Working with Yocto to Build Linux

Embedded Artists

Embedded Artists AB

Jörgen Ankersgatan 12
SE-211 45 Malmö
Sweden

http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Document Revision History

Revision	*Date*	*Description*
A	2015-09-30	First release
B	2015-11-12	- Added instructions for iMX6 UltraLite COM Board - Added Section 5.3
C	2016-01-12	- Updated Table 1 with information about new 3.14.52 branch - Updated section 4.3 with new instructions
D	2016-02-15	- Updated section 5.1
E	2016-04-06	- Added new machine configuration for iMX7 Dual uCOM Board in section 4.2
F	2016-04-21	- Added new machine configuration for iMX6 DualLite COM Board in section 4.2
G	2016-08-30	- Added new machine configuration for iMX7 Dual COM Board in section 4.2
H	2016-10-20	- Linux **4.1.15** is now supported. Removed description of working with 3.14.28.
I	2016-12-21	- Added section 8.1 describing how to build and use meta-toolchain. - Added section 9.1 describing how to put the root file system on an SD card
J	2017-05-19	- Updated section 3.4 – New distribution is now available. Linux version is still 4.1.15. U-boot has been updated to **2016.03**.
K	2018-01-29	- Removed chapter ”Ubuntu 14.04 Virtuall Machine” since it is out-dated. Use instructions in chapter 7 instead. - Updated section 3.4 with 4.9.11 branch (Linux kernel **4.9.11** is now supported) - Updated section 8.1 - Added sections 9.2 and 9.3
L	2018-11-20	- Linux 4.9.123 and u-boot 2017.03 now supported on ea-yocto-base branch ea-4.9.123. Section 3.4 updated.

2 Introduction

This document provides you with step-by-step instructions to setup the Yocto build system, build boot loaders, Linux kernel and file system for the Embedded Artists i.MX6 based COM boards.

Additional documentation you might need is.

- *NXP Yocto Project User's Guide* – The document is available in the Linux bundle found at the software and tools section on NXP's website.
- The *Getting Started* document for the board you are using.
- Yocto Project Quick Start
- Yocto Project Reference Manual
- Yocto Training – Instructions for using Yocto at the NXP community site

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

`Constant width text` – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a
document.
```

3 Linux Host Setup

3.1 Introduction

The Yocto build system requires a Linux host machine. You can either run this host as a standalone / native computer or as a virtual machine on, for example, a Microsoft Windows PC. The minimum available hard disk space is 50 GB, but it is recommended that the host machine has at least 120 GB to be able to build the largest image / distribution.

Several Linux distributions are supported by the Yocto project. Please refer to the Supported Linux Distributions section in the Yocto reference manual for a complete list.

The instructions in this document have been tested on an Ubuntu 14.04 and a Lubuntu 14.04 distribution.

- Chapter 7 describes how to install and setup Lubuntu 14.04 as a virtual machine. A VMWare appliance is not downloaded in this case. Instead an ISO image of Lubuntu is downloaded and installed into a newly created virtual machine.

3.2 Required Packages

The Yocto Project requires several packages to be installed on the host machine. If you are using any of the distributions in section 3.1 follow the instructions below. If you, however, are using another distribution refer to the Required Packages for the Host Development System section in the Yocto reference manual.

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo
gcc-multilib build-essential chrpath socat

$ sudo apt-get install libsdl1.2-dev xterm sed cvs subversion
coreutils texi2html docbook-utils python-pysqlite2 help2man make
gcc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev
mercurial autoconf automake groff curl lzop asciidoc

$ sudo apt-get install u-boot-tools
```

3.3 Install the `repo` tool

The `repo` tool has been developed to make it easier to manage multiple Git repositories. Instead of downloading each repository separately the `repo` tool can download all with one instruction. Download and install the tool by following the instructions below.

1. Create a directory for the tool. The example below creates a directory named `bin` in your home folder.

```
$ mkdir ~/bin
```

2. Download the tool

```
$ curl http://commondatastorage.googleapis.com/git-repo-
downloads/repo > ~/bin/repo
```

3. Make the tool executable

```
$ chmod a+x ~/bin/repo
```

4. Add the directory to the PATH variable. The line below could be added to your .bashrc file so the path is available in each started shell/terminal

```
$ export PATH=~/bin:$PATH
```

3.4 Download Yocto recipes

The Yocto project consists of many recipes used when building an image. These recipes come from several repositories and the repo tool is used to download these repositories.

In step 3 below a branch must be selected of the ea-yocto-base repository. The table below lists the available branches.

Branch name	Description
ea-4.9.123	u-boot: 2017.03. Linux: 4.9.123.
ea-4.9.11_1.0.0	u-boot: 2016.03. Linux: 4.9.11.
ea-4.1.15_2.0.0	u-boot: 2016.03. Linux: 4.1.15.
ea-3.14.52_1.1.0	u-boot: 2015.04. Linux: 3.14.52.

Table 1 - ea-yocto-base branches

1. Create a directory for the downloaded files (`ea-bsp` in the example below)

```
$ mkdir ea-bsp
$ cd ea-bsp
```

2. Configure Git if you haven't already done so. Change "Your name" to your actual name and "Your e-mail" to your e-mail address.

```
$ git config --global user.name "Your name"
$ git config --global user.email "Your e-mail"
```

3. Initialize repo. The file containing all needed repositories is downloaded in this step. Change <selected branch> to a branch name according to Table 1.

```
$ repo init -u https://github.com/embeddedartists/ea-yocto-base -b
<selected branch>
```

4. Start to download files

```
$ repo sync
```

All files have now been downloaded into the `ea-bsp` directory. Most of the files will actually be available in the sub-directory called `sources`.

4 Building Images

Yocto is using the BitBake tool to generate complete Linux images/distributions, that is, all needed to boot and run a Linux system. This is typically boot loader(s), Linux kernel, and root file system with selected utilities and applications.

4.1 Available Images

The recipes that have been downloaded contain many different images. The table below describe only a few of the images that are available.

Image name	Description
core-image-minimal	A small image allowing a device to boot
core-image-base	A console only image that fully supports the target device
fsl-image-gui	Build an image with GUI support, but without Qt content. Works with X11, DirectFB, frame buffer and Wayland backends.
fsl-image-qt5	Build a Qt5 image. Works with X11, Frame buffer, and Wayland backends.
fsl-image-mfgtool-initramfs	Build u-boot, kernel and file system that can be used by NXP's Manufacturing tool.
meta-toolchain	Builds an installable toolchain (cross-complier)
meta-toolchain-qt5	Builds toolchain/SDK for Qt5. Can be used when developing Qt5 applications.

Usually the name of the file that defines an image contains the string "*image*". You can search for such files using the `find` command. Please note that not all images follow this naming convention. The toolchain images are images that doesn't contain "image" in the name.

```
$ cd ~/ea-bsp/sources
$ find -name *image*.bb
```

An alternative way to find images is to use the `bitbake-layers` command.

```
$ bitbake-layers show-recipes | grep image
```

4.2 Machine Configurations

A machine configuration must be specified before a build can be started.

The table below contains the machine configurations available for Embedded Artists boards. It is also possible to find the configuration files in the directory `~/ea-bsp/sources/meta-ea/conf/machine`.

Machine	Description
imx6sxea-com	Machine configuration for Embedded Artists i.MX 6 SoloX COM Board / Kit
imx6qea-com	Machine configuration for Embedded Artists i.MX 6 Quad COM Board / Kit
imx6dlea-com	Machine configuration for Embedded Artists i.MX 6 DualLite COM Board / Kit

imx6ulea-com	Machine configuration for Embedded Artists i.MX 6 UltraLite COM Board / Kit
imx7dea-ucom	Machine configuration for Embedded Artists i.MX 7 Dual uCOM Board / Kit
imx7dea-com	Machine configuration for Embedded Artists i.MX 7 Dual COM Board / Kit

4.3 Initialize Build

Before starting the build it must be initialized. In this step the build directory and local configuration files are created.

A distribution must be selected when initializing the build, see section 4.3.1 for available distributions.

In the example below the machine `imx6sxea-com`, the build directory `build_dir` and the `fsl-imx-fb` distribution is selected.

```
$ DISTRO=fsl-imx-fb MACHINE=imx6sxea-com source ea-setup-
release.sh -b build_dir
```

4.3.1 Distro configurations

When initializing a build a distribution is specified. Several different are supported as listed in the table below.

DISTRO	Description
fsl-imx-fb	Linux Frame buffer – no X11 or Wayland
fsl-imx-x11	Only X11 (X Window System) graphics
fsl-imx-wayland	Wayland Weston graphics
fsl-imx-xwayland	Wayland graphics and X11. X11 applications using EGL are not supported

4.3.2 Restart a Build

If you need to restart a build in a new terminal window or after a restart of the host computer you don't need to run the `ea-setup-release.sh` script again. Instead you run the `setup-environment` script. If you don't run the `setup-environment` script you won't have access to needed tools and utilities, such as `bitbake`.

```
$ source setup-environment build_dir
```

4.4 Starting the Build

Everything has now been setup to start the actual build. The example below shows how the `core-image-minimal` image is being built. Please note that depending on the capabilities of your host computer building an image can take many hours.

```
$ bitbake core-image-minimal
```

When the build has finished the images will be available in the directory specified below. Please note that this directory will be different if you are using another build directory or machine configuration.

`~/ea-bsp/build_dir/tmp/deploy/images/imx6sxea-com`.

Go to chapter 5 for instructions of how to deploy images to the target hardware.

4.5 Bitbake Options

This section contains a few examples of how to use bitbake. This is by no means a complete list of all available bitbake options, but instead a list of examples that you might find useful.

4.5.1 Clean Build for a Specific Image/Recipe

The "-c" option executes a specific task for an image or recipe. In the example below a previous build of the u-boot boot loader is cleaned.

```
$ bitbake -c cleansstate u-boot-imx
```

To build u-boot after it has been cleaned just specify the image name u-boot-imx.

```
$ bitbake u-boot-imx
```

4.5.2 Kernel Configuration

If you would like to check or change the Linux kernel configuration you can start the Linux configuration tool using the option below.

```
$ bitbake -c menuconfig linux-imx
```

If you do any changes to the configuration you can generate a file (`fragment.cfg`) containing the differences between the original configuration and your changes.

```
$ bitbake -c diffconfig linux-imx
```

The Embedded Artists Yocto layer (meta-ea) is using a `fragment.cfg` file to make changes in the Linux kernel. See `~/ea-bsp/sources/meta-ea/recipes-kernel/linux` for an example.

4.5.3 Show Yocto Layer Append Dependencies

One nice feature with Yocto is the ability to extend an already existing recipe. This is done by so called bbappend files. The Embedded Artists layer (meta-ea) is constructed this way, that is, it is appending to existing recipes such as u-boot (u-boot-imx) and Linux kernel (linux-imx).

When creating your own append files it can be useful to get a list of "appends" that are considered to be active for you build.

```
$ bitbake-layers show-appends
```

5 Deploying Images

5.1 Manufacturing Tool

NXP's Manufacturing Tool (MFGTool) can be used to write images to the board. This tool is sending files and instructions over USB and the board must be set in OTG boot mode for it to work.

At the moment the tool is only available for Microsoft Windows and a version which has been prepared for Embedded Artists boards is available on the Embedded Artists support site for the board you are using.

5.1.1 Download the Tool

Download the zip file containing the manufacturing tool from http://imx.embeddedartists.com/

Unpack this zip file somewhere on your computer running Microsoft Windows. Below is a description of some of the content in the zip file.

- `mfgtool` (root): Contains the actual tool as well as vbs files which can be used to run a specific download configuration.
- `mfgtool/Document`: Contains documentation for the manufacturing tool. This documentation has been written by NXP.
- `mfgtool/Profiles/Linux/OS Firmware/ucl2.xml`: This file contains the actual download configurations.
- `mfgtool/Profiles/Linux/OS Firmware/files`: Contains pre-compiled versions of images. The tool will look in this directory when selecting images to download to the board.

5.1.2 Prepare hardware

Begin by reading the *Getting Started* document for the board you are using. This document shows how to setup the board and also gives an overview of the hardware.

The next step is to put the board into OTG boot mode. If you have an early version of the iMX6 SoloX Developer's Kit, that is, a version with a DIP switch mounted as shown in Figure 2, read section 5.1.4 for instructions. Read section 5.1.3 if you have another iMX based developer's kit or a new version of the iMX6 SoloX Developer's kit.

5.1.3 OTG boot mode – J2 jumper

To download images using the manufacturing tool the board must be put into OTG boot mode.

This is accomplished by closing the J2 jumper on the Carrier board; see Figure 1 to locate the jumper. Please note that in the figure the jumper is in open state which means that the COM board will boot from eMMC.

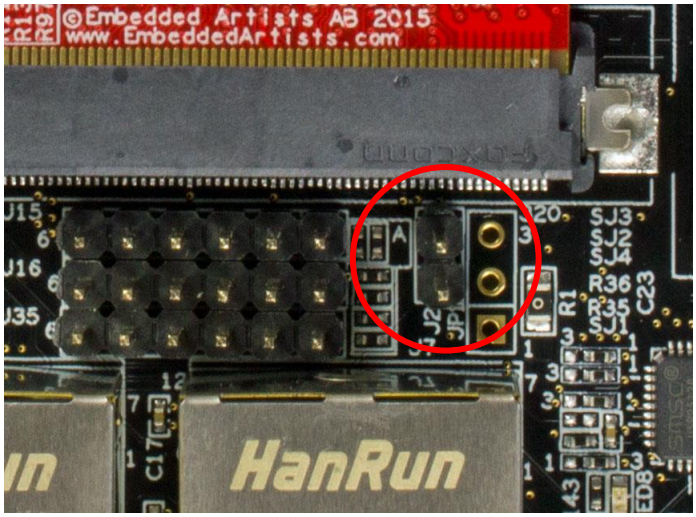

Figure 1 - J2 jumper (opened state)

5.1.4 OTG boot mode – DIP switches

The first version of the iMX6 SoloX COM boards had boot jumpers (DIP switches) mounted on them, see Figure 2. If you have such a COM board you need to set the boot jumpers as described below to force it into OTG boot mode instead of closing jumper J2 as described above.

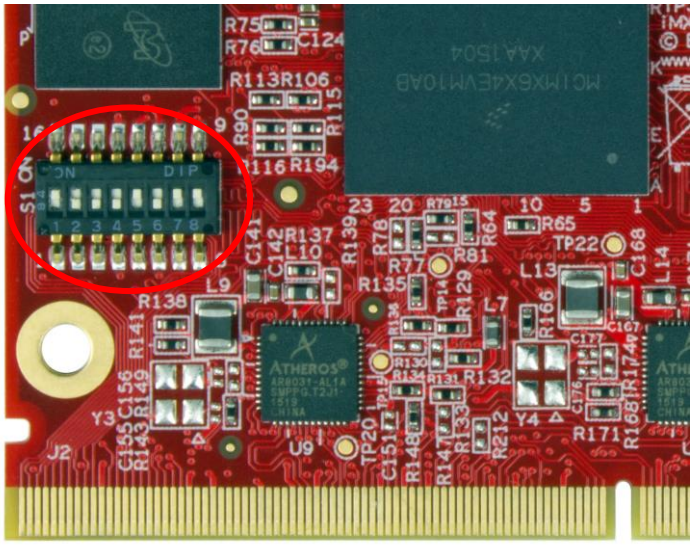

Figure 2 - DIP switch on iMX6 SoloX COM board

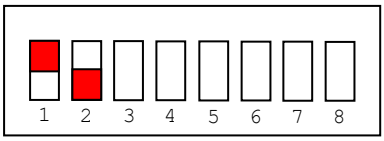

Figure 3 – Boot jumper setting for USB OTG

1. Set the boot jumpers in OTG boot mode as shown in Figure 3. First jumper in the *up* position and the second jumper in *down* position. Please note that the orientation of the DIP switch component can be different on different COM boards. Make sure you are changing the correct jumper by looking at the number by the jumper.
2. Make sure a USB cable is connected between the board (micro-B connector on carrier board) and your PC

3. Reset the board

Note: When you want to boot the software from eMMC you have to reverse the setting, that is, first jumper in *down* position and second jumper in *up* position.

5.1.5 Configurations

Several configurations of the tool have been prepared in order to help you download specific images. Shortcuts to these configurations are available as vbs files in the root of the MFGTool directory. All you need to do is to double-click on one of these files and the manufacturing tool will start.

- `ea-com-emmc_bootloaders.vbs` – will install only the bootloaders. This should only be used if you want to restore the bootloaders or download your own bootloaders to the board.
- `ea-com-emmc_kernel.vbs` – will install kernel and dtb file. This should only be used if you want to update the kernel or dtb file.
- `ea-com-emmc_full_tar.vbs` – will install bootloaders, Linux kernel and root file system. The root file system will be installed from a tar.bz2 file.
- `ea-com-emmc_update_rootfs.vbs` – will only download the root file system (the ext3 file) to the board.

5.1.6 Download Your Own Images

The simplest way to download your own images is to replace the existing file(s) with your file(s). If you keep the file names intact the vbs files will download your version of the file.

In many cases you would, however, like to keep the pre-compiled versions of the files and just add your own files. You could then copy an existing vbs file that is closest to what you want to do, for example, copy the "...`update_rootfs.vbs`" file if you want to update only the root file system. When you open this copy of the file you can see that several options are sent to the manufacturing tool. For example, one option is called `board` and another is called `rootfs`. These options are used in the configuration file (`ucl2.xml`) to construct the actual file names of the files the tool is accessing.

The name of the file for the root file system is, for example, constructed by using both the `rootfs` and `board` options. This is how it looks in the ucl2.xml file: `files/%rootfs%-%board%.rootfs.tar.bz2`.

One more alternative is to create a new configuration. In this case you need to open the ucl2.xml file and then copy an existing configuration (the LIST tag and all its children), give it a new name (change the name attribute) and then modify the instructions so your images are downloaded.

5.1.7 Run the Tool

Double click on one of the vbs files to start the manufacturing tool. If the tool can find the board it will write "HID-compliant device" in the status field, see Figure 4 below. If it cannot find the board it will write "No Device Connected".

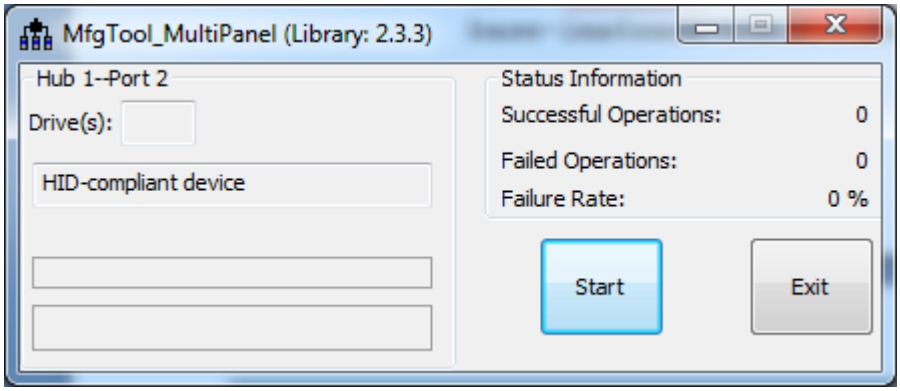

Figure 4 - Manufacturing Tool

Click the Start button to start the download of files. If all operations are successful the progress bars will turn green, see Figure 5. Click the Stop button and then Exit to close the manufacturing tool. If an operation fails the progress bars will turn red. In this case it can be helpful to have a look at the log `MfgTool.log` which is found in the same directory as the manufacturing tool.

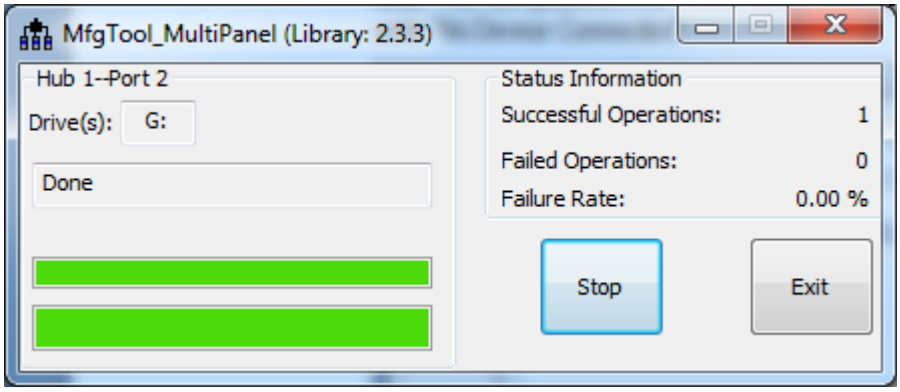

Figure 5 Manufacturing Tool successful download

5.2 From within u-boot

An alternative to the manufacturing tool is to use the u-boot bootloader. This bootloader is usually already programmed onto the board when delivered from Embedded Artists.

U-boot supports loading files from many sources (network, SD card, USB memory stick), but this section will only describe how to load from a USB memory stick. Copy your files to a USB memory stick and then insert this into the USB Host Connector on the Carrier Board, see Figure 6.

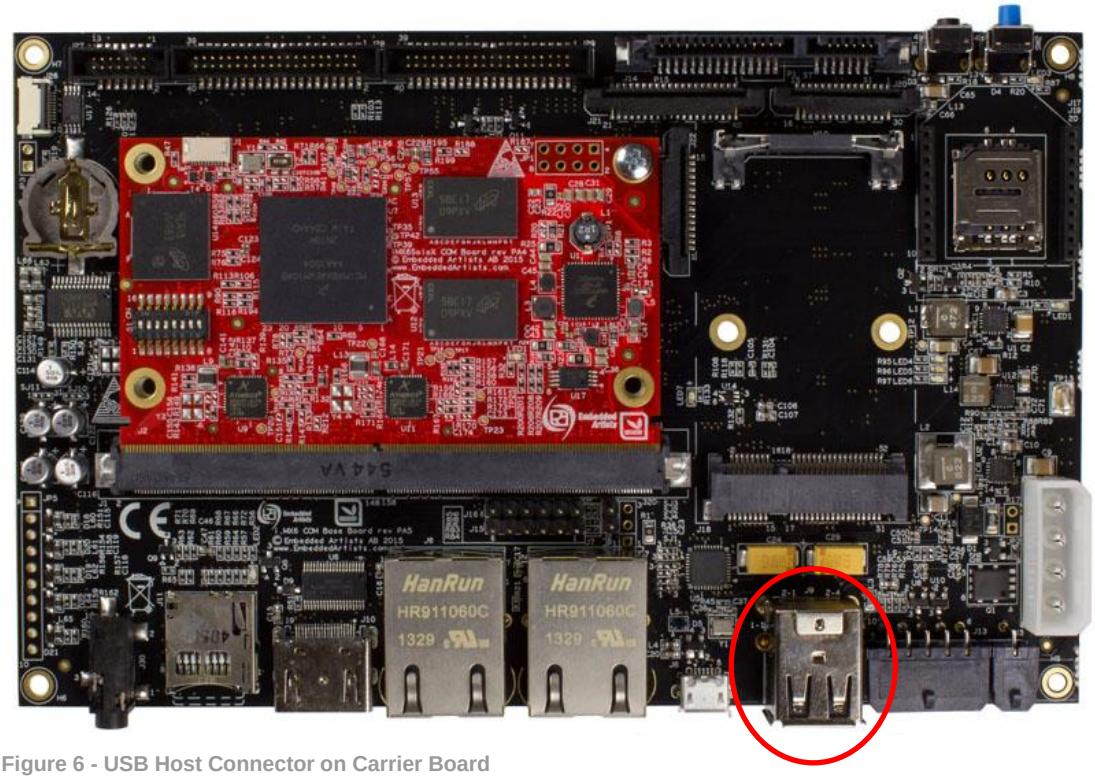

Figure 6 - USB Host Connector on Carrier Board

5.2.1 Find the USB Memory Stick

Follow the steps described in the *Getting Started* document for the board you are using to setup the board. You need to have a terminal/console application such as Tera Term connected to the board. When the board boots stop u-boot from continuing with the boot (by default you have 3 seconds to hit any key so the autoboot stops).

1. Enable the USB interface

```
=> usb start
```

2. Find attached storage devices. In the example below only one device is attached and will therefore get device index zero (‘0’). You need to know this index when accessing the device in later steps.

```
=> usb storage
  Device 0: Vendor: Kingston Rev: 1.00 Prod: DataTraveler R
            Type: Removable Hard Disk
            Capacity: 959.0 MB = 0.9 GB (1964032 x 512)
```

3. List the content on the USB device to see that your files are available on the device. The zero in the instructions below is the device index.

```
=> fatls usb 0
```

5.2.2 Load the Root File System

To replace only the root file system use the ext3 image, that is the file with extension ext3. You must have followed the instructions in section 5.2.1 to make sure the USB interface is enabled.

Important: Files are loaded into RAM memory which means that they must be smaller than the memory on the board.

1. Load the ext3 image into memory. In the example below USB device 0 is accessed and the file `rootfs.ext3` is loaded to address 0x83000000. The size of the loaded file is **79691776** bytes. This information is **important** and needed in step 3. The size may be different in your case.

```
=> fatload usb 0 0x83000000 rootfs.ext3
reading rootfs.ext3
79691776 bytes read in 3466 ms (21.9 MiB/s)
```

2. Find the offset to the file system partition. If you are using the default setup from Embedded Artists the root file system will be in partition 2 which as shown in the example below is available at offset 24576 (0x6000). Partition 1 contains the Linux kernel and device tree (dtb) files.

```
=> mmc part

Partition Map for MMC device 1  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     8192            16384           00027f23-01     0c
  2     24576           155648          00027f23-02     83
```

3. Write the file to eMMC. The eMMC memory is available on mmc device 1 and as found in the previous step partition 2 starts at offset 0x6000. The amount of data to write to the mmc device must be given in number of blocks where each block is 512 bytes. The value must be written as a hexadecimal value. 79691776 / 512 = 155648 ➔ 0x26000.

```
=> mmc write 0x83000000 0x6000 0x26000
```

4. A new root file system has now been written to the device and the board can be rebooted.

5.3 From within Linux

It is also possible to do updates to the system from within Linux.

5.3.1 Kernel image and dtb files

Kernel images and dtb (device tree) files are available on a partition of eMMC that is normally not mounted. The instructions below show how to update the kernel image and dtb file (or add new dtb files) from a USB memory stick to the eMMC partition.

1. Copy the files to a USB memory stick
2. Boot into Linux and then insert this into the USB Host Connector on the Carrier Board, see Figure 6. You will see output in the console as below.

```
new high-speed USB device number 3 using ci_hdrc
usb-storage 1-1.3:1.0: USB Mass Storage device detected

scsi0 : usb-storage 1-1.3:1.0
scsi 0:0:0:0: Direct-Access     SanDisk  Cruzer           8.02 PQ:
0 ANSI: 0 CCS
```

```
scsi 0:0:0:1: CD-ROM            SanDisk  Cruzer           8.02 PQ:
0 ANSI: 0
sd 0:0:0:0: [sda] Attached SCSI removable disk
sd 0:0:0:0: [sda] 15704063 512-byte logical blocks: (8.04 GB/7.48
GiB)
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
```

3. Mount the USB memory stick and eMMC partition. The USB memory stick has in this example the device name **sda1** as can be seen in the output in step 2 above. The partition on the eMMC is in this example available at **/dev/mmcblk2p1**, but this could be different on different COM boards.

```
# mkdir /mnt/usb
# mount /dev/sda1 /mnt/usb
# mkdir /mnt/mmcboot
# mount /dev/mmcblk2p1 /mnt/mmcboot
```

4. Copy the bin file and / or dtb file from the USB memory stick to the boot partition.

```
# cp /mnt/usb/zImage /mnt/mmcboot
# cp /mnt/usb/imx6sxea-com-kit.dtb /mnt/mmcboot
```

5. Un-mount devices

```
# umount /mnt/usb
# umount /mnt/mmcboot
```

6 Extend Image with Additional Functionality

There are several ways to enable and add more functionality to an image than what's included by the image recipe. The functionality is enabled by modifying the `build_dir/conf/local.conf` file.

6.1 Image Features

Several predefined packages can be enabled by using the `EXTRA_IMAGE_FEATURES` variable in the `local.conf` file. More information about this variable and the features that are available can be found in the Image Features section in the Yocto reference manual.

As an example the OpenSSH SSH server can be installed by adding `ssh-server-openssh` to the variable.

```
EXTRA_IMAGE_FEATURES = "debug-tweaks ssh-server-openssh"
```

6.2 Additional Packages

The Yocto project includes a lot of recipes for different packages and utilities. Some of them are included in the recipe for the image you are building, but more can be installed into the root file system by adding them to the `IMAGE_INSTALL_append` variable in `local.conf`.

Get a list of available packages in your Yocto setup by running bitbake as below.

```
$ bitbake -s
```

In the example below `e2fsprogs` (file system utilities) and `parted` (manipulates partition tables) has been added to the variable. Please **note** that `IMAGE_INSTALL_append` must start with a space character as in the example below.

```
IMAGE_INSTALL_append = " e2fsprogs parted"
```

7 Lubuntu 14.04 Virtual Machine Setup

This chapter describes the steps needed to create a virtual machine and install Lubuntu 14.04 on it.

7.1 VMware Workstation Player

The virtual machine is run within the VMware Workstation Player. Download and install VMware Player.

Go to www.vmware.com and select Downloads → Free Product Downloads → VMware Workstation Player. Please note that the download path may change. The path described above was valid when writing this document. If you cannot find the player please search on www.vmware.com.

7.2 Download Installation Media

This guide is for Lubuntu 14.04 LTS (Long Time Support) 64 bit. The reason for using a 64-bit version is that it is a requirement when compiling Android. If you don't plan to build Android images then the 32-bit version will work just as well.

> **NOTE**. You can use a newer version of Lubuntu also. These instructions are for 14.04, but we have tested to use Yocto on Lubuntu/Ubuntu 16.04 also.

The 64-bit ISO image can be downloaded from GetLubuntu/LTS

7.3 Creating the VMware Virtual Machine

Start the VMware Player application and select to create a new virtual machine.

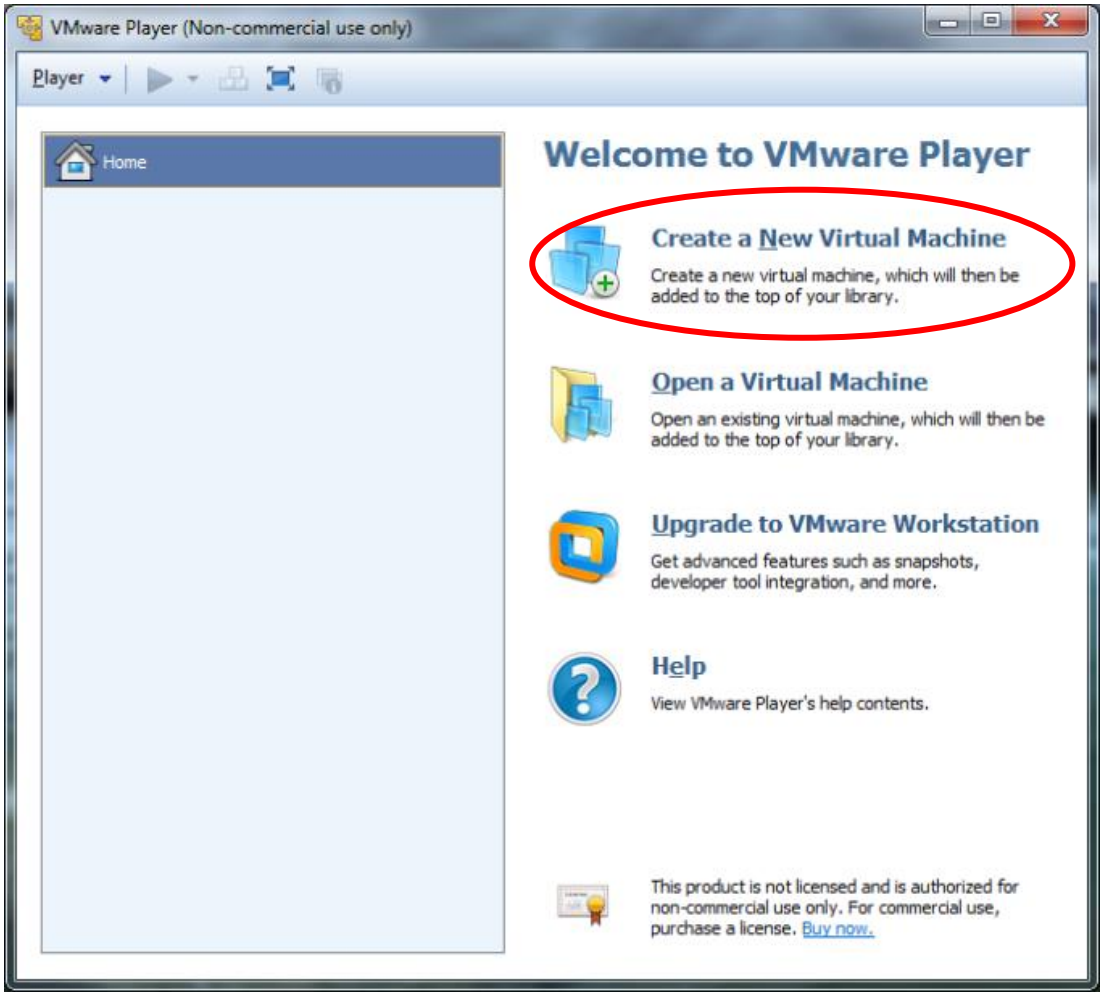

Figure 7 – VMware Player

Follow the instructions in the dialog that appears:

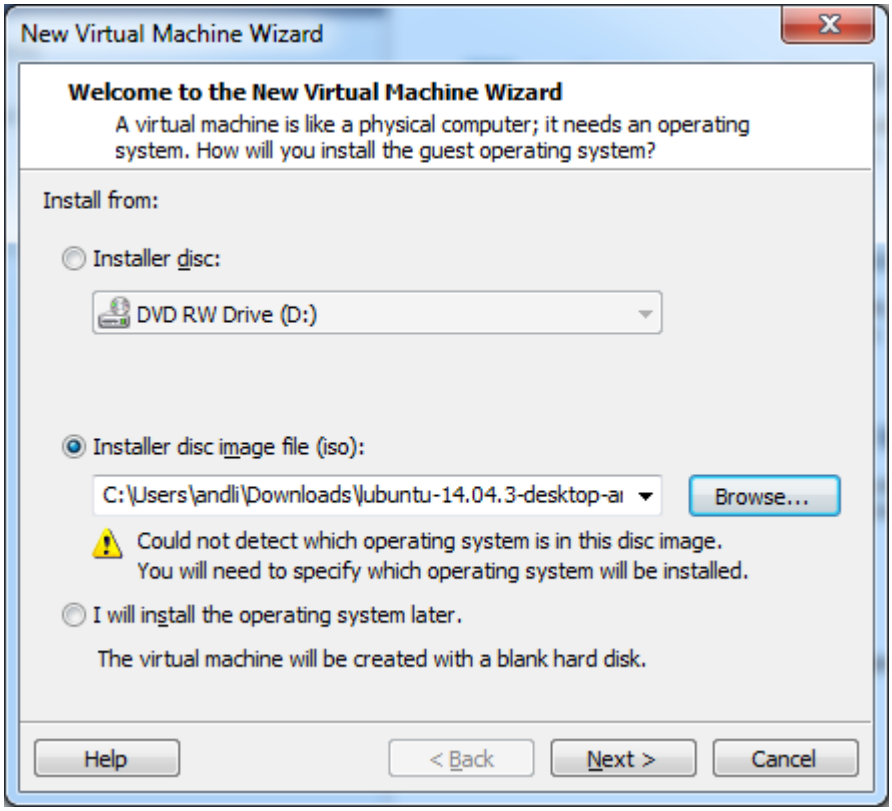

Figure 8 – New Virtual Machine – Select Installation Media

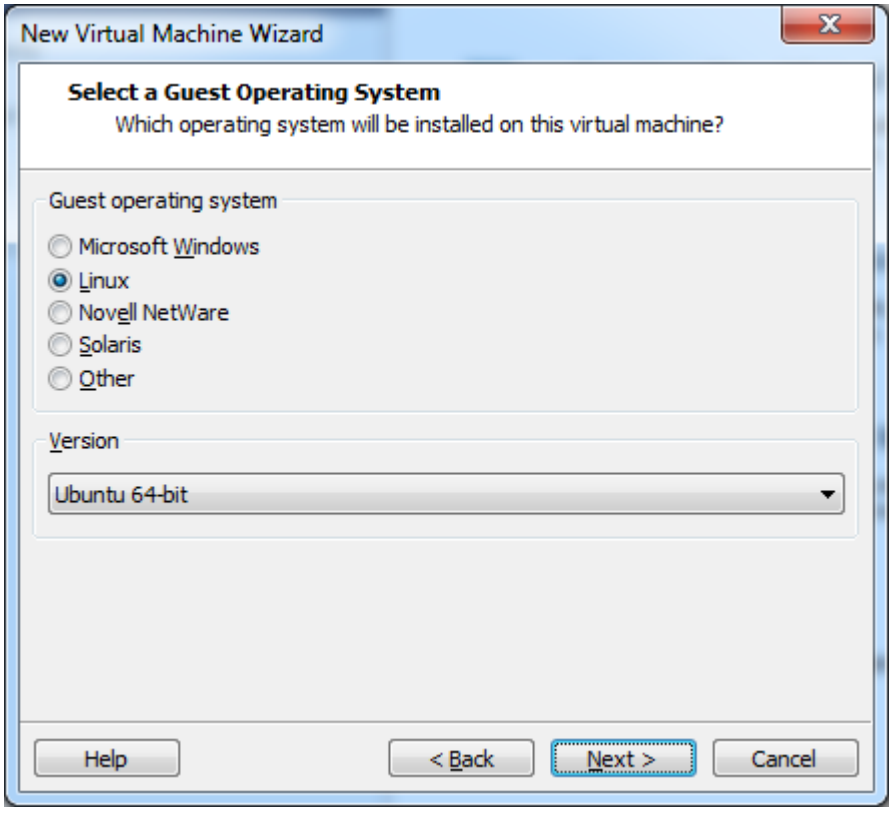

Figure 9 – New Virtual Machine – Select OS Type

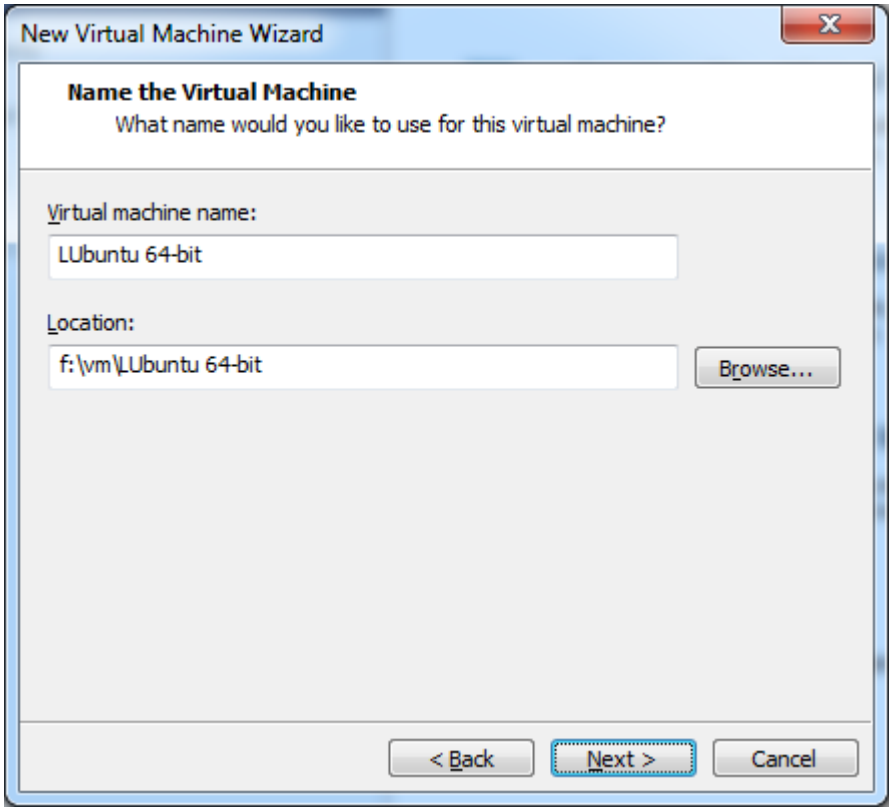

Figure 10 – New Virtual Machine – Select Name and Location

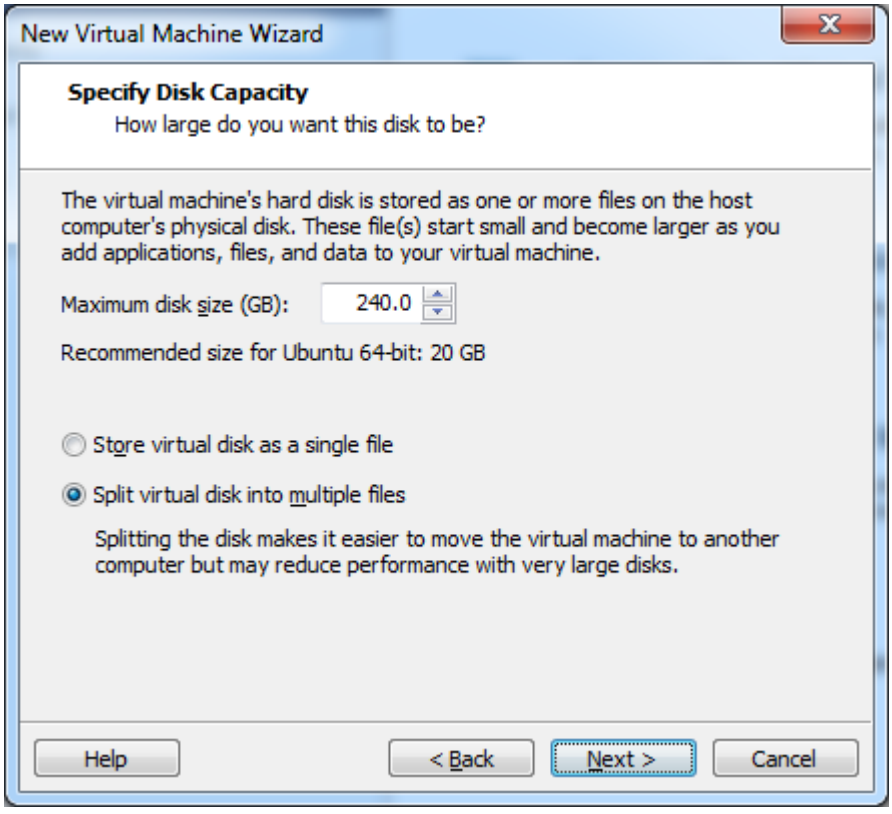

Figure 11 – New Virtual Machine – Select Disc Capcity

The default disc capacity is 20GB which is much too low. It is recommended to have at least 120GB to be able to build. If you plan to build Android then at least 200GB is needed.

In the summary dialog, select to customize the hardware.

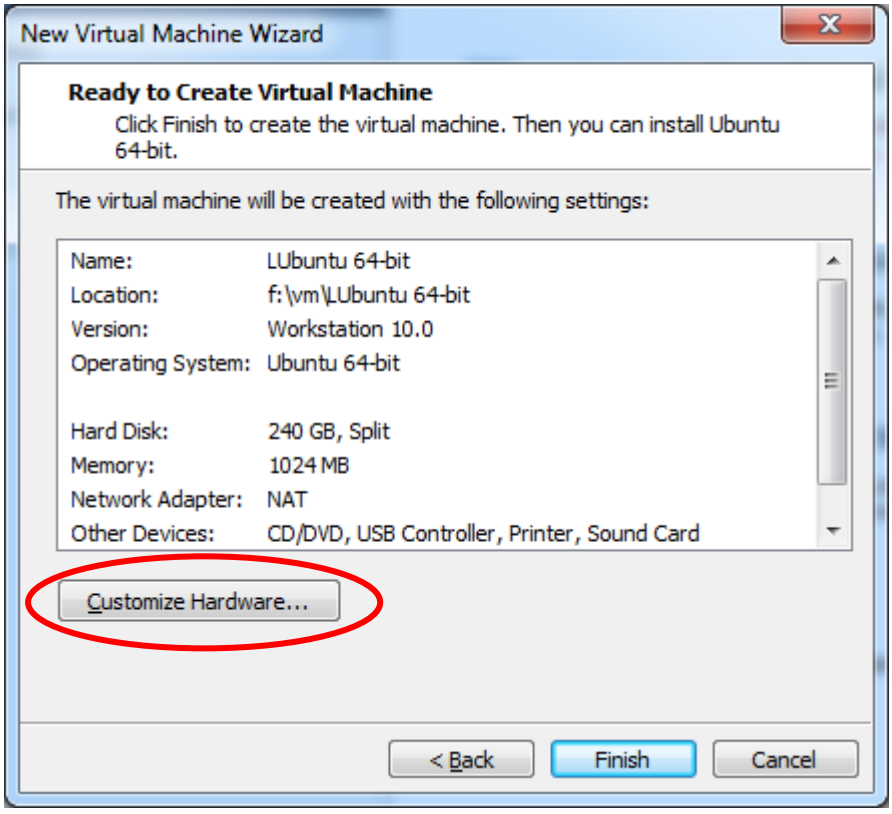

Figure 12 – New Virtual Machine – Summary

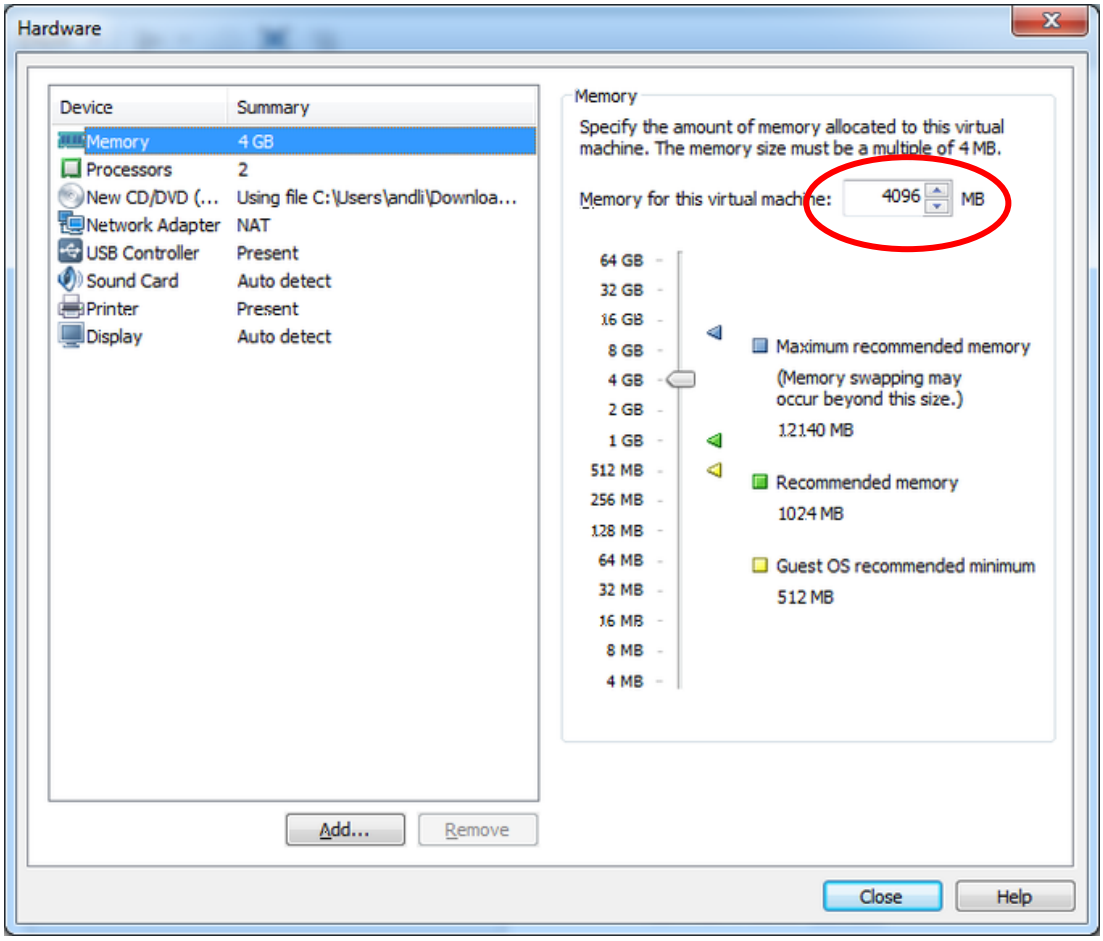

Figure 13 – New Virtual Machine – Memory Configuration

The virtual machine requires at least 3GB of memory but more is better if the host machine (the machine running the VMware Player application) can spare it.

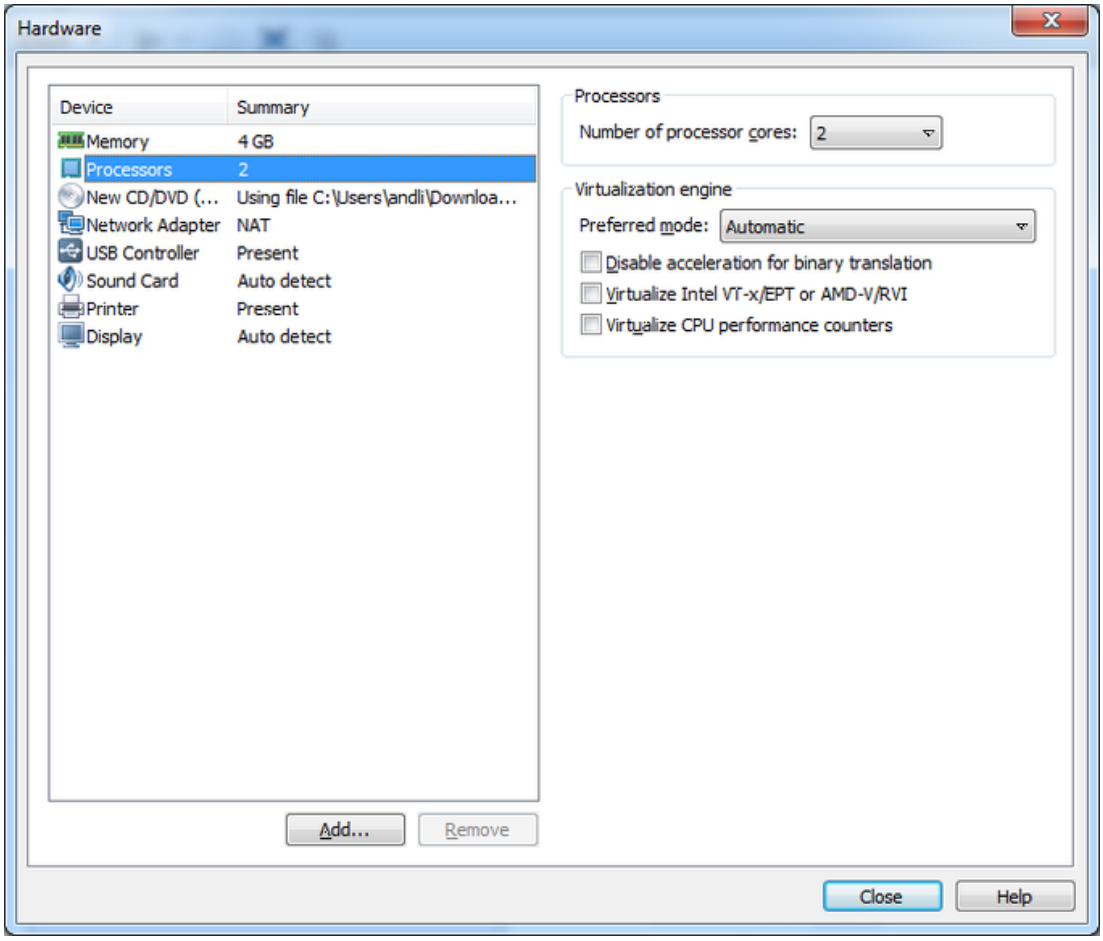

Figure 14 – New Virtual Machine – Processors

If the host machine has a processor with multiple cores then it is possible to assign more than one of those to the virtual machine which will decrease build times.

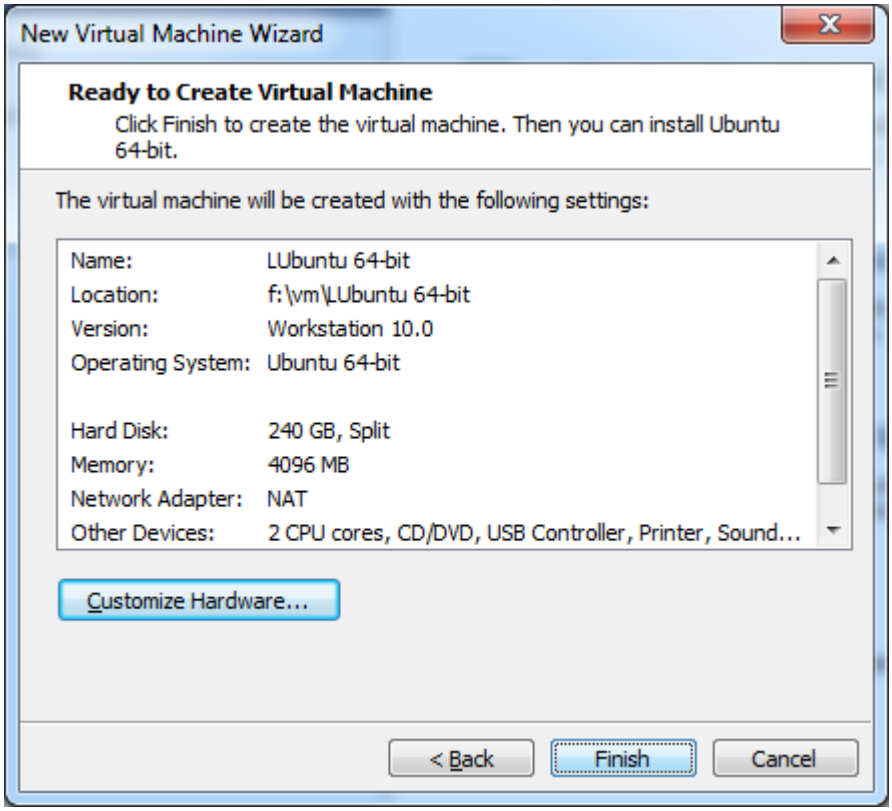

Figure 15 – New Virtual Machine – Finished Configuration

The new virtual machine is now ready to be started. Press the Play button to boot the machine for the first time.

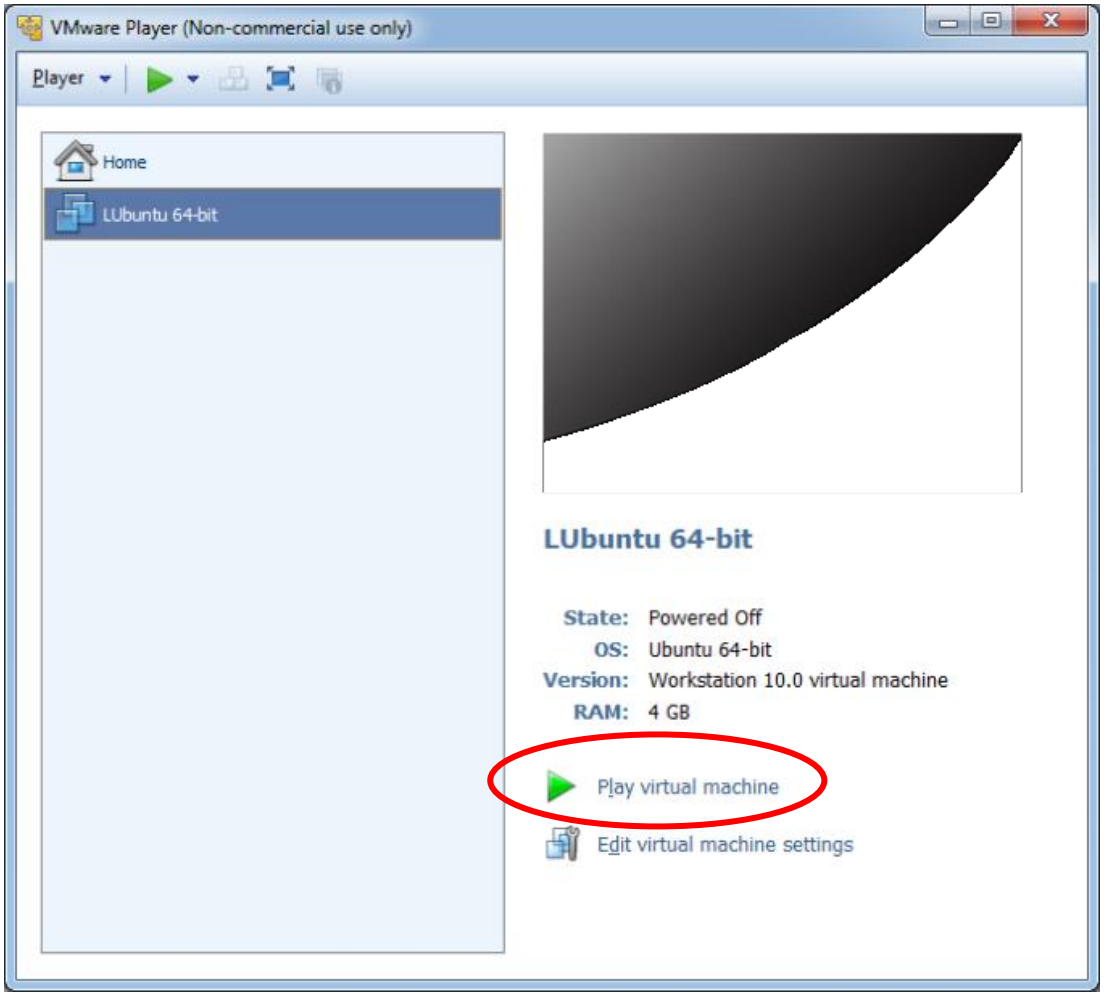

Figure 16 – New Virtual Machine – Start

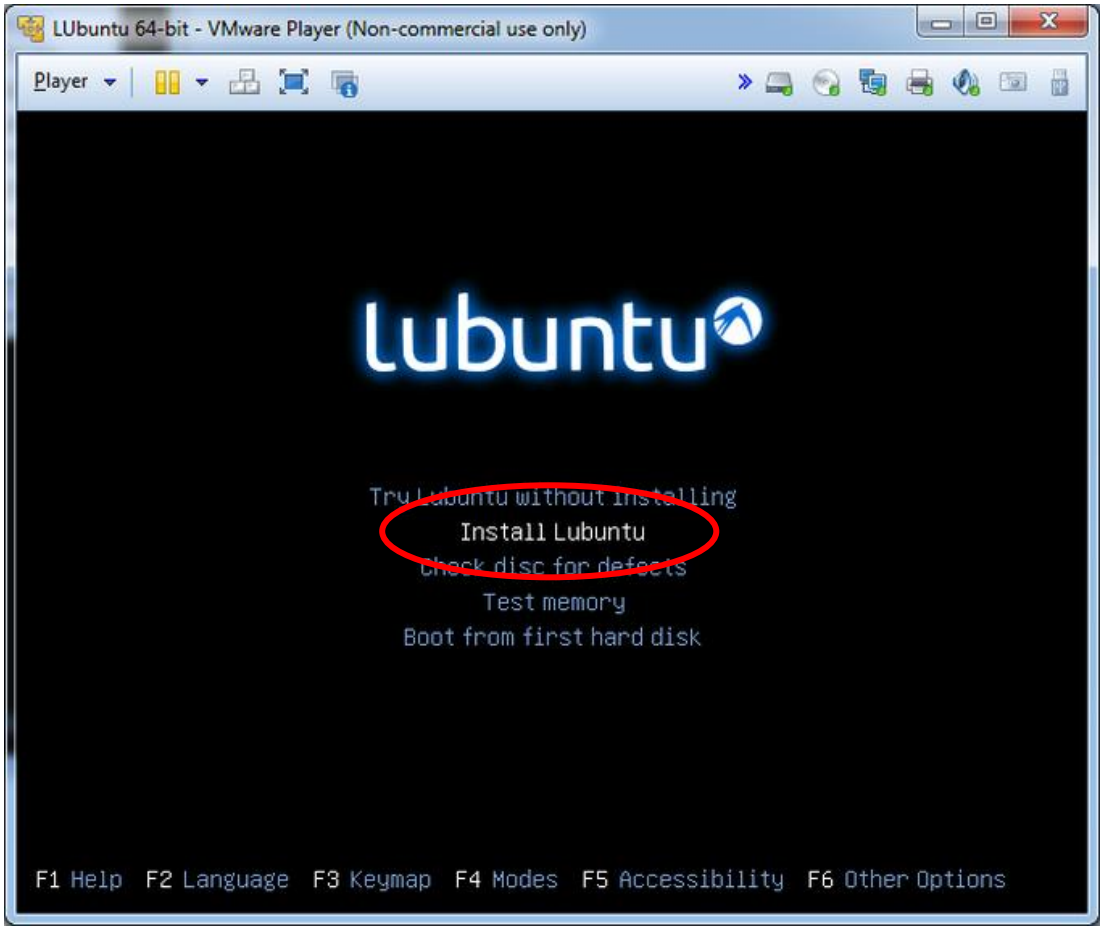

Figure 17 – Installing - Boot Option

Select to install Lubuntu and then follow the on-screen configuration guide.

The first important page is the installation type page where it is a good idea to click the “Use LVM” option as it will make it easier to expand the disc space in the future. It is optional.

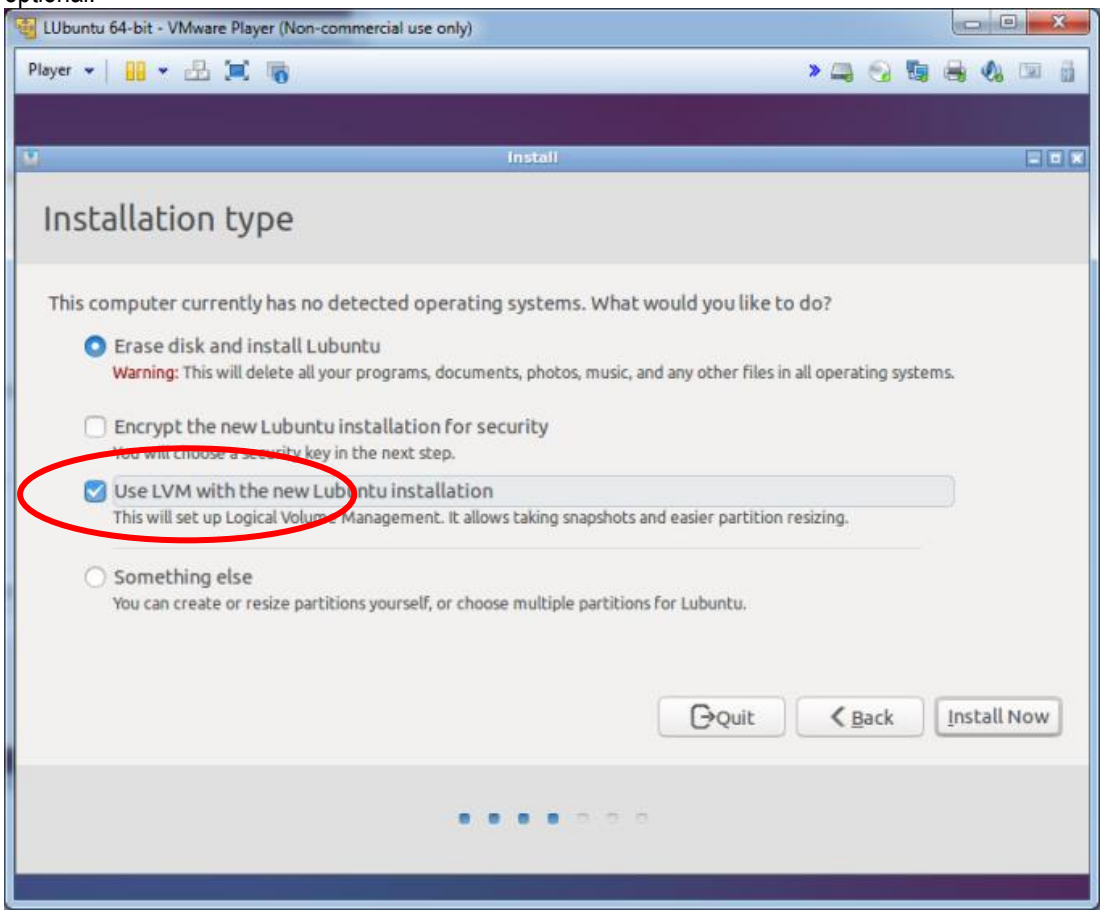

Figure 18 – Installing - Logical Volume Management

The installation will complete and at the end a restart is required. When rebooting there will be a message stating to remove the installation media:

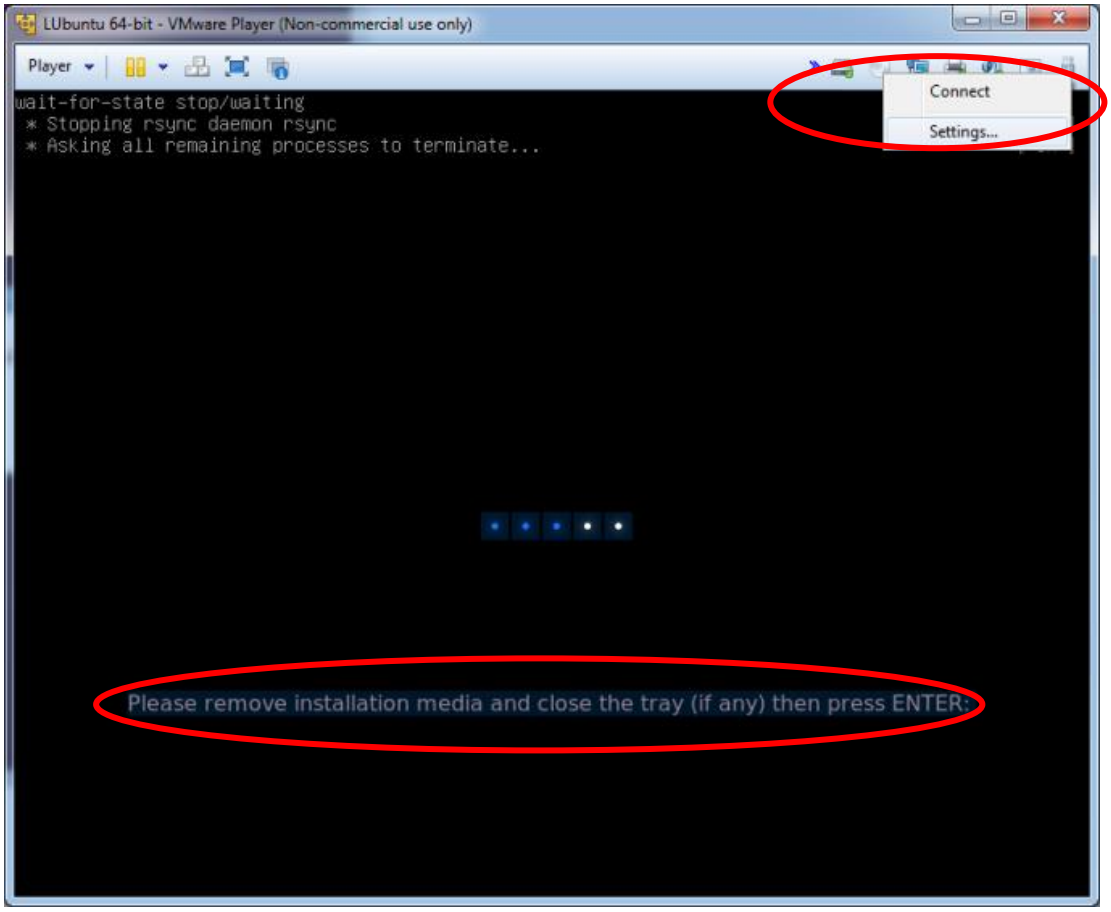

Figure 19 – Installing - Remove Media

To remove the media, right click on the CD icon at the top right of the window and select Settings. Make sure that the "Connect at power on" option is not selected. This is the same as ejecting a CD.

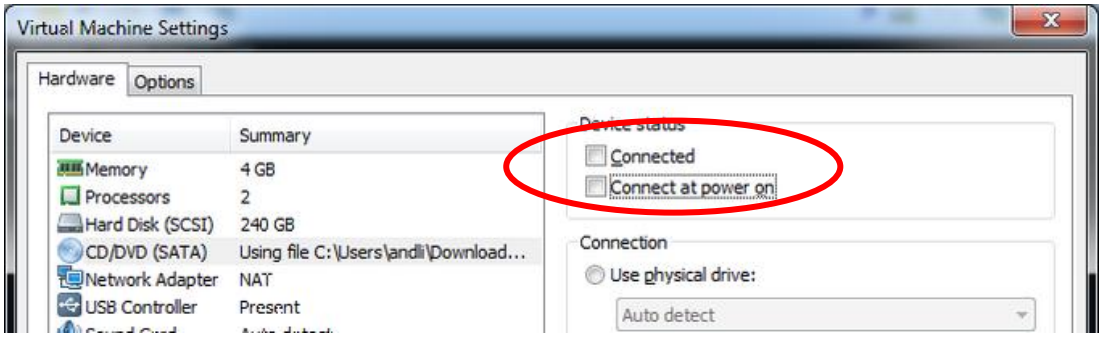

Figure 20 – Installing - Disconnect CD

After removing the CD and the booting has completed the virtual machine should boot into the desktop (possibly asking for login information depending on the options selected during installation).

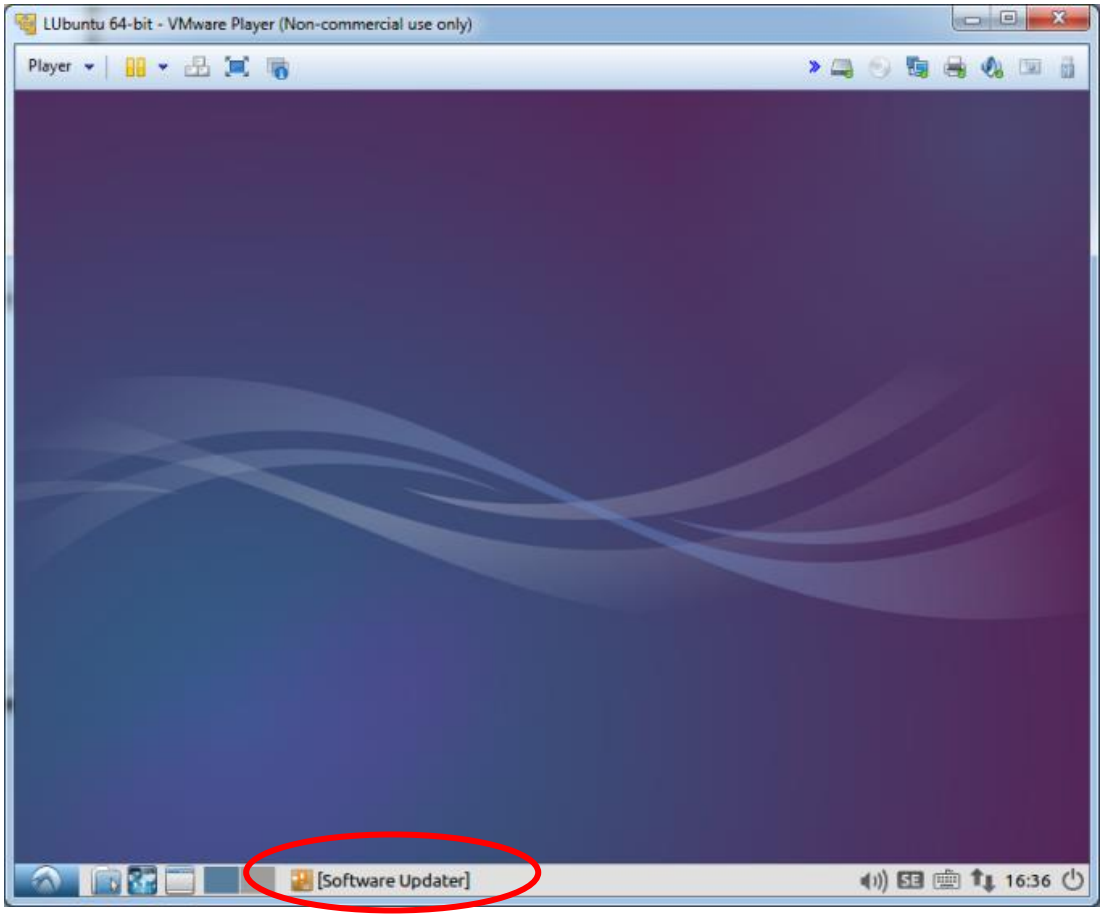

Figure 21 – Desktop

It is recommended to run the Software Updater to get all the fixes that have been made. Click the icon on the bottom of the screen an follow the instructions.

The last thing to do is to install VMware Tools. This is optional but it adds a couple of useful features including

- Move the mouse pointer out of the VMware Player window without having to press Ctrl+Alt
- Copy text between the virtual machine and the host machine

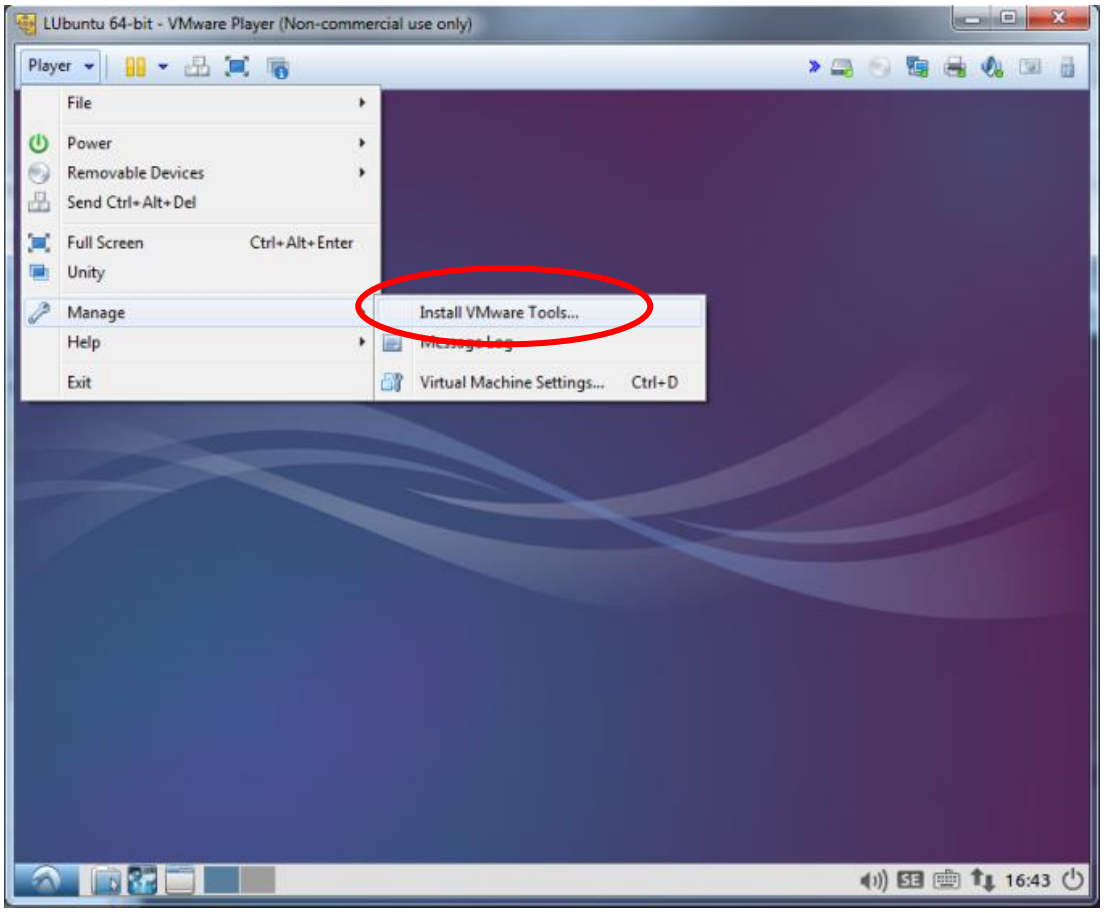

Figure 22 – VMware Tools – Start Installation

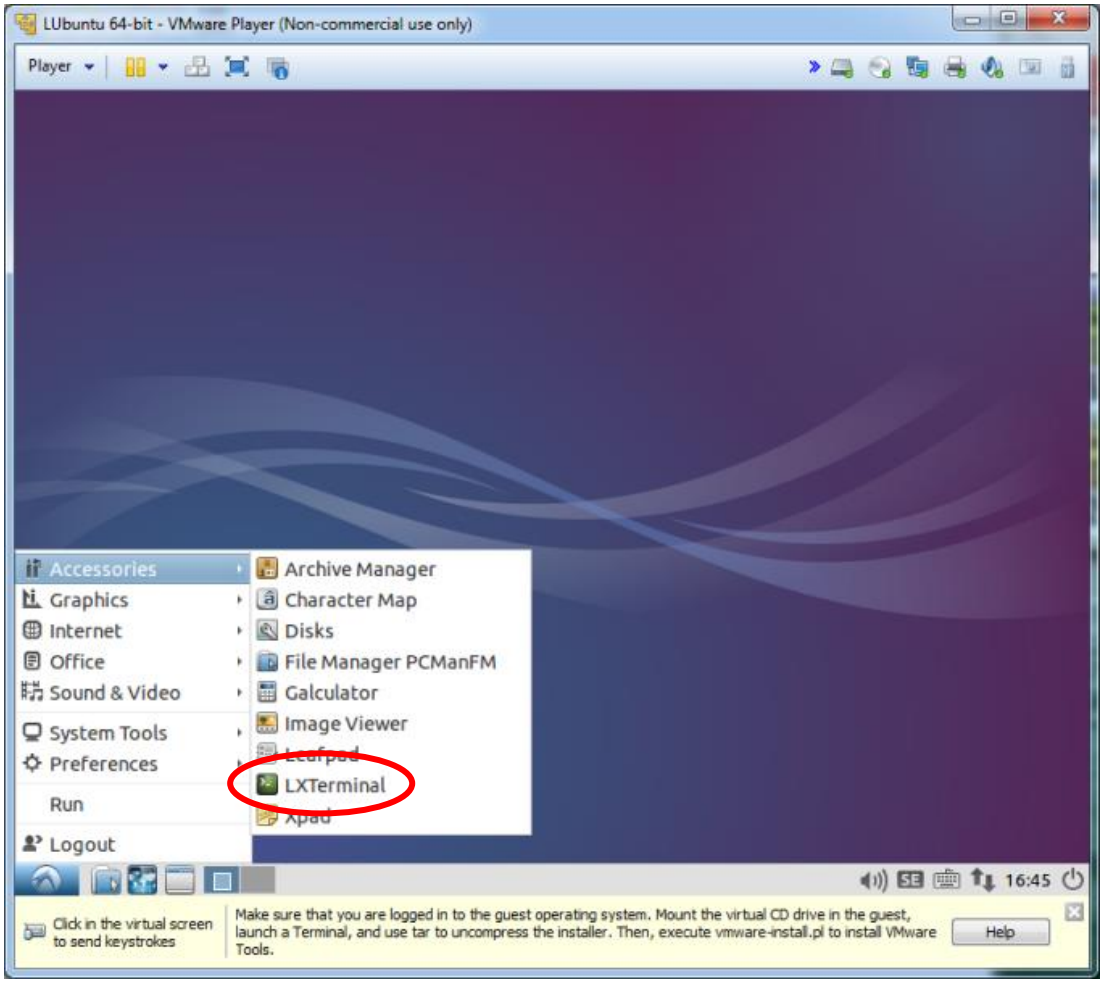

Figure 23 – VMware Tools – Start LXTerminal

Start the LXTerminal and run the following commands in the terminal window that appear:

```
$ cd /tmp/
$ tar -xf /media/user/VMware\ Tools/VMwareTools*.tar.gz
$ cd vmware-tools-distrib
$ sudo ./wmware-install.pl
```

Note that the path above assumes that the login name that you have is “user”. Replace that part with the name used to login to the virtual machine.

The wmware-intall.pl script will ask a lot of questions but always select the default answer (shown as e.g. [yes]) unless you are sure you know what you are doing.

After the last question is answered the VMware tools will be installed. Reboot when the installation program completes.

You now have your own virtual machine!

8 Yocto Images

8.1 meta-toolchain

To be able to build your own application or, for example, u-boot and the Linux kernel outside of Yocto you need a toolchain. The toolchain consists of cross-compiler, linker, and necessary libraries. As mentioned in section 4.1 there is an image named **meta-toolchain** that will create the necessary toolchain.

1. Build the image

```
$ bitbake meta-toolchain
```

2. The build will result in a file located at `<build dir>/tmp/deploy/sdk`. The exact name of the file depends on several, but in our example it is called:

   ```
   fsl-imx-fb-glibc-x86_64-meta-toolchain-cortexa9hf-vfp-neon-
   toolchain-4.1.15-1.2.0.sh
   ```

3. Install the toolchain

```
$ cd <build dir>/tmp/deploy/sdk
$ sudo ./fsl-imx-fb-glibc-x86_64-meta-toolchain-cortexa9hf-vfp-
neon-toolchain-4.1.15-1.2.0.sh
```

4. If you select the default settings the toolchain will in this example be installed in `/opt/fsl-imx-fb/4.1.15-1.2.0`
5. Before building an application run the command below to setup environment variables

```
$ source /opt/fsl-imx-fb/4.1.15-1.2.0/environment-setup-
cortexa9hf-vfp-neon-poky-linux-gnueabi
```

6. You can verify that the environment variables has been correctly setup by running the command below that will show the version of the GCC compiler used.

```
$ $CC –version
arm-poky-linux-gnueabi-gcc (GCC) 5.2.0
...
```

NOTE 1: Setting up environment variables in step 5 may overwrite other variables you already have in your environment. It is, for example, not recommended to do this in the same terminal where you run bitbake to build Yocto images.

It is recommended to build this toolchain on your host computer where you will do the development, but if you have a 64-bit Intel x86 based host computer you can download a pre-built version from imx.embeddedartsits.com

```
$ wget imx.embeddedartists.com/common/fsl-imx-fb-glibc-x86_64-
meta-toolchain-cortexa9hf-vfp-neon-toolchain-4.1.15-1.2.0.sh
```

9 Miscellaneous

9.1 Root file system on SD card

By default u-boot, kernel, and root file system are stored on the onboard eMMC flash. The instructions in this section show how to put the root file system on an external SD card. The u-boot, kernel and dtb files are still stored on the eMMC.

Detect which device file the SD card is available on

1. Boot into Linux without having an SD card inserted in the SD card slot
2. Insert the SD card and you will see output similar to below in the terminal. In this example the device file to use will be `/dev/mmcblk1p1`.

```
mmc1: new SD card at address 9047
mmcblk1: mmc1:9047 SU01G 968 MiB
 mmcblk1: p1
```

Put the root file system on the SD card

1. Make sure you have downloaded and unpacked the manufacturing tool for the board you are using.
2. Open the file `<mfgtool dir>/Profiles/Linux/OS Firmware/ucl2.xml`
3. Look for the section that begins with <LIST name="AllToEmmcRootfsTar.
4. Within this section find the comment "burn rootfs". Here you will find a list of commands that all use the variable %mmc% to indicate which mmc device to update. Remove all these commands using a block comment. Then copy and paste the commands and replace all places `mmcblk%mmc%p2` with the device file previously retrieved (`mmcblk1p1` in this example)

```
<!-- burn rootfs -->

<!--

<CMD state="Updater" type="push" body="$ mkfs.ext3 -F -j
/dev/mmcblk%mmc%p2">Formatting rootfs partition</CMD>
<CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk%mmc%p2"/>
<CMD state="Updater" type="push" body="$ mount -t ext3 /dev/mmcblk%mmc%p2
/mnt/mmcblk%mmc%p2"/>
<CMD state="Updater" type="push" body="pipe tar -jx -C /mnt/mmcblk%mmc%p2"
file="files/%rootfs%-%board%.rootfs.tar.bz2">Sending and writing rootfs</CMD>
<CMD state="Updater" type="push" body="frf">Finishing rootfs write</CMD>

<CMD state="Updater" type="push" body="$ umount /mnt/mmcblk%mmc%p2">Unmounting
rootfs partition</CMD>

-->

<CMD state="Updater" type="push" body="$ mkfs.ext3 -F -j
/dev/mmcblk1p1">Formatting rootfs partition</CMD>
<CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk1p1"/>
<CMD state="Updater" type="push" body="$ mount -t ext3 /dev/mmcblk1p1
/mnt/mmcblk1p1"/>
<CMD state="Updater" type="push" body="pipe tar -jx -C /mnt/mmcblk1p1"
file="files/%rootfs%-%board%.rootfs.tar.bz2">Sending and writing rootfs</CMD>
<CMD state="Updater" type="push" body="frf">Finishing rootfs write</CMD>

<CMD state="Updater" type="push" body="$ umount /mnt/mmcblk1p1">Unmounting rootfs
partition</CMD>
```

5. Run `ea-com-emmc_full_tar.vbs` to update the SD card with the root file system. See chapter 5 for more information about deploying images.

NOTE: In this section we are modifying an existing configuration (AllToEmmcRootfsTar). An alternative is to copy the configuration and renaming it instead of modifying it.

Update u-boot variables:

1. Boot into u-boot
2. Set `mmcautodetect` to `no`. If you don't do this u-boot will detect and set the value of `mmcroot` by itself, that is, overwriting any value you set.

```
# setenv mmcautodetect no
```

3. Set mmcroot to the device file for the SD card

```
# setenv mmcroot '/dev/mmcblk1p1 rootwait rw'
```

4. Save the changes

```
# saveenv
```

5. When you reset/boot the board it will use the root file system stored on the SD card.

9.2 Build Linux kernel from source code

You can build the Linux kernel outside of Yocto by following the instructions in this section. Please note that it is recommended that the kernel is built by Yocto when you are generating your final distribution images since there can be dependencies between the root file system and the kernel.

The instructions in this section assume that you have built and installed the toolchain as described in section 8.1 above.

Setup the **environment variables** for the toolchain. We are using the same installation path as described in section 8.1 above. If you have installed the toolchain in a different path use that path in the instructions below.

```
$ source /opt/fsl-imx-fb/4.1.15-1.2.0/environment-setup-
cortexa9hf-vfp-neon-poky-linux-gnueabi
```

Get the **source code** from the Embedded Artists GitHub repository. In this example we are checking out branch **ea_4.1.15_2.0.0**.

```
$ git clone https://github.com/embeddedartists/linux-imx.git
$ cd linux-imx
$ git checkout ea_4.1.15_2.0.0
```

Use Embedded Artists **kernel configurations**.

```
$ make ea_imx_defconfig
```

(Optional) If you want to change kernel configurations you can at this point run the **menuconfig** tool.

```
$ make menuconfig
```

Build the kernel.

```
$ make
```

When the build process has finished the kernel will be available here: **`arch/arm/boot/zImage`**. Device tree files are available in the following directory: `arch/arm/boot/dts`. A compiled device tree file has the file extension `dtb`.

Updating the system

You have several options to update the system with the new kernel. The first option is to use the manufacturing tool as described in section 5.1 . The other option is to update the boot partition from within Linux as described in section 5.3.1

9.3 Build u-boot from source code

You can build u-boot outside of Yocto by following the instructions in this section.

The instructions in this section assume that you have built and installed the toolchain as described in section 8.1 above.

Setup the **environment variables** for the toolchain. We are using the same installation path as described in section 8.1 above. If you have installed the toolchain in a different path use that path in the instructions below.

```
$ source /opt/fsl-imx-fb/4.1.15-1.2.0/environment-setup-
cortexa9hf-vfp-neon-poky-linux-gnueabi
```

Get the **source code** from the Embedded Artists GitHub repository. In this example we are checking out branch **ea_v2016.03_4.1.15_2.0.0**.

```
$ git clone https://github.com/embeddedartists/uboot-imx.git
$ cd uboot-imx
$ git checkout ea_v2016.03_4.1.15_2.0.0
```

Use the Embedded Artists **configuration** for the COM board you are using. In the example below the configuration for the iMX6 SoloX COM board is used.

```
$ make mx6sxea-com_config
```

Build the bootloader.

```
$ make
```

When the build process has finished the u-boot image will be available directly in the `uboot-imx` directory.

Updating the system

Use the manufacturing tool as described in section 5.1 to update the system with the new u-boot image.