebone, Beaglebone Black or AM335x Starterkit, select **Player -> Removable Devices -> Netchip Mass Storage Gadget> Connect (Disconnect from Host)**

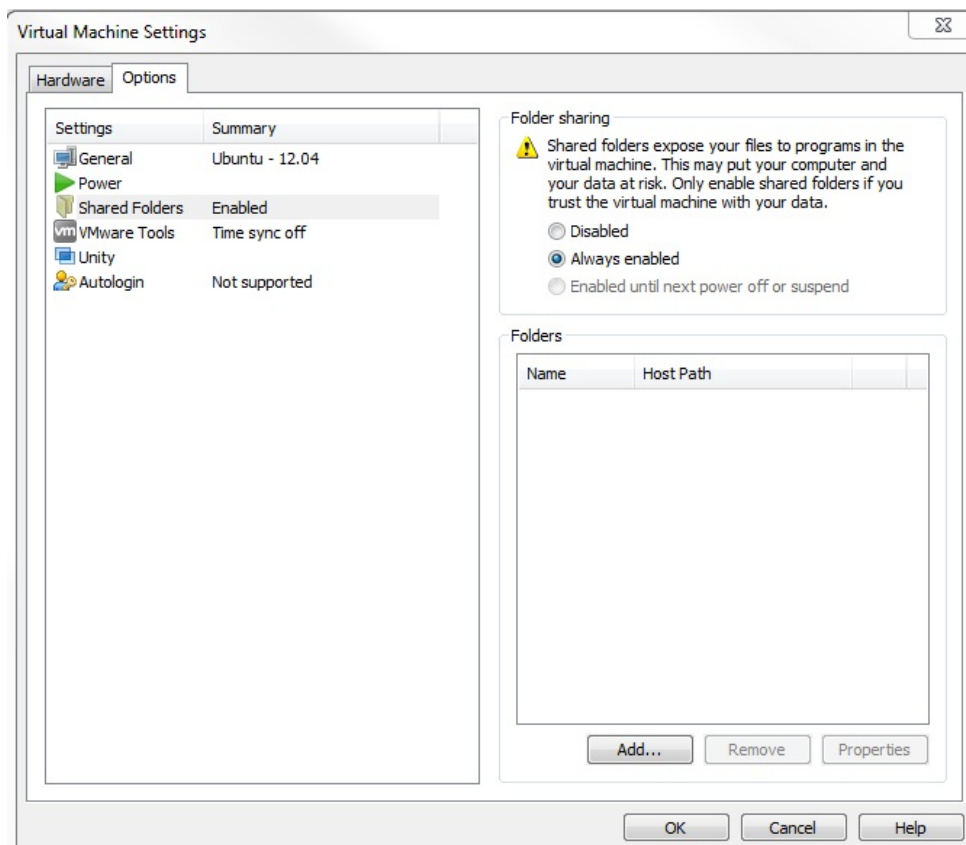


1. You will be prompted that you are about to remove a device from the Host system and connect to the virtual machine. Select **OK** to connect the device.
2. The device should now be available within the virtual machine

## How to Set up Shared Folder in VMWare

The following steps show how to enable Shared Folders within VMWare which allows you to easily share files between Ubuntu 10.04 running on VMWare and your Windows host.

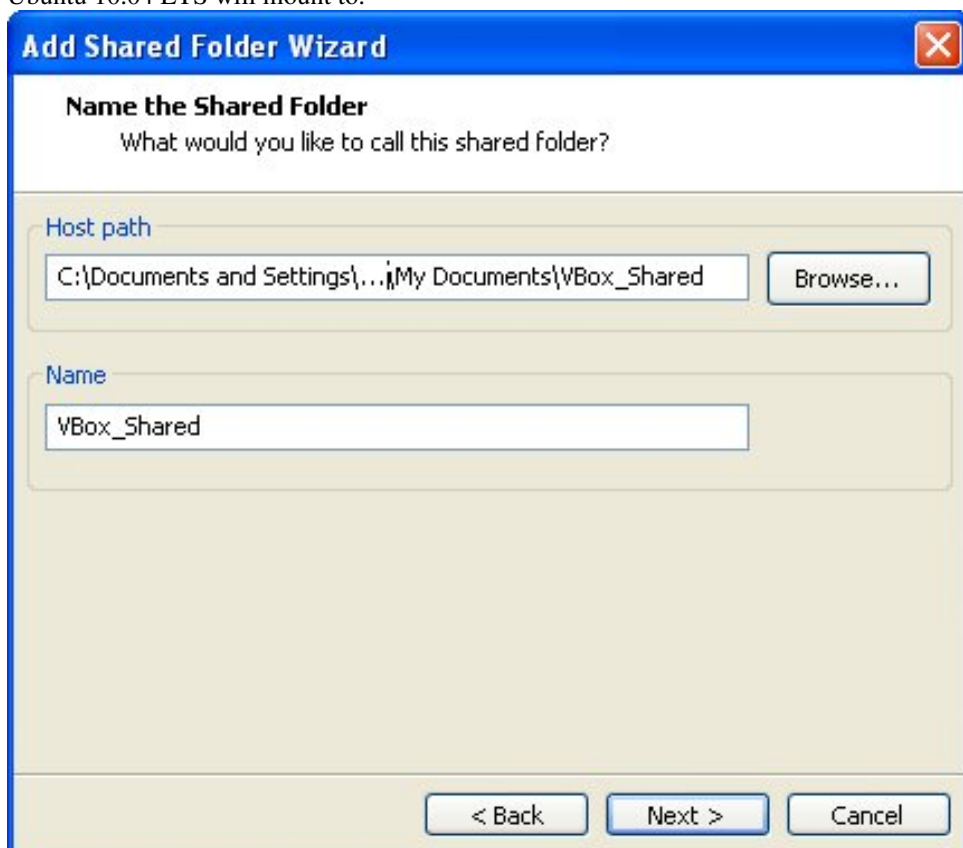
1. Under **Virtual Machine Settings**, the **Options** tab, select **Shared Folders** and **Always enabled**.



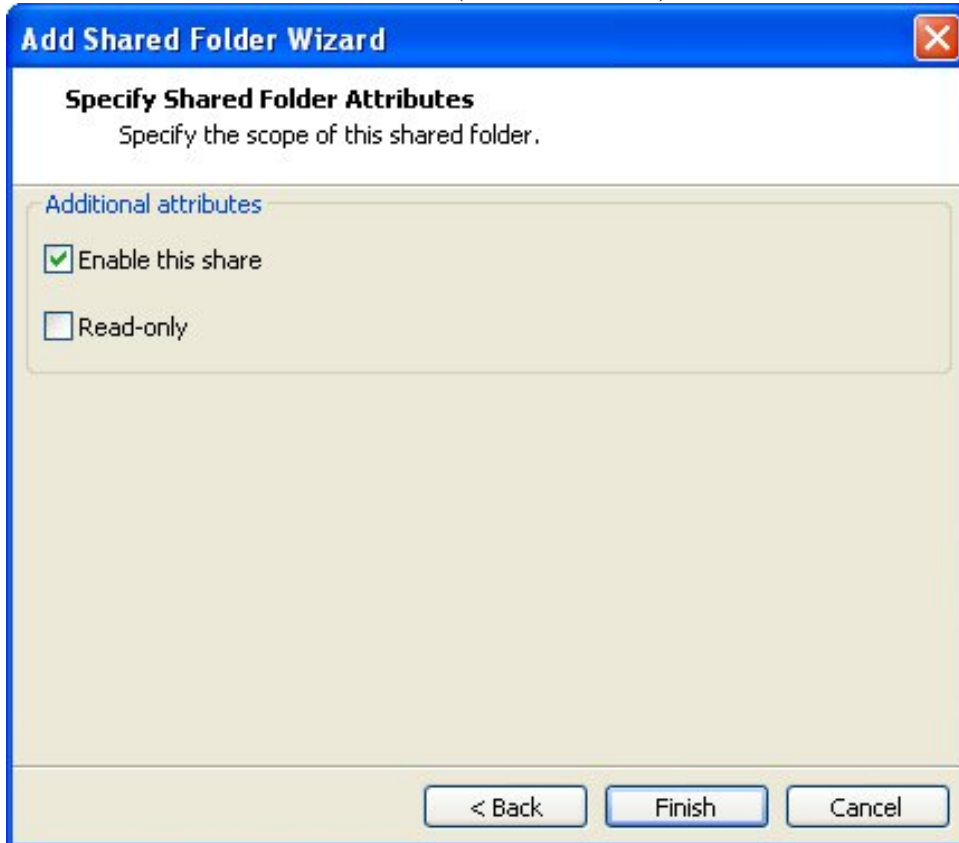
2. Click **Add...** and the following dialog should display.



3. **Browse the Windows folder you want to Share.** And provide a Name for that folder. This Name is what Ubuntu 10.04 LTS will mount to.



4. Ensure **Enable this share** is checked (should be default). And click Finish.



5. **Start your virtual machine and log into Ubuntu 12.04 LTS.** Create a Desktop short-cut to the Shared Folder you just set up.

- See the vmware site for further details
- [http://www.vmware.com/support/ws5/doc/ws\\_running\\_shared\\_folders.html](http://www.vmware.com/support/ws5/doc/ws_running_shared_folders.html)

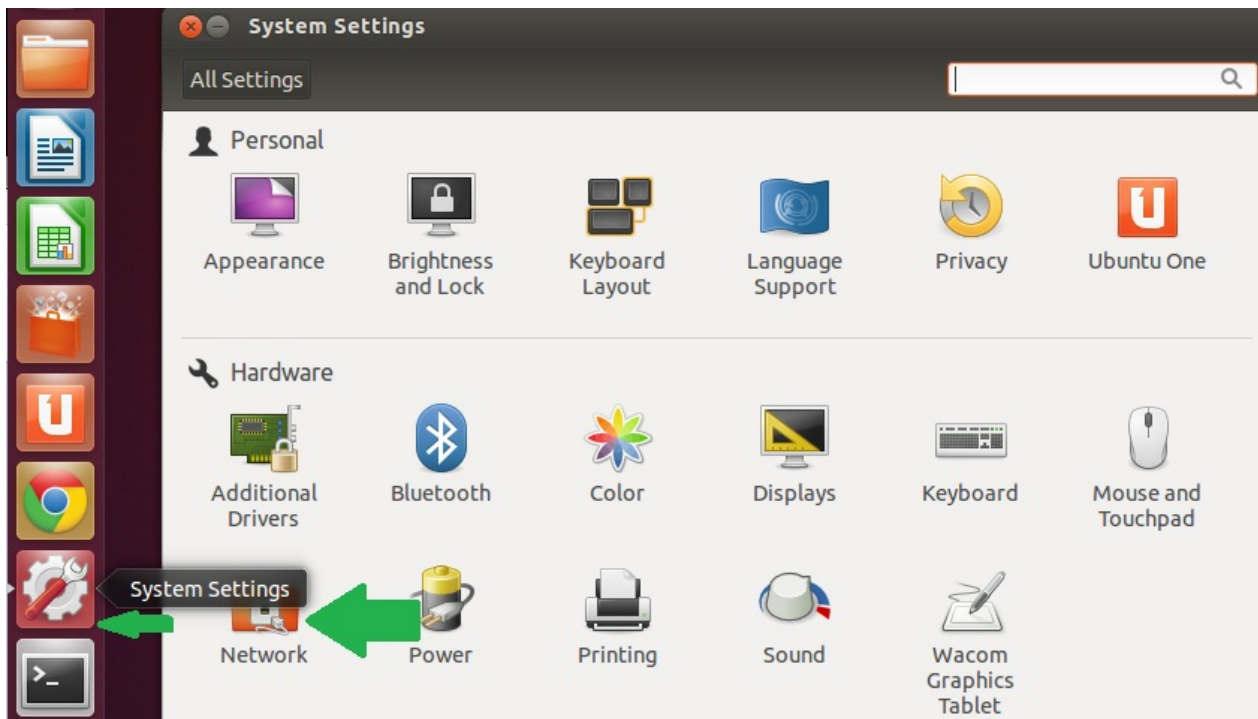
**NOTE - shared\_folder\_name is what ever name your provide in Settings**

After clicking OK, you should have a desktop shortcut to your Shared Folder.

## Configuring a Proxy in Ubuntu

If your network is behind a firewall you will need to configure the network proxy for Ubuntu in order to successfully download the applications required to complete your development environment or to browse the Internet on your Linux Host machine.

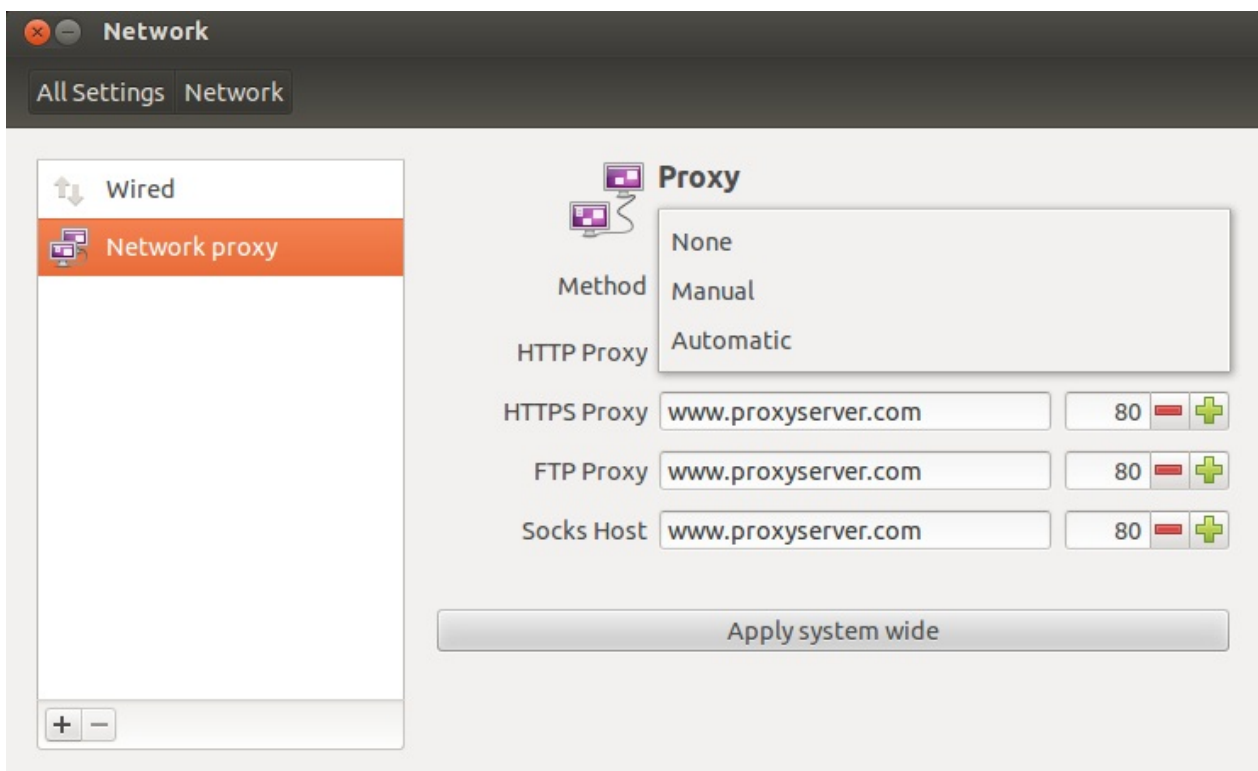
To configure the network proxy in Ubuntu go to System-Settings on the left side in the Launcher and then click Network as seen below.



As seen in the image below, click Network Proxy on the left panel and then Manual Proxy Configuration for the Method. Specify the HTTP proxy server used by your company. You may find this information under your Windows OS inside the Internet Explorer Network Connections. Be sure to specify the port.

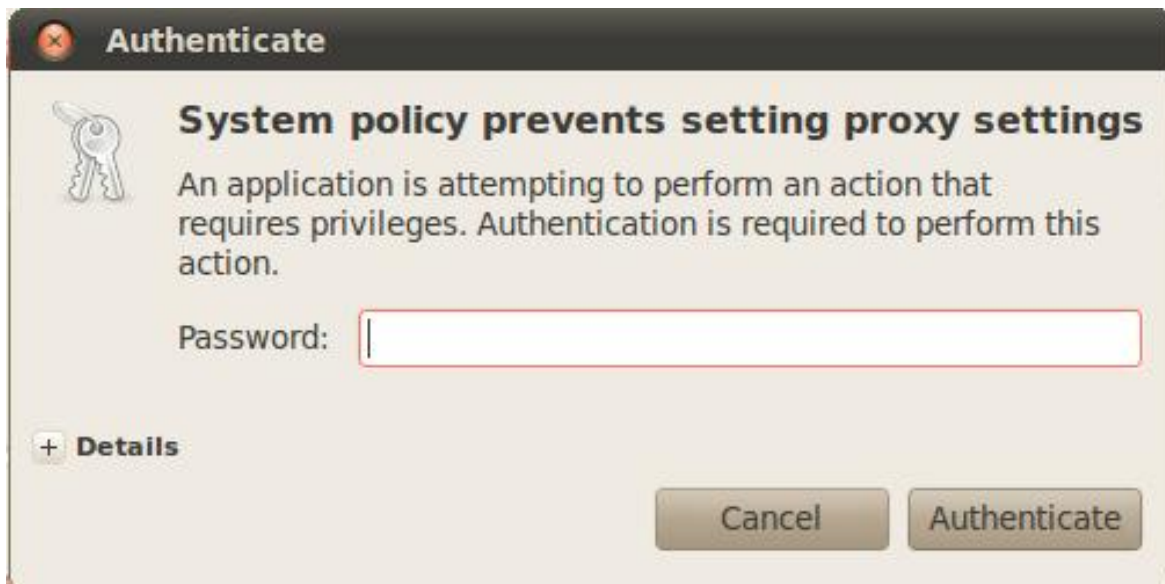
**NOTE - [www.proxyserver.com](http://www.proxyserver.com) is not a valid HTTP proxy. It is shown as an example only. You need use the HTTP proxy server & Port used by your company.**

Make sure you check "Use the same proxy for all protocols". Also be sure to click "Apply System-Wide". Then Close



Finally you will be asked to entered your password in order to set your network proxy information. This is typically the same password you used to log into Ubuntu on boot.





You should now be able to browse the Internet using Firefox within Ubuntu.

### Configuring apt-get to use the proxy explicitly

In case you are unable to use apt-get to download and install packages, then you may need to explicitly setup the proxies in /etc/apt.

Create /etc/apt/apt.conf file as root

```
host$ sudo vi /etc/apt/apt.conf
```

Now add the following text to the file.

```
Acquire::ftp::proxy "ftp://www.proxyserver.com:80";  
Acquire::http::proxy "http://www.proxyserver.com:80";  
Acquire::https::proxy "https://www.proxyserver.com:80";
```

Replace www.proxyserver.com with your proxy server's address and 80 with your proxy server's port.

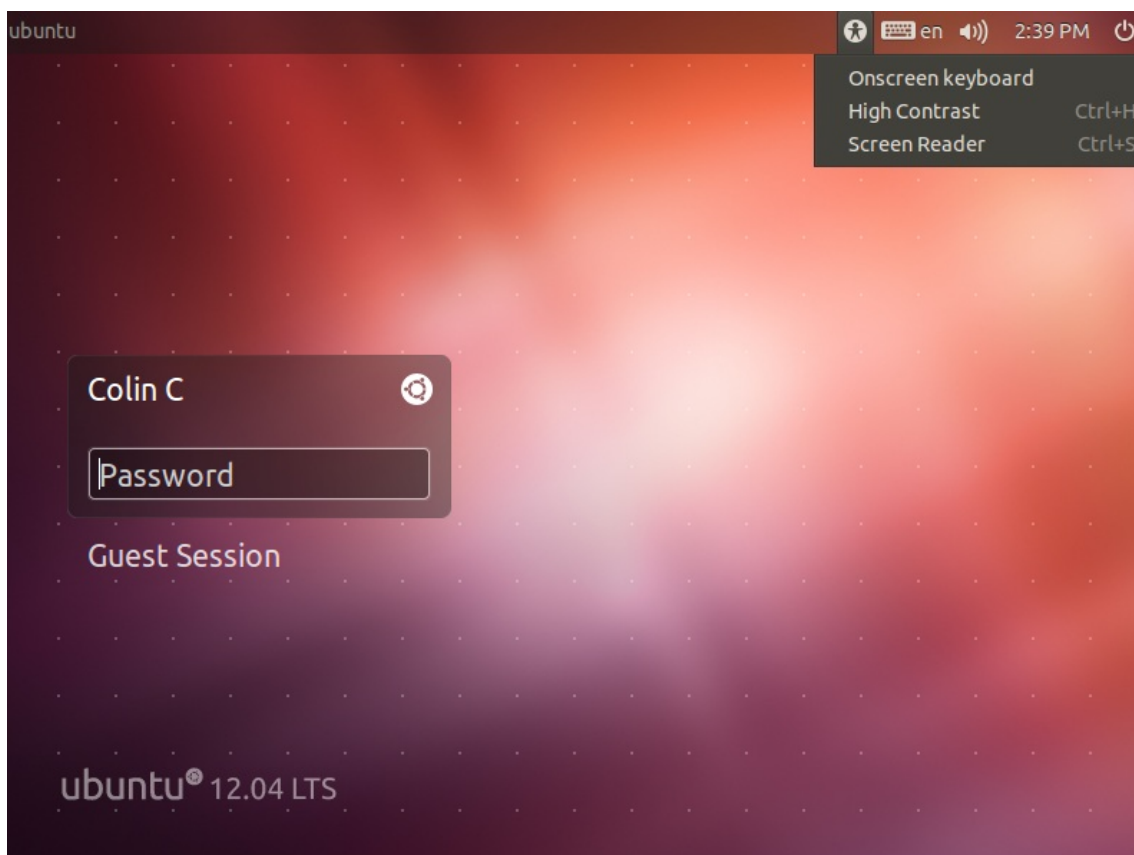
### Keyboard does not Work in Ubuntu 12.04 LTS

If your physical keyboard does not work when you first start VMWare running Ubuntu 12.04 LTS, then you may want to try the following steps to resolve the issue. This assumes your mouse works properly.

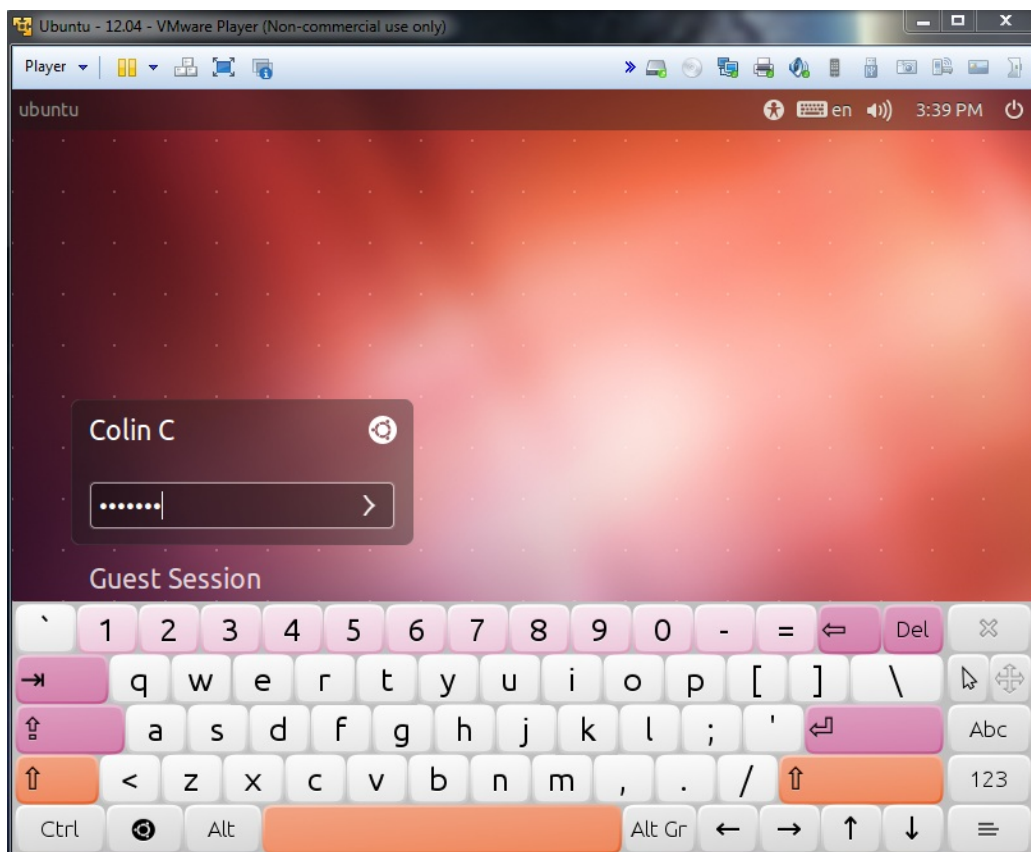
NOTE - The following makes use of the on-screen keyboard provided in Ubuntu 12.04 LTS. Once you are able to log into Ubuntu 12.04 LTS using the on-screen keyboard, issues with the physical keyboard should be resolved.

**Click the figure on the top right panel that looks like a little human; This will display a drop down including an option for a keyboard. Click the Onboard Keyboard option and a keyboard will populate the screen**





NOTE - If the on-screen keyboard does not display, *Shutdown* Ubuntu 12.04 LTS and restart your Ubuntu 12.04 LTS virtual machine again in VMWare. Once Ubuntu 12.04 starts again the on-screen keyboard should be displayed.



**Next, after logging in and launching a terminal, modify /etc/default/console-setup using gedit to enable the physical keyboard**

```
username@ubuntu:~$ sudo gedit /etc/default/console-setup
```

**NOTE - You will need sudo access and therefore must enter your password to access this file.**

At the bottom change the following lines:

```
<original>
XKBMODEL="SKIP"
XKBLAYOUT="us"
XKBVARIANT="U.S. English"
XKBOPTIONS=""

<new changes>
XKBMODEL="pc105"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS=""
```

The physical keyboard should now work from here out. You may also now disable the on-screen keyboard at the login screen using the same steps listed above to toggle the on screen keyboard on.

## Adding Hard Drive space to the Virtual Machine

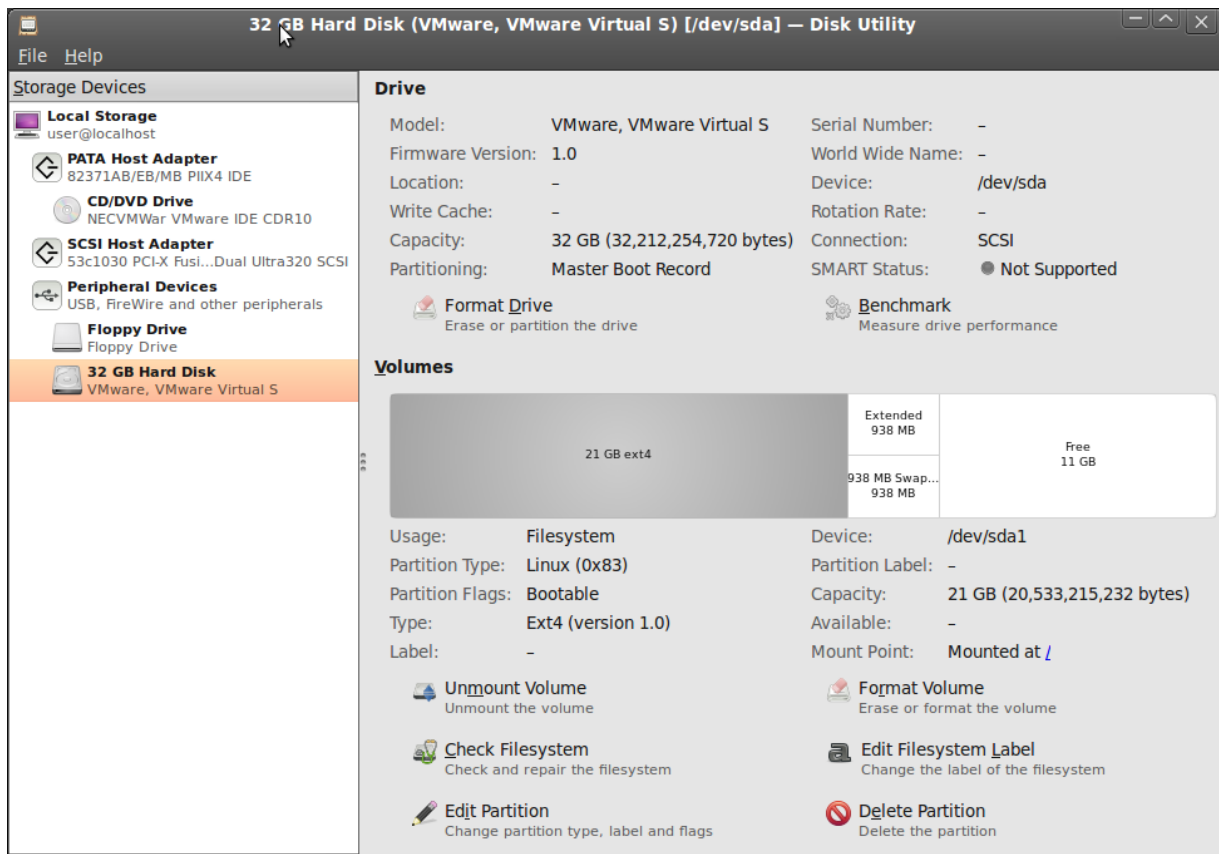
If you followed the instructions in this Wiki and created a 40G Ubuntu 10.04 machine, you will find that this is a good size for the installation of one Sitara SDK and all of its tools. However, there may come a time when you run out of space in the virtual machine. If you have plenty of hard drive space on the Windows host machine then it is very easy to expand the drive of the virtual machine.

The instructions here are for VMware player version 3.1.5 and later.

In VMware, before starting the virtual machine, click on "Edit virtual machine settings" and then click "Hard Disk" on the Hardware tab. In the Utilities menu select "Expand..." and enter a new value for the "Maximum disk size". Click "Expand" then "Ok" to close the settings menu. Now play the machine as usual.

When the machine boots up, Click System --> Administration --> Disk Utility. This will bring up a view of all the storage devices on the system. The Hard Drive here should show a value equal to the amount that was just edited in the machine settings. Click on the Hard Drive and a Volume view should be presented. Here you should see the original Hard Drive as a formatted ext volume. And you should see an additional "Free" partition that is so far unformatted. Click on this "Free" partition and then click the "Create Partition" button. Enter a name for this partition. Now click the "Mount Volume" button. The drive should now show up on the desktop and it is ready to use for additional data space.

The example below shows a machine that was originally 20G and was expanded to 30G. This creates a new 10G partition.



## References

- [1] <http://releases.ubuntu.com/precise/>
- [2] <http://www.vmware.com>
- [3] [http://processors.wiki.ti.com/index.php/Setting\\_up\\_Minicom\\_in\\_Ubuntu](http://processors.wiki.ti.com/index.php/Setting_up_Minicom_in_Ubuntu)

# Sitara SDK Installer



## Overview

The SDK Installer (ti-sdk-amxx-vx.x.x.x) will install the necessary components to start your development on the AMxx microprocessor. The SDK consists of source for the Matrix App launcher starting point application, a development filesystem, a target filesystem, an IDE (CCSV5), Pin Mux Utility, Flash Tool applications, the PSP and documentation.

The Sitara SDK now includes the GCC toolchain from Open Embedded. The ti-sdk was built and tested against a specific Linux Distribution name and version, Ubuntu 10.04 and 12.04. Note this **does not** prevent the user from installing the SDK on other Linux distributions.

## How to Get the SDK Installer

There are two ways you can get the installer:

1. From a file downloaded from the SDK download page <sup>[1]</sup>. This will always host the latest version of SDK.
2. From the SD Card included with a TI EVM. This may not be the latest version of the SDK. We recommend checking the above site and using the latest version if at all possible.

Before running the SDK Installer from the SD card, the SD Card from the EVM box needs to be mounted to your Linux Host PC (using a USB SD Card reader). The SDK Installer is found in the START\_HERE partition of the SD card.

## How to Run the SDK Installer

Run the SDK Installer by double clicking on it within your Linux host PC. Alternatively, bring up a terminal window and change directories to the where the installer is located (probably the Downloads directory if downloaded or the START\_HERE partition mounted from the SD Card) and run the SDK Installer with the following command:

```
[prompt]:~$ ./ti-sdk-amxx-evm-x.x.x.x-Linux-x86-Install
```

### NOTE

If nothing seems to happen, you are probably running a 64-bit version of Linux. The installer is 32-bit, and will not execute properly. See this page for more information and a fix.

## SDK Installer Execution Steps

### 1. Confirm

User is to confirm if loading the ti-sdk is ok. This is important to note in the user is trying to over-install on an existing directory and has made changes to the directory.

### 2. Directory Install Location

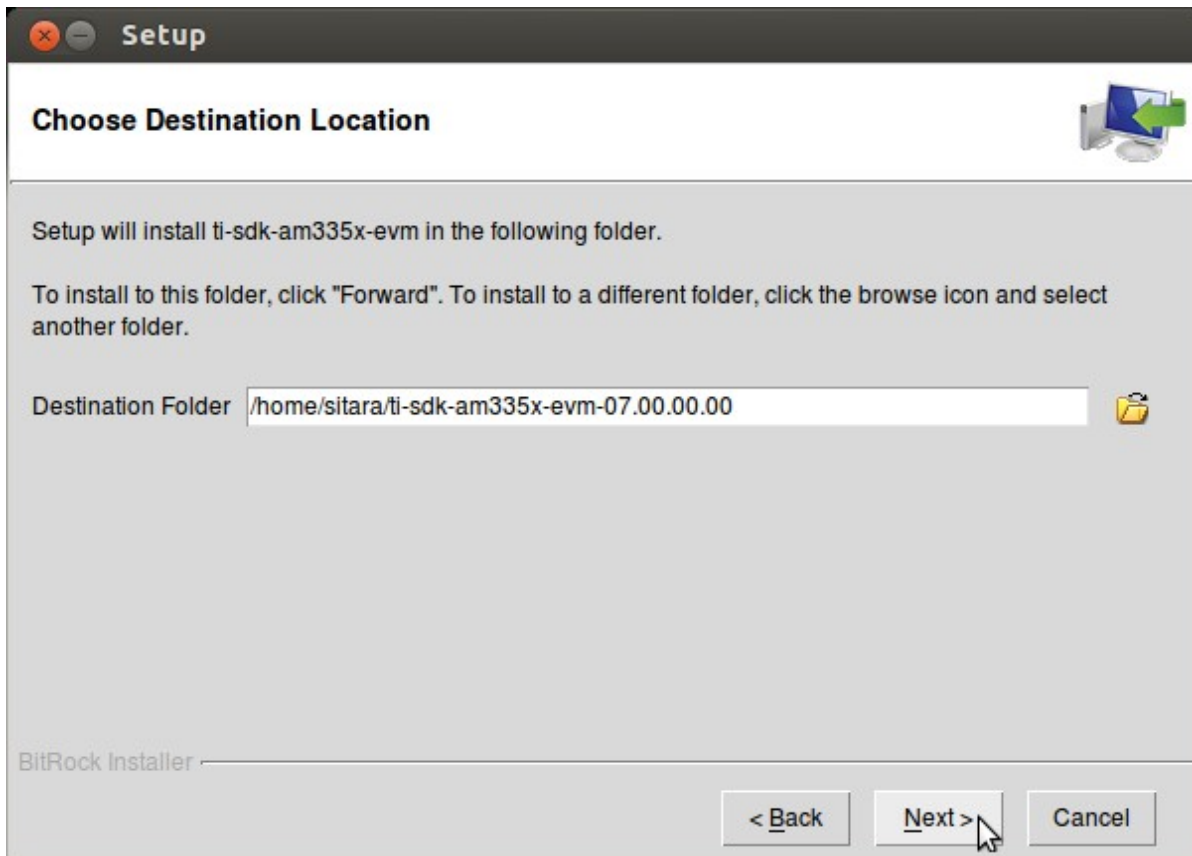
The user will be prompted for a location on where to put the ti-sdk. An example is given below.

### 3. Installation of software

The software is installed.

## Where to install the ti-sdk package

The default selection of where to install is the user's home directory. In this particular example the name of the user is user.



## Archived Versions

### Archived - Sitara Linux SDK Installer User's Guide

- Sitara SDK 6.0 - SDK Installer User's Guide (archived) <sup>[2]</sup>
- Sitara SDK 5.03 - SDK Installer User's Guide (archived) <sup>[3]</sup>
- Sitara SDK 4.01 - SDK Installer User's Guide (archived) <sup>[4]</sup>

## References

- [1] <http://www.ti.com/tool/linuxezsdk-sitara>
- [2] [http://processors.wiki.ti.com/index.php?title=Sitara\\_SDK\\_Installer&oldid=171489](http://processors.wiki.ti.com/index.php?title=Sitara_SDK_Installer&oldid=171489)
- [3] [http://processors.wiki.ti.com/index.php?title=Sitara\\_SDK\\_Installer&oldid=85510](http://processors.wiki.ti.com/index.php?title=Sitara_SDK_Installer&oldid=85510)
- [4] [http://processors.wiki.ti.com/index.php?title=Sitara\\_SDK\\_Installer&oldid=50935](http://processors.wiki.ti.com/index.php?title=Sitara_SDK_Installer&oldid=50935)

# Sitara Linux SDK Setup Script



Return to the [Sitara Linux Software Developer's Guide](#)

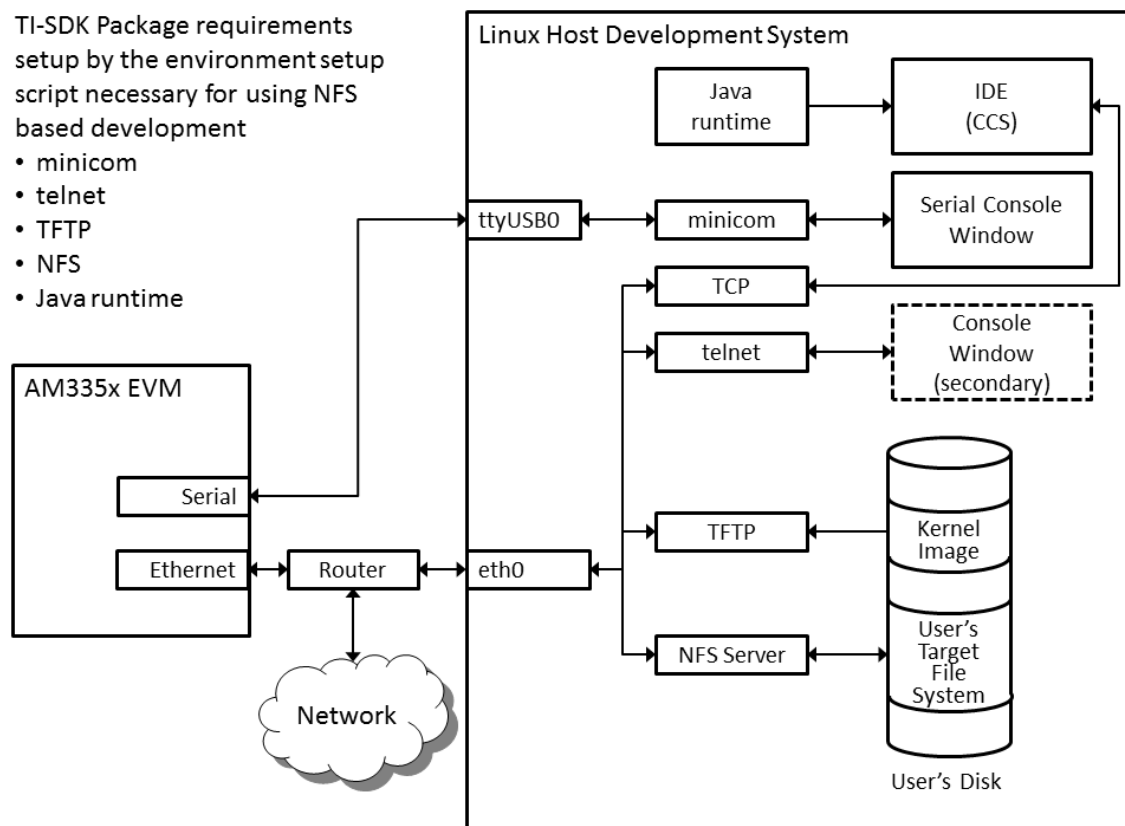
## Overview

After installation of the SDK on the Linux host, the setup script should be run to prepare the host for software development. Some of the tasks require administrator privileges. The script will prompt you when these administrator privileges are required. The setup script does the following things:

- Verification that the Linux host is the recommended Ubuntu LTS version
- Installation of required host packages
- Target FileSystem installation
- NFS setup
- TFTP setup
- Minicom setup
- uboot setup
- Load uboot script

TI-SDK Package requirements setup by the environment setup script necessary for using NFS based development

- minicom
- telnet
- TFTP
- NFS
- Java runtime



## BeagleBone Black Users

To run the SDK's setup scripts the following cables are required to be connected to the BeagleBone Black and your Linux PC. Please ensure both are connected before following any of the steps in this guide.

- USB Mini cable (included with BBBlack)
- FTDI Serial cable <sup>[1]</sup>

## Clearing the eMMC

The BeagleBone Black includes an eMMC device on it which comes pre-flashed with an Angstrom distribution. Because eMMC is the default boot mode for this board we need to prevent it from being able to boot by either removing or renaming the MLO.

To do this you will need to wipe out the MLO file stored in the eMMC.

To eliminate the MLO first boot up the board with the USB micro cable connected to the board and your PC. Once the Angstrom kernel loads your host will mount the eMMC boot partition on your Linux host under /media/BEAGLEBONE. You can then erase or rename the MLO file here. You can also login to the BeagleBone Black and rename or remove /boot/MLO (e.g., mv /boot/MLO /boot/OLDMLO).

Once the above steps are completed you can follow the remaining steps on this guide to execute the setup script.

## Restoring the eMMC

Instructions on restoring the eMMC can be found here <sup>[2]</sup>.

## How to run the setup script

The Setup Script is located in the Sitara SDK installation directory. By default, this directory has a name that has the form

"ti-sdk-<Hardware-Platform>-<Version>". Change to that ti-sdk install directory. Then run the script:

```
[host prompt]:~$ ./setup.sh
```

**NOTE - The Setup Script will first check to see if the user is running the recommended Ubuntu Long Term Support (LTS) distribution, if not it will exit. If the user is running on a different Ubuntu version or another Linux distribution, they are encouraged to modify the environment setup script to match their distribution.**

See which version of Ubuntu is currently supported here: [Start\\_your\\_Linux\\_Development](#)

## Detailed step by step description through the setup script

The following sections describe in more detail how to run the script and what is doing.

### Installation of Required Host Packages

This section will check to make sure you have the proper host support packages to allow you do the following tasks:

- telnet
- bring up menuconfig, the kernel configuration tool
- mounting filesystem via nfs
- tftp
- bring up minicom
- rebuild u-boot

If your host lacks any of the needed packages, they will automatically be installed in this step.

**Note! This command requires you to have administrator privileges (sudo access) on your host.**

The command below is an example of what this script is doing. The actual packages may vary for different releases:

```
[host prompt]:~$ sudo apt-get install xinetd tftpd nfs-kernel-server minicom build-essential libncurses5-dev uboot-mkimage autoconf automake
```

## Add to Dialout Group

**Note! This part requires you to have administrator privileges (sudo access)**

A unique step is required for users using Ubuntu 12.04+. By default the user does not have the proper permissions to access a serial device ( ex ttyS0, ttyUSB0, etc...). A user must be apart of a "dialout" group to access these serial device without root privileges.

During this step the script will check if the current Linux user is apart of the dialout group. If not the current Linux user will automatically be added to the dialout group.

The Linux user will still be required to use sudo when accessing the serial device until the user logs out and then logs back in.

## Target FileSystem Installation

This step will extract the target filesystem.

**Note! This part requires you to have administrator privileges (sudo access)**

The default locations is: /home/user/ti-sdk-amxx-evm-x.x.x.x/targetNFS

```
In which directory do you want to install the target filesystem?(if this directory does not exist it will be created)
[ /home/user/ti-sdk-amxx-evm-x.x.x.x/targetNFS ]
```

You can override the default location by typing in another location or by hitting <Enter> you can accept the default location. This can take a little time to untar and unzip the filesystem.

If you have run this script more than once and the filesystem already exists, you will be asked to either:

- rename the filesystem
- overwrite the filesystem
- skip filesystem extraction

(see actual prompt below)

```
/home/jlance/ti-sdk-am335x-evm-05.04.00.00/targetNFS already exists
(r) rename existing filesystem (o) overwrite existing filesystem (s) skip filesystem extraction
```

## NFS Setup

This step will allow you to export your filesystem which was extracted in the previous step.

**Note! This command requires you to have administrator privileges (sudo access) on your host.**

- This step adds the path to root filesystem from the previous step to the file /etc/exports on your host.
- The NFS kernel daemon is then stopped and then restarted to make sure the exported file system is recognized.

## TFTP Setup

This section will setup tftp access on your host.

**Note! This command requires you to have administrator privileges (sudo access) on your host.**

```
-----
Which directory do you want to be your tftp root directory?(if this directory does not exist it will be created for you)
[ /tftpboot ]
```



The default location is /tftpboot which is off of the root directory on your linux host and requires administrator privileges. You can hit <Enter> to select the default location or type in another path to override the default. Then the following task occur:

- A tftp config file is created for you on your host at /etc/xinetd.d/tftp
- The tftp server is stopped and then restarted to insure the changes are picked up.

If you have run this script more than once or the filename already exists, you will be asked to select one of the following options.

- rename the filesystem
- overwrite the filesystem
- skip filesystem extraction

## Minicom Setup

This step will set up minicom (serial communication application) for SDK development

```
Which serial port do you want to use with minicom?  
[ /dev/ttyS0 ]
```

For most boards, the default /dev/ttyS0 should be selected. For Beaglebone which has a USB-to-Serial converter, just hit enter and the proper serial port will be setup in a later step.

- A minicom configuration will be saved for you at /home/user/.minirc.dfl
- The old configuration if there was one will be saved at /home/user/.minirc.dfl.old

The configuration saved to /home/user/.minirc.dfl can be changed, see the Software Development Guide for more information.

NOTE: If you are using a USB-to-Serial converter, your port should be configured for /dev/ttyUSBx

## uboot Setup

This section will create the necessary u-boot commands to boot up your board based on your answers to the questions below.

The script will detect your ip address and display it. You can override the detected value by entering an alternate value. See sample below:

```
-----  
This step will set up the u-boot variables for booting the EVM.  
Autodetected the following ip address of your host, correct it if necessary  
[ xxx.xxx.xxx.xxx ]
```

Next you will be prompted where you prefer your kernel and file system to be located.

- Kernel location
  - TFTP - located on your Host in your designated /tftpboot directory
  - SD card - located in the 1st partition named "boot" of your SD card
- Filesystem location
  - NFS - located on your Host. The location is where the file system was extracted in an earlier step.
  - SD card - located on the 2nd partition named "rootfs" of your SD card.

Next if you have selected TFTP, you will be prompted which uImage you want to boot using TFTP. You will be given a list of existing uImage's and you can type one in from the list or hit <Enter> to select the default option. The default option will be the uImage corresponding to the SDK installation. This will be used in the next step to create the necessary u-boot options to boot up your device.

## Load uboot Script

This section creates a minicom script or a uEnv.txt file which will be used by u-boot to provide the necessary commands to boot up in the preferred configuration.

- For boards with straight serial connectors, a minicom script is created
- For boards like beaglebone with a USB-to-Serial configuration, then a uEnv.txt script is created and placed in the /boot partition of the SD card.

**NOTE: For devices which create a uEnv.txt, the device must already be booted up with the USB-to-Serial connector attached to the Host. Further the Host must recognize the boot and START\_HERE partitions.**

## References

[1] [http://circuitco.com/support/index.php?title=BeagleBone\\_Black\\_Accessories#Serial\\_Debug\\_Cables](http://circuitco.com/support/index.php?title=BeagleBone_Black_Accessories#Serial_Debug_Cables)

[2] [http://circuitco.com/support/index.php?title=Updating\\_The\\_Software](http://circuitco.com/support/index.php?title=Updating_The_Software)

# Sitara Linux SDK create SD card script

Return to the [Sitara Linux Software Developer's Guide](#)



## Overview

Starting with the Sitara Linux SDK version 05.04.00.00 there is now a script in the **<SDK INSTALL DIR>/bin** directory named **create-sdcard.sh**. The purpose of this script is to create SD cards for the following high-level use cases:

1. Create the **SD card using default images** from the Sitara Linux SDK
2. Create the **SD card using custom images**
3. Create the **SD card using partition tarballs** (This is not common and is used most often by board vendors)

The script will give you information about each step, but the following sections will go over the details for the use cases above and walk you through how to use the script as well.

## Common Steps

No matter which use case above that you are creating an SD card for the following steps are the same.

## Invoking the Script

The **create-sdcard.sh** script can be run from any location but must be run with **root** permissions. This usually means using the **sudo** command to start execution of the script. For example:

```
host# sudo <SDK INSTALL DIR>/bin/create-sdcard.sh
```

If you fail to execute the script with root permissions you will receive a message that root permissions are required and the script will exit.

## Select the SD Card Device

The first step of the script will ask you to select the drive representing the SD card that you want to format. In most cases your host root file system drive has been masked off to prevent damage to the host system. When prompted enter the device number corresponding to the SD card. For example if the output looks like:

```
Available Drives to write images to:
```

```
#  major    minor    size    name
1:   8        16    7761920 sdb
```

```
Enter Device Number:
```

You would enter **1** to select the *sdb* device

## Re-Partitioning the SD Card

Any partitions of the device that are already mounted will be un-mounted so that the device is ready for partitioning.

If the SD Card already has partition you will see a prompt like the following asking you if you would like to repartition the card. If the card was not already partitioned then this step will be skipped and you can move on to the next step.

```
Would you like to re-partition the drive anyways [y/n] :
```

- Options:

- y** - This will allow you to change the partitioning of the SD card. For example if you have a 3 partition card and want to create a 2 partition card to give additional storage space to the root file system you would select **y** here.

**NOTE:** This operation **WILL ERASE** the contents of your SD card

- n** - If the SD card already has the desired number of partitions then this will leave the partitioning alone. If you select **n** here skip on to the [Installing SD Card Content](#) section.

## Select Number of Partitions

You should now see a prompt like the following which will ask you how many partitions you want to create for the SD card

```
Number of partitions needed [2/3] :
```

- Options:

- 2** - This is the most common use case and will give the most space to the root file system.
- 3** - This case should only be used by board manufacturers making SD cards to go in the box with the EVM. This requires access to the partition tarballs used for Out-Of-Box SD cards. This option should be selected if you are going to follow the [SD card using partition tarballs](#) steps below

After selecting the number of partitions move on to the next section.

## Installing SD Card Content

After the SD card is partitioned you will be prompted whether you want to continue installing the file system or safely exit the script.

- Options:
  - **y** - Selecting yes here will begin the process of installing the SD card contents. This operation **WILL ERASE** any existing data on the SD card. Refer to one of the following sections for additional instructions depending on which use case you are creating an SD card for
    - Create the [SD card using default images](#)
    - Create the [SD card using custom images](#)
    - Create the [SD card using partition tarballs](#)
  - **n** - Selecting no here will allow you to have partitioned your card but will leave the partitions empty.

## SD Card Using Default Images

The purpose of this section is to cover how to use the **create-sdcard.sh** script to populate an SD card that can be used to boot the device using the default images that ship with the Sitara Linux SDK.

**NOTE:** For devices like AM180x see the [AM180x SD card boot](#) section below for information on booting that device from an SD card. This script will create a card suitable for the root file system but additional steps are required to load the boot loaders from the SD card.

## Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the [Common Steps](#) section above.

## Choose Install Pre-built Images

You should now see a prompt like:

```
#####

Choose file path to install from

1 ) Install pre-built images from SDK
2 ) Enter in custom boot and rootfs file paths

#####

Choose now [1/2] :
```

You should choose option **1** to create an SD card using the pre-built images from the SDK.

If you executed this script from within the SDK then the script can determine the SDK path automatically and will start copying the contents to the SD card. Once the files are copied the script will exit.

If you executed the script from outside of the SDK (i.e. you copied it to some other directory and executed it there) please see the next section.

## Enter SDK Path

In the case that the script was invoked from a directory without the SDK installation in the path, i.e. the script was copied to your home directory and executed there you may see a prompt like

```
no SDK PATH found
Enter path to SDK :
```

Enter the path to the SDK installation directory here. For example if the SDK was installed into the home directory of the *sitara* user the path to enter would be `/home/sitara/ti-sdk-<machine>-<version>`. You will be prompted to confirm the installation directory. The SD card will then be created using the default images and the script will exit when finished.

## SD Card Using Custom Images

Often times you will use TFTP and NFS during development to transfer your kernel images and boot your root file systems respectively. Once you are done with your development you may want place these images onto an SD card so that they can be used stand-alone without requiring a network connection to a server.

## Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the [Common Steps](#) section above.

## Choose Custom Images

You should now see a prompt like:

```
#####

Choose file path to install from

1 ) Install pre-built images from SDK
2 ) Enter in custom boot and rootfs file paths

#####

Choose now [1/2] :
```

Select option **2** to create an SD card with your custom images.

## Select Boot Partition

You will now be prompted to provide a path to the location of the boot partition files. The prompt will explain the requirements of the files to be placed at the path, but the basic options are:

1. Point to a tarball containing all of the files you want placed on the boot partition. This would include the boot loaders and the kernel image as well as any optional files like uEnv.txt
2. Point to a directory containing the files for the boot partition like those in the first option.

The script is intelligent enough to recognize whether you provided a tarball or a directory path and will copy the files accordingly. You will be given a list of the files that are going to be copied and given the option to change the path if the list of files is not correct.

## Select Root Partition

You will now be prompted to provide a path to the location of the root file system partition files. The prompt will explain the requirements of the files to be placed at the path, but the basic options are:

1. Point to a tarball of the root file system you want to use
2. Point to a directory containing the root file system such as an NFS share directory.

The script is intelligent enough to recognize whether you provided a tarball or a directory path and will copy the files accordingly. You will be given a list of the files that are going to be copied and given the option to change the path if the list of files is not correct.

## SD Card Using Partition Tarballs

This option is meant for board vendors to create SD cards to go in the box with the EVM. It requires access to the three tarballs representing the partitions of the SD card shipped with the EVM.

## Prerequisites

1. The Sitara Linux SDK is installed on your host system
2. The SD card you wish to create is inserted into the host system and has a size sufficiently large to hold at least the bootloaders, kernel, and root file system.
3. You have started running the script as detailed in the [Common Steps](#) section above.

## Provide Tarball Location

After the SD card has been partitioned you will be prompted to

```
Enter path where SD card tarballs were downloaded :
```

Point to the directory containing the following tarball files:

- **boot\_partition.tar.gz**
- **rootfs\_partition.tar.gz**
- **start\_here\_partition.tar.gz**

The script will show you the contents of the directory given and ask you to verify that the tarballs are present in that directory. The SD card will then be populated with the contents of the tarballs and be ready for inclusion in the box with the EVM.

## AM180x SD Card Boot

For the AM180x device the boot ROM on most silicon versions does not support reading the boot loaders from the SD card vfat partition like other devices. For these devices you have the following options:

1. Flash the boot loaders into persistent storage on the device and use the SD card for the root file system by creating an SD card as detailed above. In this case the SD card will have a boot partition but the boot loaders on that partition will not be used. The kernel image can still be loaded from the SD card boot partition.
  - Please refer to the [AM18x Flash Tool User's Guide](#) for instructions on how to flash the boot loaders into SPI Flash
2. It is also possible to use the tool described in the [AM18x Flash Tool User's Guide](#) to flash only the UBL into the SPI Flash and then to place u-boot.bin in a special partition of the SD card.
  - To learn how to put u-boot.bin on the SD card please see [Creating bootable SD card for OMAP-L138 EVM board](#) section on the wiki.
  - **NOTE:** Use of this method is incompatible with the `create-sdcard.sh` script.
3. For information on using the native MMC/SD0 boot mode that was introduced in OMAP-L13x/AM18xx Silicon Rev 2.1 (Boot ROM Rev D800K008) please see this page

## Archived Versions

- [Sitara Linux SDK 05.07](#) <sup>[1]</sup>

## References

[1] [http://processors.wiki.ti.com/index.php?title=Sitara\\_Linux\\_SDK\\_create\\_SD\\_card\\_script&oldid=114505](http://processors.wiki.ti.com/index.php?title=Sitara_Linux_SDK_create_SD_card_script&oldid=114505)

---

# Article Sources and Contributors

**Sitara Linux SDK Getting Started Guide** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171664> *Contributors:* RonB

**Sitara Linux SDK Supported Platforms and Versions** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171525> *Contributors:* RonB

**Sitara Linux SDK Software Stack** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171455> *Contributors:* RonB

**Sitara Linux SDK Creating a SD Card with Windows** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171444> *Contributors:* RonB, RonBirkett

**How to Build a Ubuntu Linux host under VMware** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171583> *Contributors:* Alanc, Cem8101, Cjc211, Fcooper, Gregturne, Jefflance01, Kevinsc, Pprakash, RonB, SiddharthHeroor

**Sitara SDK Installer** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171519> *Contributors:* Andreas, Cem8101, Kevinsc, Mike Tadyshak, Nangelou, RonB

**Sitara Linux SDK Setup Script** *Source:* <http://processors.wiki.ti.com/index.php?oldid=171512> *Contributors:* Fcooper, Kevinsc, RonB, TimHarron

**Sitara Linux SDK create SD card script** *Source:* <http://processors.wiki.ti.com/index.php?oldid=169099> *Contributors:* BradGriffis, Cem8101, Jefflance01, Kevinsc, Nangelou



# Image Sources, Licenses and Contributors

**Image:TIBanner.png** *Source:* <http://processors.wiki.ti.com/index.php?title=File:TIBanner.png> *License:* unknown *Contributors:* Nsnehaprabha

**Image:AM335x Development Environment.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:AM335x\\_Development\\_Environment.png](http://processors.wiki.ti.com/index.php?title=File:AM335x_Development_Environment.png) *License:* unknown *Contributors:* RonB

**Image:Sitara linux stack.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Sitara\\_linux\\_stack.png](http://processors.wiki.ti.com/index.php?title=File:Sitara_linux_stack.png) *License:* unknown *Contributors:* Hazeljojo, Kevinsc

**File:7zip to extract image.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:7zip\\_to\\_extract\\_image.png](http://processors.wiki.ti.com/index.php?title=File:7zip_to_extract_image.png) *License:* unknown *Contributors:* RonB, RonBirkett

**File:Win32 Disk Imager Extracting.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_Extracting.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_Extracting.png) *License:* unknown *Contributors:* RonB, RonBirkett

**File:7zip image file extracted.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:7zip\\_image\\_file\\_extracted.png](http://processors.wiki.ti.com/index.php?title=File:7zip_image_file_extracted.png) *License:* unknown *Contributors:* RonB, RonBirkett

**File:Win32 Disk Imager open.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_open.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_open.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 disk imager select a disk image.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_disk\\_imager\\_select\\_a\\_disk\\_image.png](http://processors.wiki.ti.com/index.php?title=File:Win32_disk_imager_select_a_disk_image.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 Disk Imager select disk.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_select\\_disk.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_select_disk.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 Disk Imager write disk.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_write\\_disk.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_write_disk.png) *License:* unknown *Contributors:* RonBirkett

**File:Light\_bulb\_icon.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Light\\_bulb\\_icon.png](http://processors.wiki.ti.com/index.php?title=File:Light_bulb_icon.png) *License:* unknown *Contributors:* DanRinkes, PagePusher

**File:Win32 disk imager Confirm overwrite.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_disk\\_imager\\_Confirm\\_overwrite.png](http://processors.wiki.ti.com/index.php?title=File:Win32_disk_imager_Confirm_overwrite.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 Disk Imager writing to disk.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_writing\\_to\\_disk.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_writing_to_disk.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 Disk Imager Complete.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_Complete.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_Complete.png) *License:* unknown *Contributors:* RonBirkett

**File:Win32 Disk Imager exit.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win32\\_Disk\\_Imager\\_exit.png](http://processors.wiki.ti.com/index.php?title=File:Win32_Disk_Imager_exit.png) *License:* unknown *Contributors:* RonBirkett

**File:Win7 eject disk.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win7\\_eject\\_disk.png](http://processors.wiki.ti.com/index.php?title=File:Win7_eject_disk.png) *License:* unknown *Contributors:* RonBirkett

**File:Win7 eject disk detail.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win7\\_eject\\_disk\\_detail.png](http://processors.wiki.ti.com/index.php?title=File:Win7_eject_disk_detail.png) *License:* unknown *Contributors:* RonBirkett

**File:Win7 device can be safely removed.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Win7\\_device\\_can\\_be\\_safely\\_removed.png](http://processors.wiki.ti.com/index.php?title=File:Win7_device_can_be_safely_removed.png) *License:* unknown *Contributors:* RonBirkett

**Image:VMwareNetwork.PNG** *Source:* <http://processors.wiki.ti.com/index.php?title=File:VMwareNetwork.PNG> *License:* unknown *Contributors:* Cjc211, Gregturne

**Image:Vmware sh1.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh1.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh1.jpg) *License:* unknown *Contributors:* Kevinsc

**Image:Vmware sh2.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh2.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh2.jpg) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Sitara-Linux-VMWare-removable-devices2.png** *Source:* <http://processors.wiki.ti.com/index.php?title=File:Sitara-Linux-VMWare-removable-devices2.png> *License:* unknown *Contributors:* Cem8101

**Image:Sitara\_Linux\_VMWare\_connect\_bbb.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Sitara\\_Linux\\_VMWare\\_connect\\_bbb.png](http://processors.wiki.ti.com/index.php?title=File:Sitara_Linux_VMWare_connect_bbb.png) *License:* unknown *Contributors:* Fcooper

**Image:Vmware sh4.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh4.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh4.jpg) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Vmware sh6.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh6.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh6.jpg) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Vmware sh7.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh7.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh7.jpg) *License:* unknown *Contributors:* Kevinsc

**Image:Vmware sh8.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_sh8.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_sh8.jpg) *License:* unknown *Contributors:* Kevinsc

**Image:Network proxy setup.JPG** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Network\\_proxy\\_setup.JPG](http://processors.wiki.ti.com/index.php?title=File:Network_proxy_setup.JPG) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Network proxy preferences.JPG** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Network\\_proxy\\_preferences.JPG](http://processors.wiki.ti.com/index.php?title=File:Network_proxy_preferences.JPG) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Network proxy password.JPG** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Network\\_proxy\\_password.JPG](http://processors.wiki.ti.com/index.php?title=File:Network_proxy_password.JPG) *License:* unknown *Contributors:* Kevinsc

**Image:Vmware kb1.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_kb1.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_kb1.jpg) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:Vmware kb3.jpg** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Vmware\\_kb3.jpg](http://processors.wiki.ti.com/index.php?title=File:Vmware_kb3.jpg) *License:* unknown *Contributors:* Cjc211, Kevinsc

**Image:VMWareDisk.png** *Source:* <http://processors.wiki.ti.com/index.php?title=File:VMWareDisk.png> *License:* unknown *Contributors:* Gregturne

**Image:AM335x SDL dir selection.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:AM335x\\_SDL\\_dir\\_selection.png](http://processors.wiki.ti.com/index.php?title=File:AM335x_SDL_dir_selection.png) *License:* unknown *Contributors:* RonB

**File:Linux Host Development System.png** *Source:* [http://processors.wiki.ti.com/index.php?title=File:Linux\\_Host\\_Development\\_System.png](http://processors.wiki.ti.com/index.php?title=File:Linux_Host_Development_System.png) *License:* unknown *Contributors:* RonB