UEFI & EDK II Training

PLATFORM BUILD LAB

tianocore.org

tianocore

PLATFORM BUILD LABS

Lab Setup and Build for OVMF or Minnowboard Max/Turbot

- Lab 1: Build a EDK II Platform using OVMF package
- Lab 2: Hardware Setup for Minnowboard Max/Turbot
- Lab 3: Build a EDK II Platform using Minnowboard Max/Turbot

tianocore

LAB 1: BUILD OVMFPKG

Setup OvmfPkg to build and run w/ QEMU and Ubuntu

PRE-REQUISITES UBUNTU 16.04

Instructions from: tianocore wiki Ubuntu_1610
Example Ubuntu 16.04
The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo apt-get install build-essential uuid-dev iasl git gcc-5 nasm
```

```
build-essential - Informational list of build-essential packages
uuid-dev - Universally Unique ID library (headers and static libraries)
iasl - Intel ASL compiler/decompiler (also provided by acpica-tools)
git - support for git revision control system
gcc-5 - GNU C compiler (v5.4.0 as of Ubuntu 16.04 LTS)
nasm - General-purpose x86 assembler
```

```
bash$ sudo apt-get install qemu
```

Qemu – Emulation with Intel architecture with UEFI Shell

CREATE QEMU RUN SCRIPT

1. Create a run-ovmf directory under the home directory

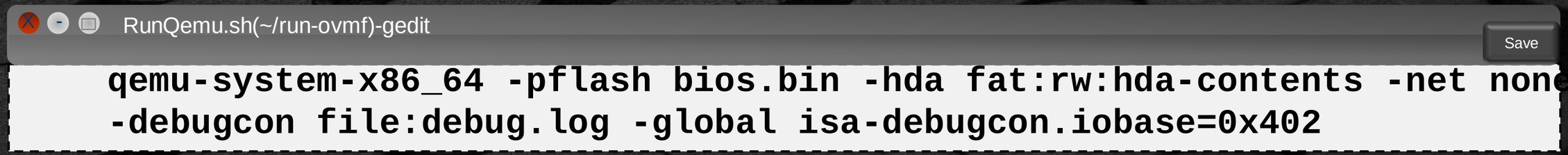
```
bash$ cd ~
bash$ mkdir ~run-ovmf
bash$ cd run-ovmf
```

2. Create a directory to use as a hard disk image

```
bash$ mkdir hda-contents
```

3. Create a Linux shell script to run the QEMU from the run-ovmf directory

```
bash$ gedit RunQemu.sh
```



4. Save and Exit

DOWN LOAD THE EDK II SOURCE - *OPTIONAL*

OPTIONAL **- Open a terminal prompt and create a source working directory**

```
bash$ mkdir ~/src
bash$ cd ~/src
```

OPTIONAL **- Internet Proxies – (company Firewall used for example)**

```
bash$ export http_proxy=http://proxy-us.company.com:911
bash$ export ftp_proxy=$http_proxy
```

OPTIONAL **- Download edk2 source tree using Git**

```
bash$ git clone https://github.com/tianocore/edk2
```

OPTIONAL **- Build the tools**

```
bash$ make -C edk2/BaseTools
```

NOTE: Lab Material will have a different “edk2”

tianocore

SETUP LAB MATERIAL

Lab_Material_FW.zip

DOWN LOAD LAB MATERIAL

Download the Lab_Material_FW.zip from : github.com
Lab_Matrial_FW.zip
OR
Use git clone to download the Lab_Material_FW

```
bash$ cd $HOME
bash$ git clone https://github.com/Laurie0131/Lab_Material_FW.git
```

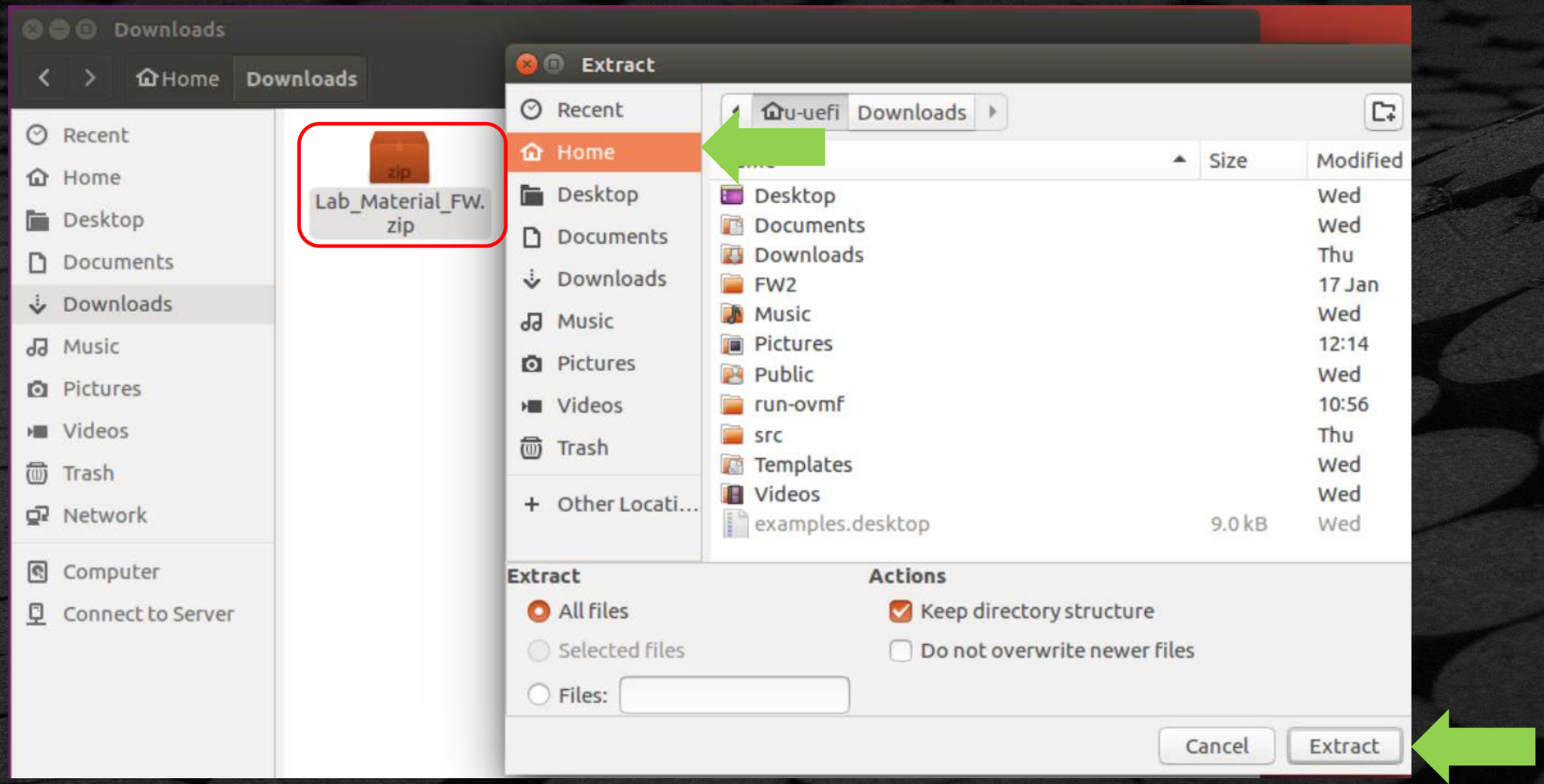
Directory Lab_Material_FW will be created

```
FW
 - Documentation
 - DriverWizard
 - edk2
 - edk2Linux
 - LabSampleCode
```

BUILD EDK II OVMF

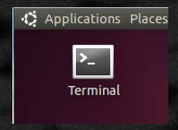
-Extract the Source

1. Extract the Downloaded **`Lab_Material_FW.zip`** to **`Home`** (this will create a directory FW)



BUILD EDK II OVMF

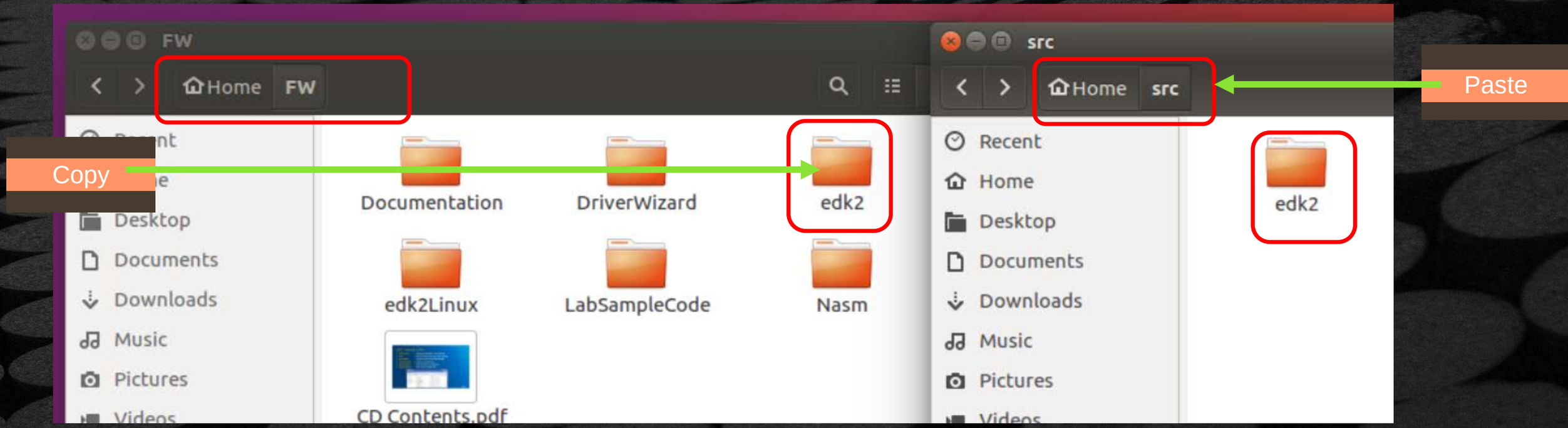
-Getting the Source

2. Open a terminal prompt (Alt-Cnt-T) 

3. Create a working space source directory under the home directory

```
bash$ mkdir ~src
```

4. From the FW folder, copy and paste folder “~FW/edk2” to ~src



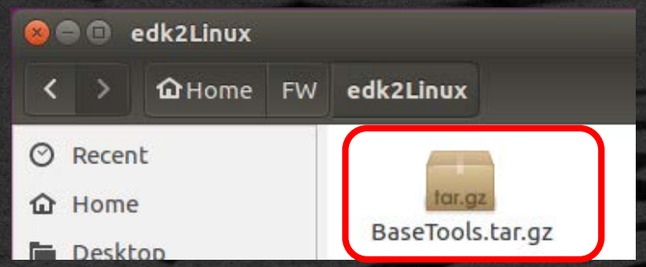
BUILD EDK II OVMF

-Getting the Source

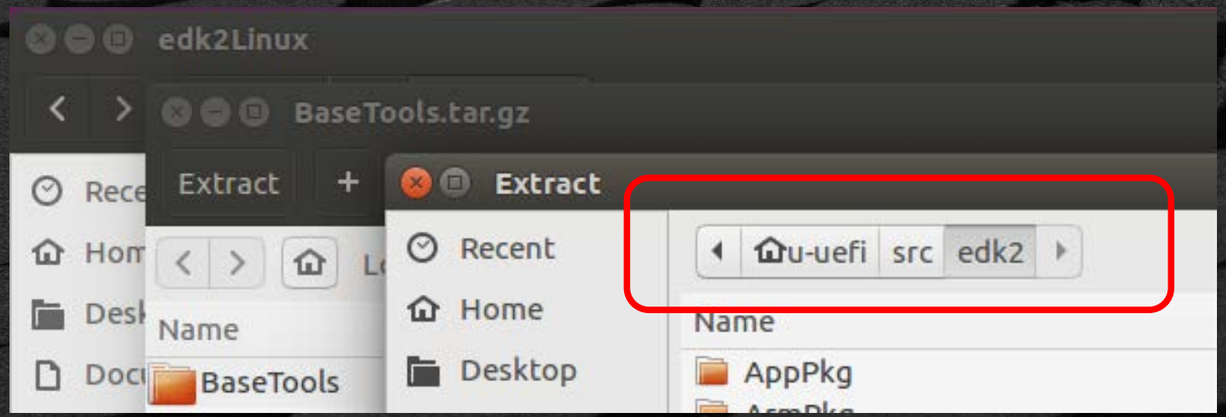
5. Rename or mv the directory "**~src/edk2/BaseTools**"

```
bash$ cd ~src/edk2
bash$ mv BaseTools BaseToolsX
```

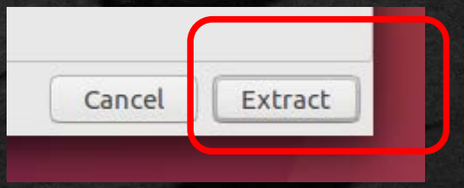
6. Extract the file **~FW/edk2Linux/BaseTools.tar.gz** to **~src/edk2**



Double click



Select Extract directory
$HOME/src/edk2



click

BUILD EDK II OVMF

-Getting the Source

7. Run Make from the Terminal prompt

```
bash$ cd ~src/edk2
bash$ make –C BaseTools
```

```
testSurrogatePairUnicodeCharInUtf16File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ..
. ok
testUtf16InUniFile (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ... ok

----------------------------------------------------------------------
Ran 263 tests in 1.256s

OK
make[1]: Leaving directory '/home/u-uefi/src/edk2/BaseTools/Tests'
make: Leaving directory '/home/u-uefi/src/edk2/BaseTools'
u-uefi@uuefi-TPad:~/src/edk2$
```

About 15 seconds

8. Run edksetup (note This will need to be done for every new Terminal prompt)

```
bash$ . edksetup.sh
```

```
lju@lju-VirtualBox:~/src/edk2$  . edksetup.sh BaseTools
WORKSPACE: /home/lju/src/edk2
EDK_TOOLS_PATH: /home/lju/src/edk2/BaseTools
Copying $EDK_TOOLS_PATH/Conf/build_rule.template
     to $WORKSPACE/Conf/build_rule.txt
Copying $EDK_TOOLS_PATH/Conf/tools_def.template
     to $WORKSPACE/Conf/tools_def.txt
Copying $EDK_TOOLS_PATH/Conf/target.template
     to $WORKSPACE/Conf/target.txt
lju@lju-VirtualBox:~/src/edk2$
```

tianocore

BUILD OVMF PLATFORM

tianocore

BUILD EDK II OVMF

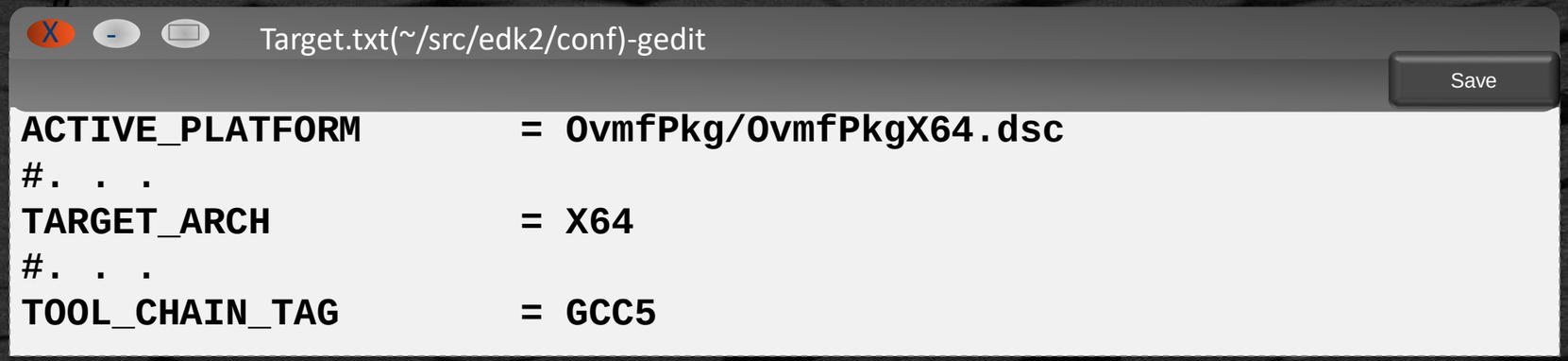
-Update Target.txt

What is OVMF?

Open Virtual Machine Firmware - Build with edk2

Edit the file Conf/target.txt

```
bash$ gedit Conf/target.txt
```



Save and Exit

```
bash$ cd ~src/edk2
bash$ build
```

BUILD EDK II OVMF

-Update Target.txt – Copy and Paste

Edit the file Conf/target.txt – Copy and Paste

```
bash$ gedit Conf/target.txt
```

In the gedit update: #

```
ACTIVE_PLATFORM       = OvmfPkg/OvmfPkgX64.dsc
   #. . .
TARGET_ARCH           = X64
   #. . .
TOOL_CHAIN_TAG        = GCC5
```

Then Build

```
bash$ cd ~src/edk2
bash$ build
```

tianocore

BUILD EDK II OVMF

-Inside Terminal

```
u-uefi@uuefi-TPad: ~/src/FW/edk2
BuildEnv.sh  build_rule.txt  ReadMe.txt  target.txt  tools_def.txt
u-uefi@uuefi-TPad:~/src/FW/edk2$ gedit Conf\target.txt
u-uefi@uuefi-TPad:~/src/FW/edk2$ gedit
u-uefi@uuefi-TPad:~/src/FW/ed
u-uefi@uuefi-TPad:~/src/FW/ed
Build environment: Linux-4.8.
Build start time: 11:51:26, J

WORKSPACE        = /home/u-ue
ECP_SOURCE       = /home/u-ue
EDK_SOURCE       = /home/u-ue
EFI_SOURCE       = /home/u-ue
EDK_TOOLS_PATH   = /home/u-ue
CONF_PATH        = /home/u-ue


Architecture(s)  = X64
Build target     = DEBUG
Toolchain        = GCC5

Active Platform          = /h
Flash Image Definition   = /h

Processing meta-data .....
```

```
ome/lju/src/edk2/MdePkg/Include -I/home/lju/src/edk2/MdePkg/Include/X64 /home/l
u/src/edk2/MdePkg/Library/BasePrintLib/PrintLib.c
"echo" Symbol renaming not needed for /home/lju/src/edk2/Build/OvmfX64/DEBUG_GC
44/X64/MdePkg/Library/BasePrintLib/BasePrintLib/OUTPUT/./PrintLib.obj
Symbol renaming not needed for /home/lju/src/edk2/Build/OvmfX64/DEBUG_GCC44/X64
MdePkg/Library/BasePrintLib/BasePrintLib/OUTPUT/./PrintLib.obj
"/usr/bin/ar" -cr /home/lju/src/edk2/Build/OvmfX64/DEBUG_GCC44/X64/MdePkg/Libra
y/BasePrintLib/BasePrintLib/OUTPUT/BasePrintLib.lib  @/home/lju/src/edk2/Build/
vmfX64/DEBUG_GCC44/X64/MdePkg/Library/BasePrintLib/BasePrintLib/OUTPUT/object_f
les.lst
Building ... /home/lju/src/edk2/PcAtChipsetPkg/Library/SerialIoLib/SerialIoLib.
nf [X64]
"/usr/bin/gcc" -g -fshort-wchar -fno-stack-protector -fno-strict-aliasing -Wall
-Werror -Wno-missing-braces -Wno-array-bounds -ffunction-sections -fdata-sectio
s -c -include AutoGen.h -DSTRING_ARRAY_NAME=PcAtSerialPortLibStrings -m64 "-DEF
API=__attribute__((ms_abi))" -DNO_BUILTIN_VA_FUNCS -mno-red-zone -Wno-address -
cmodel=large -mno-mmx -mno-sse -o /home/lju/src/edk2/Build/OvmfX64/DEBUG_GCC44/
64/PcAtChipsetPkg/Library/SerialIoLib/SerialIoLib/OUTPUT/./SerialPortLib.obj -I
home/lju/src/edk2/PcAtChipsetPkg/Library/SerialIoLib -I/home/lju/src/edk2/Build
Ov
ho
2/
lP
```

```
GUID cross reference file can be found at /home/u-uefi/src/edk2/Build/OvmfX64/DE
BUG_GCC5/FV/Guid.xref

FV Space Information
SECFV [10%Full] 212992 total, 22288 used, 190704 free
FVMAIN_COMPACT [33%Full] 3440640 total, 1157592 used, 2283048 free
DXEFV [38%Full] 10485760 total, 3990040 used, 6495720 free
PEIFV [19%Full] 917504 total, 177768 used, 739736 free

- Done -
Build end time: 09:20:45, Jun.14 2017
Build total time: 00:00:19

u-uefi@uuefi-TPad:~/src/edk2$
```

Finished build

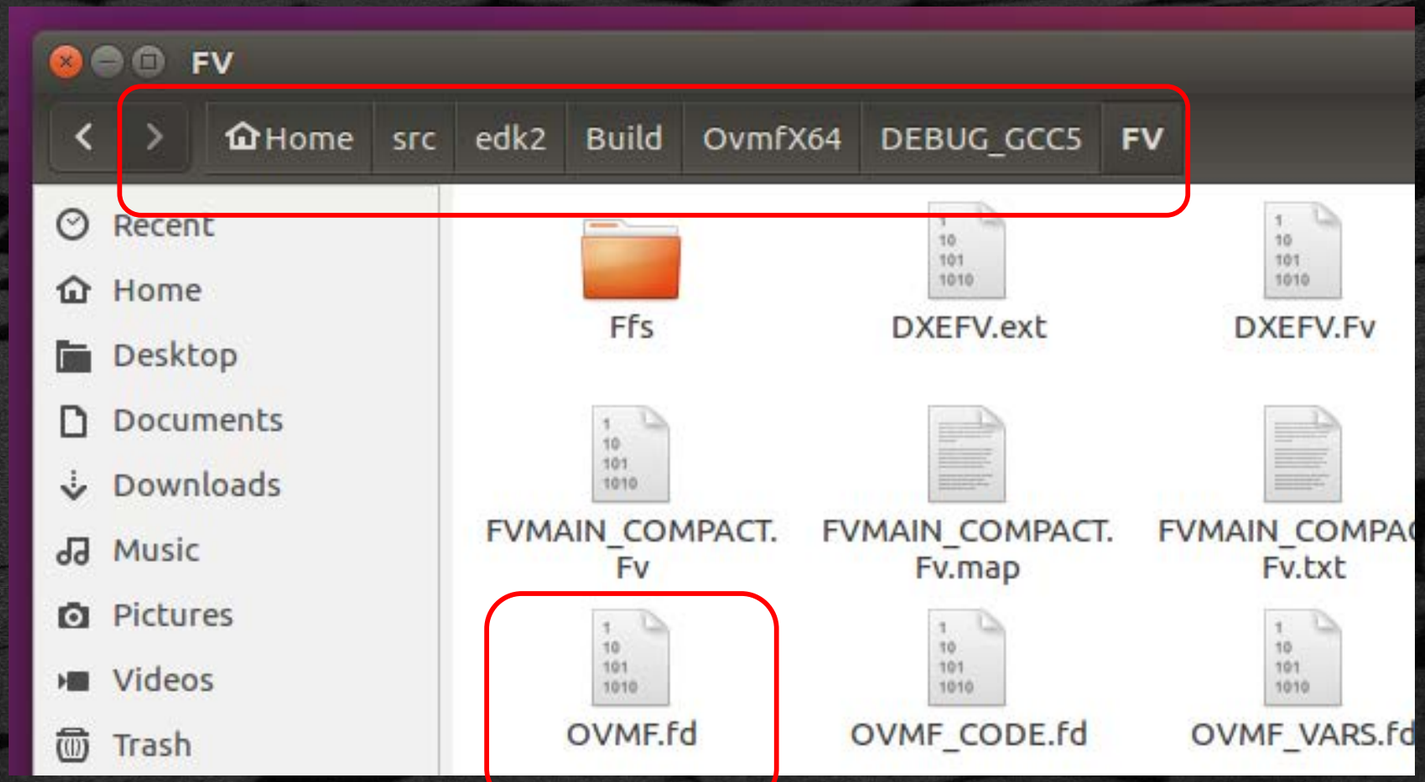
BUILD EDK II OVMF

-Verify Build Succeeded

OVMF.fd should be in the Build directory

- For GCC5 with X64, it should be located at

```
~/src/edk2/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd
```



INVOKE QEMU

Change to run-ovmf directory under the home directory

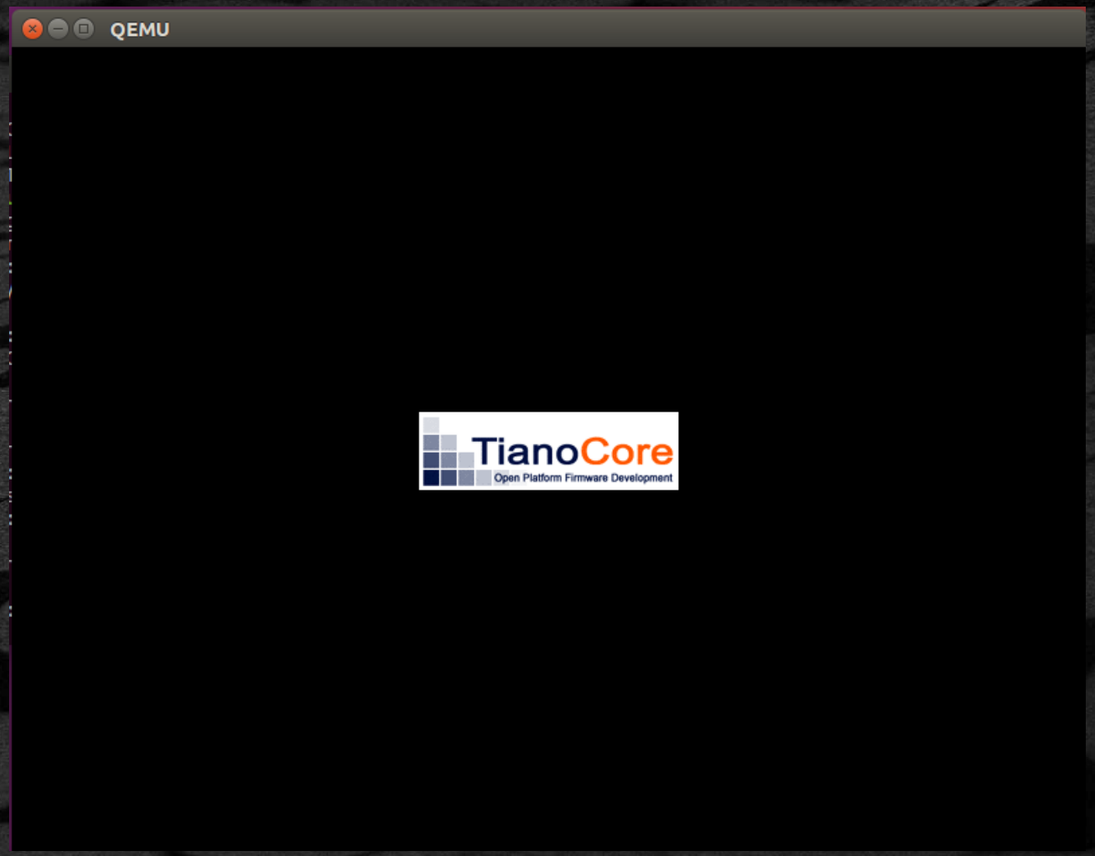
```
bash$ cd $HOME/run-ovmf
```

Copy the OVMF.fd BIOS image created from the build to the run-ovmf directory naming it bios.bin

```
bash$ cp
~/src/edk2/Build/OvmfX64/DEBUG_GCC5/FV/
OVMF.fd bios.bin
```

Run the RunQemu.sh Linux shell script

```
bash$ . RunQemu.sh
```



tianocore

END OF LAB

Return to the Beginning

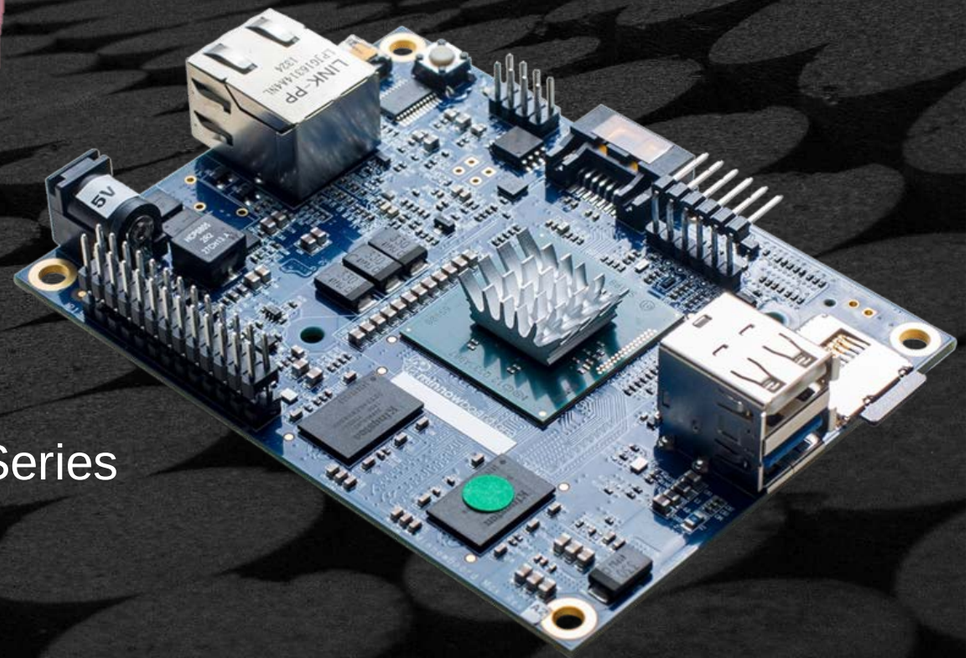
tianocore

LAB 2: PLATFORM HW SETUP

Setup hardware for the MinnowBoard Max/Turbot

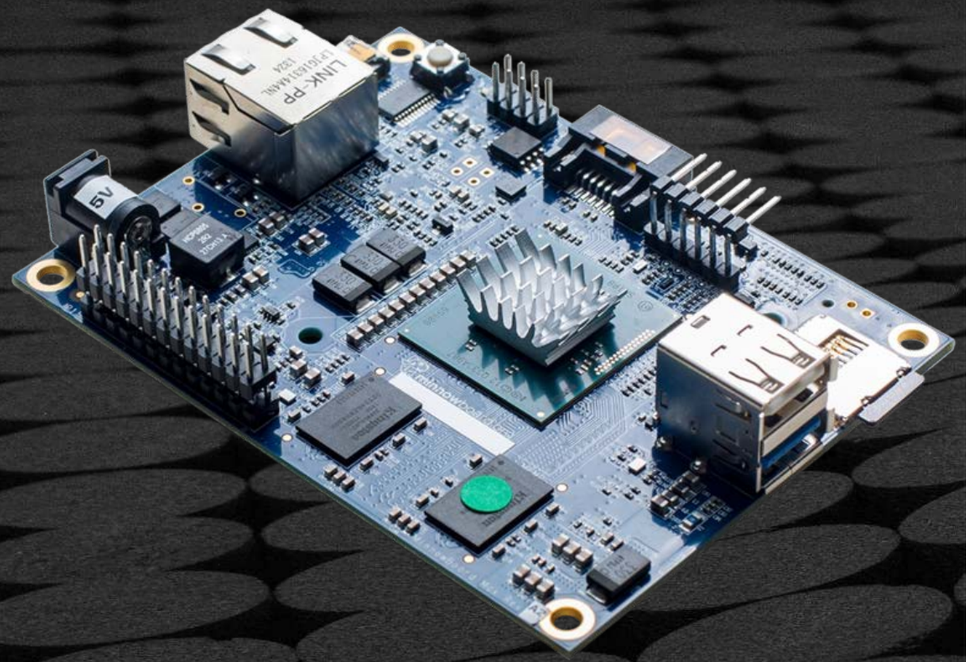
EDK II PLATFORM (MINNOWBOARD MAX/TURBOT)

minnowboard.org



Intel Atom® processor E3800 Series
(Formerly Bay Trail-I)

MINNOWBOARD MAX WORKSHOP LAB HARDWARE



FTDI USB Cable

5V** Power Supply

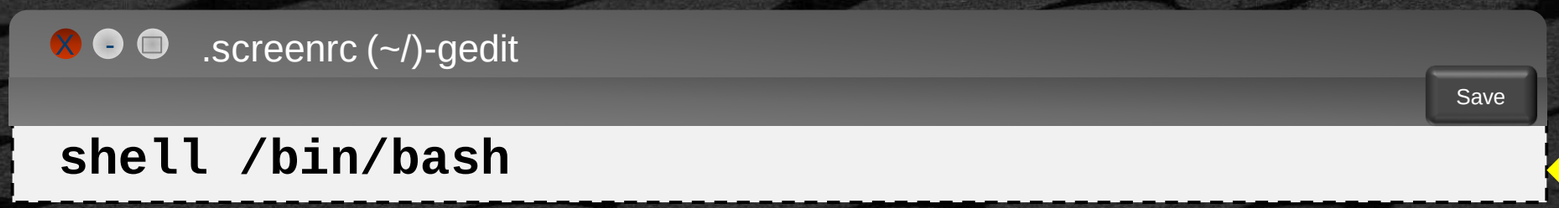
USB thumb drive

**Warning do not use any other power supply than 5V or the board will Fry

INSTALL UBUNTU “SCREEN”

Terminal prompt (Cnt-Alt-T)

```
bash$ sudo apt-get install screen
bash$ cd $Home
bash$ gedit ~.screenrc
```



Click Save

While in screen

Cnt-A then **D** goes back to Terminal

```
bash$ screen –r
```

(returns to screen)

SETUP MINNOWBOARD MAX TEST SYSTEM

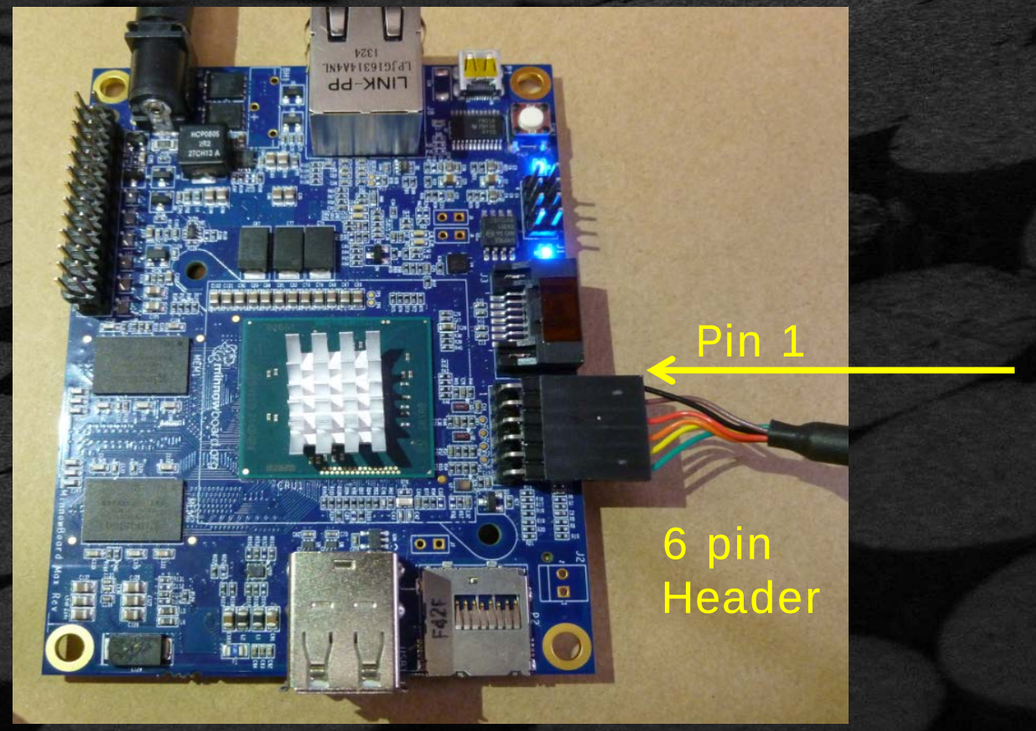
Hardware:

- System Under Test (SUT) – MinnowBoard Max or Turbot
- 5V** power supply
- USB to 3.3V TTL Cable (6 pin to USB Type A)

  Connect the USB w/ 6 pin header to SUT

  - black wire(pin 1) is closest to the SATA connector

  Connect the USB Type A connector to Host



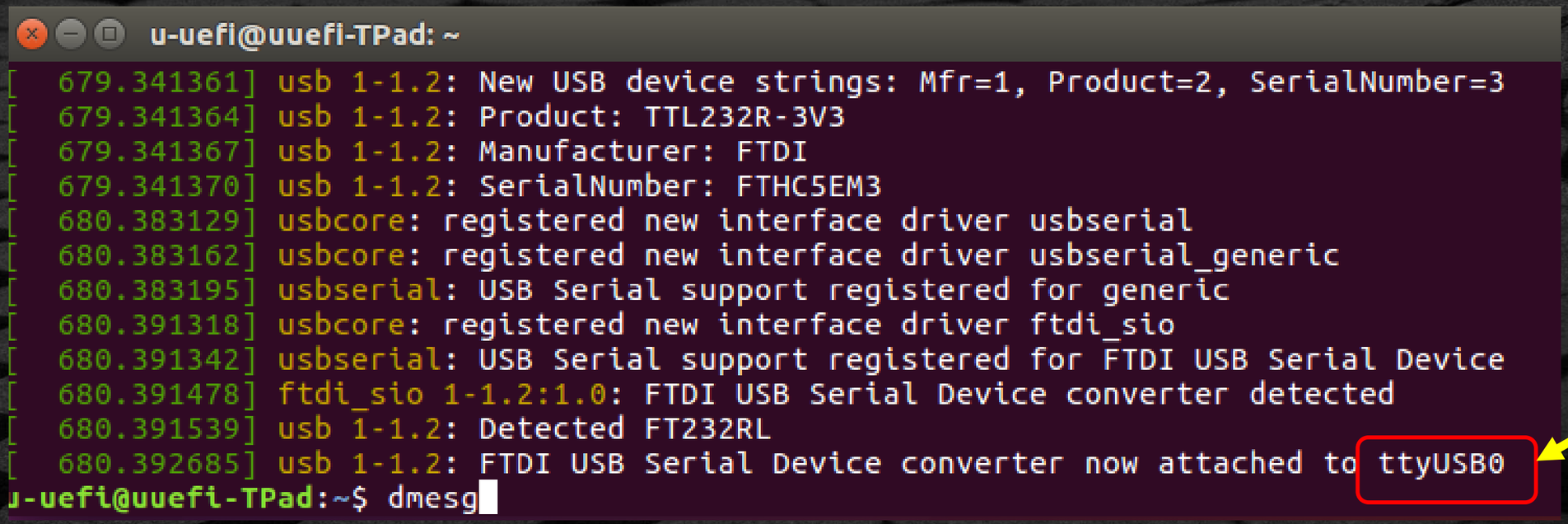
**Warning do not use any other power supply than 5V or the board will Fry

SETUP MINNOWBOARD MAX TEST SYSTEM

Open Terminal Prompt (Cnt-Alt-T)

```
bash$ dmesg
bash$ sudo chmod 666 /dev/ttyUSBn
```

(to check which USB port is assigned)
(where ***n*** is the FTDI number)



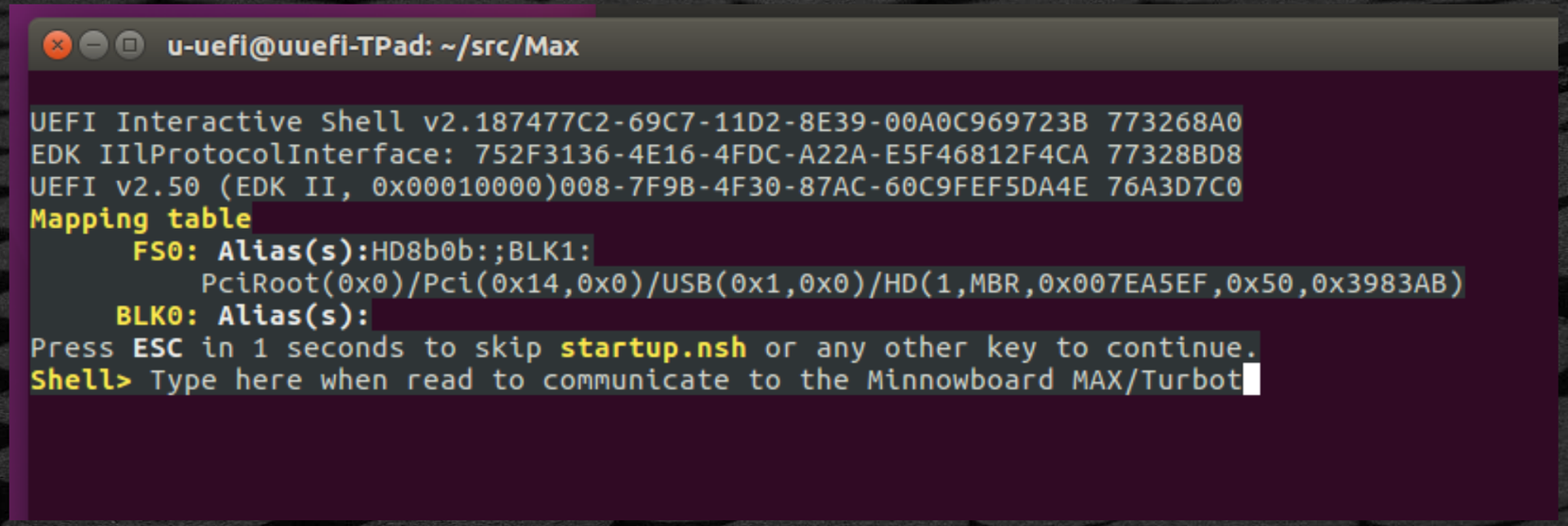
dmesg command
- ttyUSB0

POWER ON MINNOWBOARD MAX

Connect the Power supply cable to the MinnowBoard MAX

```
bash$ screen /dev/ttyUSBn 115200
```

MinnowBoard MAX should boot to the UEFI Shell in the Terminal – Screen .



While in screen

Cnt-A then **D** goes back to terminal

```
bash$ screen –r
(returns to screen)
```

Note: Cnt-H for Backspace

tianocore

END OF LAB

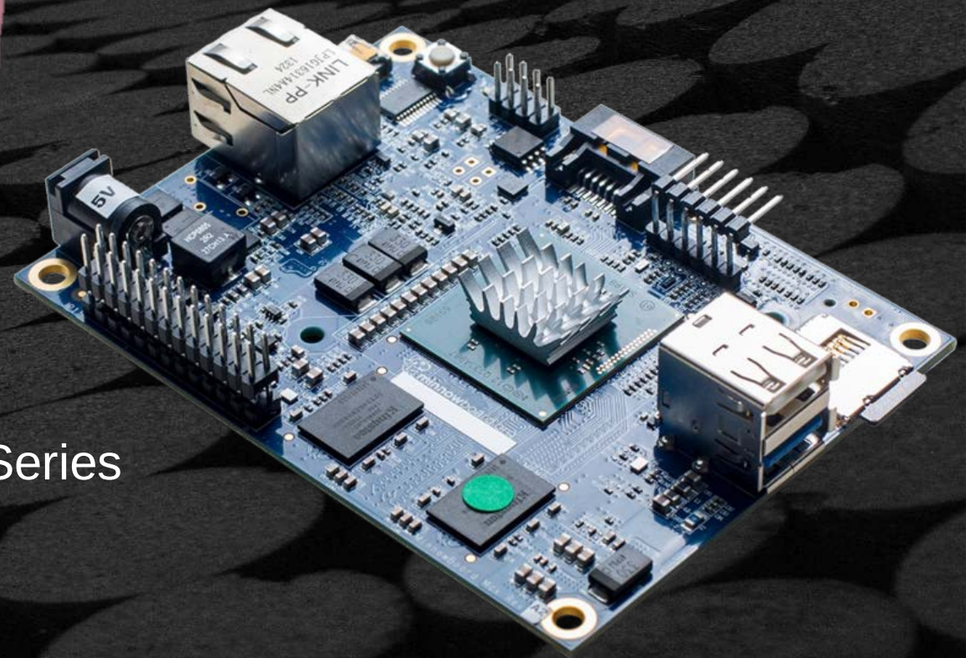
Return to the Beginning or > to continue

tianocore

LAB 3: BUILD MINNOWBOARD TURBOT

EDK II PLATFORM (MINNOWBOARD MAX/TURBOT)

minnowboard.org

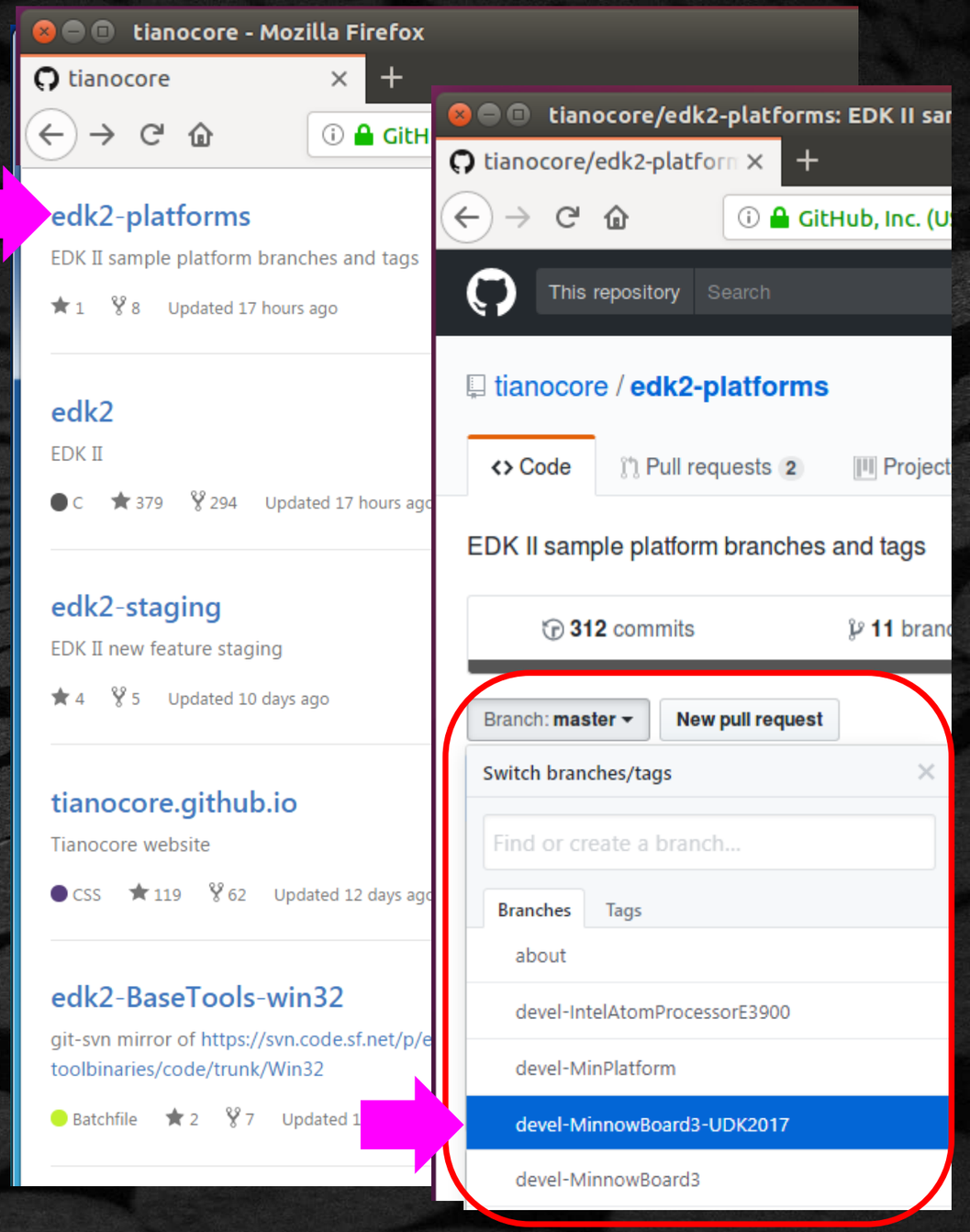


Intel® Atom processor E3800 Series
(Formerly Bay Trail-I)

tianocore

Where to get Open Source MinnowBoard Max

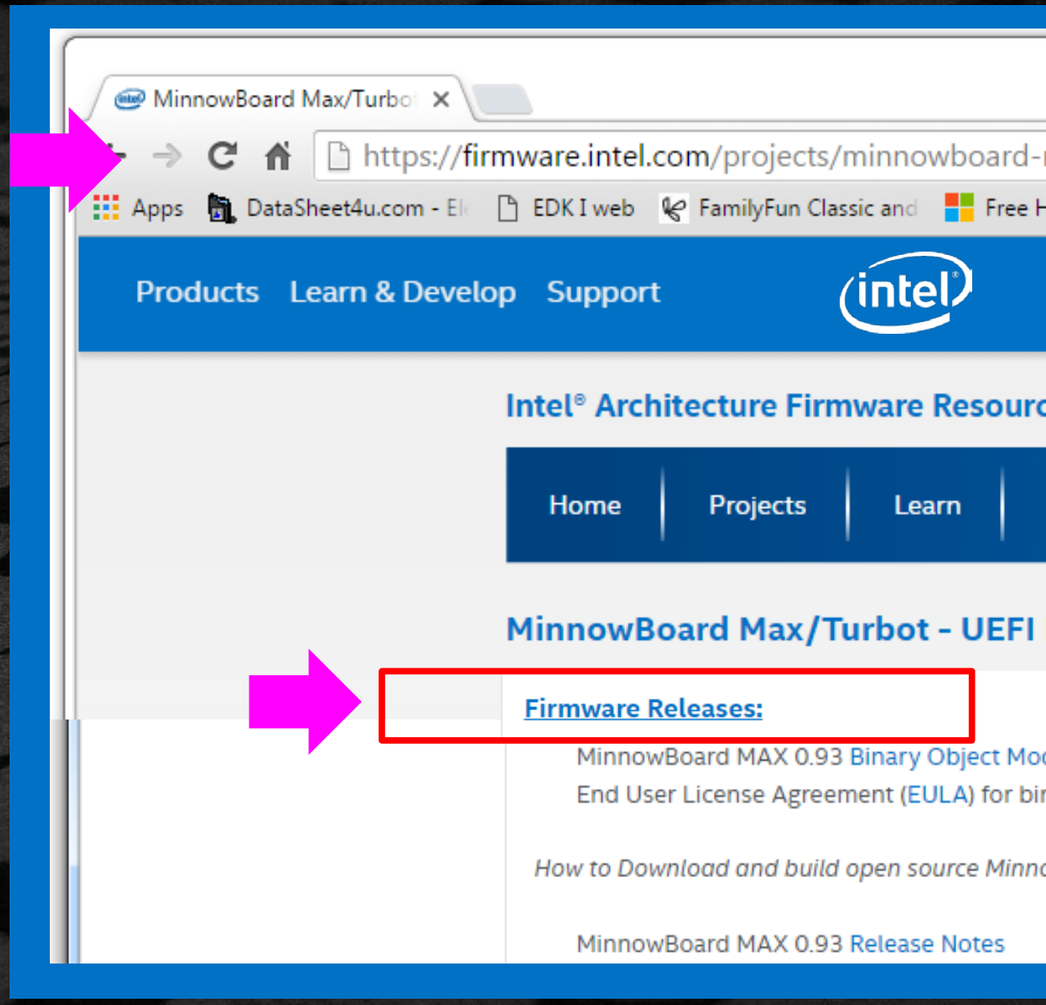
- Open Source Max Wiki
  - V.98 -Github Link
- Binary Object Modules firmware.intel.com
- How to Build: Release Notes



tianocore

Where to get Open Source MinnowBoard Max

- Open Source Max Wiki
  - V.98 -Github Link
- Binary Object Modules firmware.intel.com
- How to Build: Release Notes



DOWN LOAD MAX LAB SOURCE

Download the Lab_Material_FW.zip from : github.com PlatformBuildLab_FW.zip

OR

Use **`git clone`** to download the PlatformBuildLab_FW

```
bash$ cd $HOME
bash$ git clone https://github.com/Laurie0131/PlatformBuildLab_FW.git
```

Directory Lab_Material_FW will be created

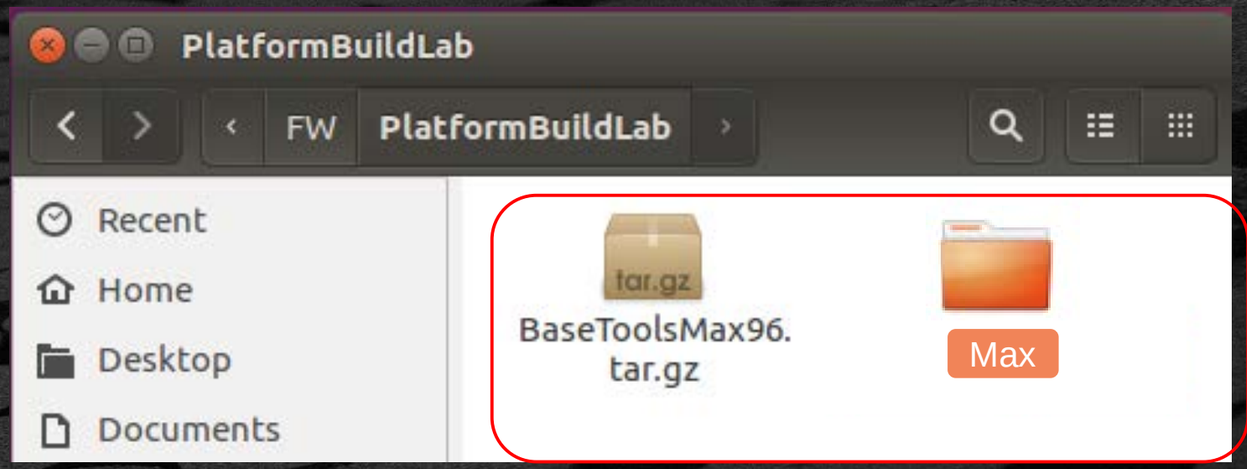
```
FW
- PlatformBuildLab
        - Max                              - Minnowboard Max Source for the Labs
        - BaseToolsMax.tar.gz              - BaseTools for Linux GCC5 build
        - MinnowBoard.MAX.FirmwareUpdateX64.efi  - UEFI App to flash
```

MINNOWBOARD MAX LAB SETUP

Previous Lab Setup Requirements

```
bash$ sudo apt-get install build-essential uuid-dev iasl git gcc-5 nasm
```

Additional Lab Setup – . . ./FW/PlatformBuildLab



- Max – MinnowBoard Max Project source code
- BuildToolsMax.tar.gz – build tools for GCC compiler
- At Terminal prompt - Install Screen utility for Serial Console to run UEFI Shell

```
bash$ sudo apt-get install screen
```

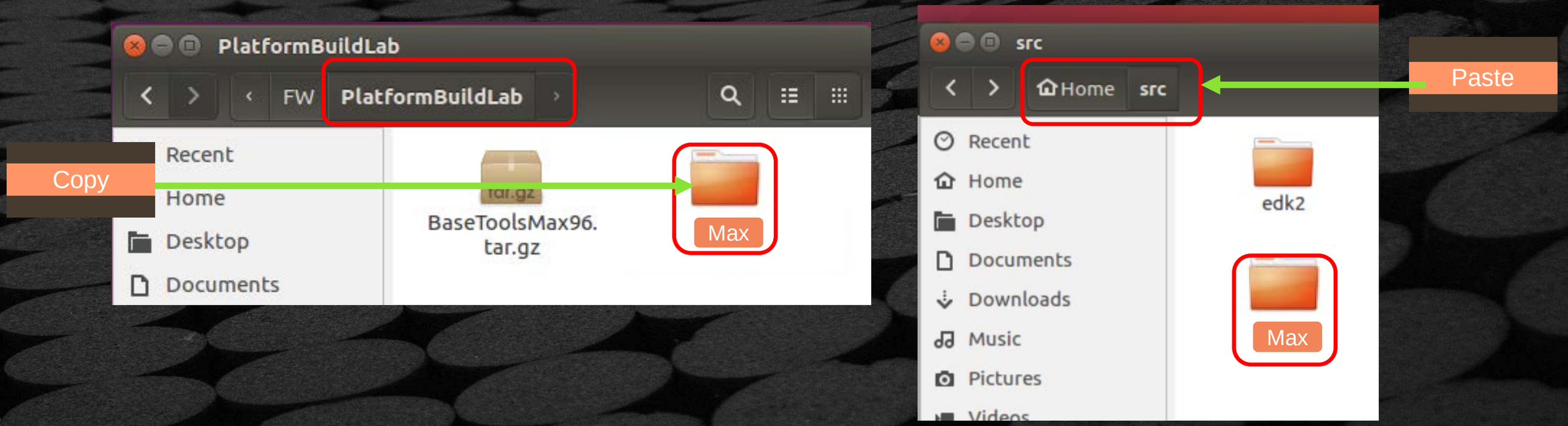
COPY MINNOWBOARD MAX SOURCE

Open a terminal prompt (Alt-Cnt-T)

Create a working space source directory under the home directory

```
bash$ mkdir ~src
```

From the FW/PlatformBuildLab folder, copy and paste folder “~FW/Max” to ~src

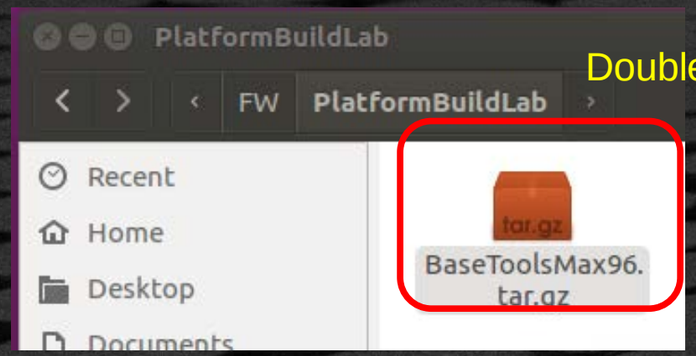


GET THE BASETOOLS FOR MAX

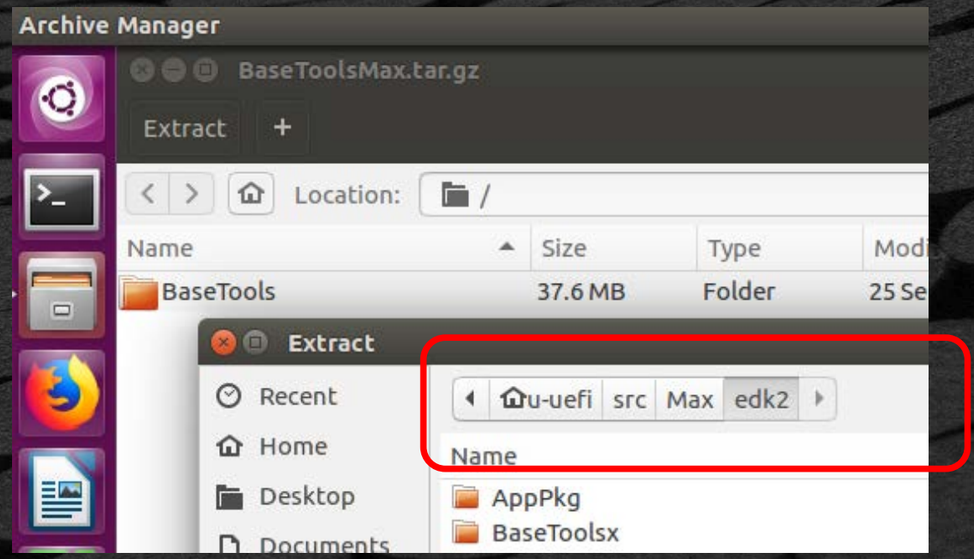
Rename or **mv** the directory "**~src/Max/edk2/BaseTools**"

```
bash$ cd ~src/Max/edk2
bash$ mv BaseTools BaseToolsX
```

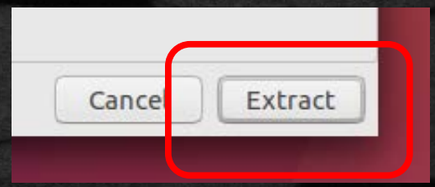
Extract the file **~FW/PlatformBuildLab/BaseToolsMax.tar.gz** to **~src/Max/edk2**



Double click



Select Extract directory $HOME/src/Max/edk2



click

PLATFORM SOURCE DIRECTORY STRUCTURE

```
./Max
    /edk2
        /(UDK2017 Directories)
        /BaseTools (from BaseToolsMax.tar.gz)
    /edk2-platforms
        /Vlv2DeviceRefCodePkg
        /Vlv2TbltDevicePkg
    /silicon
        /IA32FamilyCpuPkg
        /Vlv2BinaryPkg
        /Vlv2MiscBinariesPkg
```

Invoke the Build script from here

tianocore

STEPS TO BUILD & INSTALL FIRMWARE

1. Open Terminal prompt (Cnt-Alt-T)
2. Cd to project directory : $HOME/src/Max/edk2-platforms/Vlv2TbltDevicePkg
3. Invoke the build process
4. Locate build output (.BIN file for BIOS image)
5. Flash binary image onto the platform
6. Reset and boot new firmware to UEFI Shell

Next slide will follow the above steps

tianocore

Fix Script Properties to Execute

- Terminal prompt (Cnt-Alt-T)
- CD to workd space directory
- Fix Scipt files to “execute” with `chmod +x`

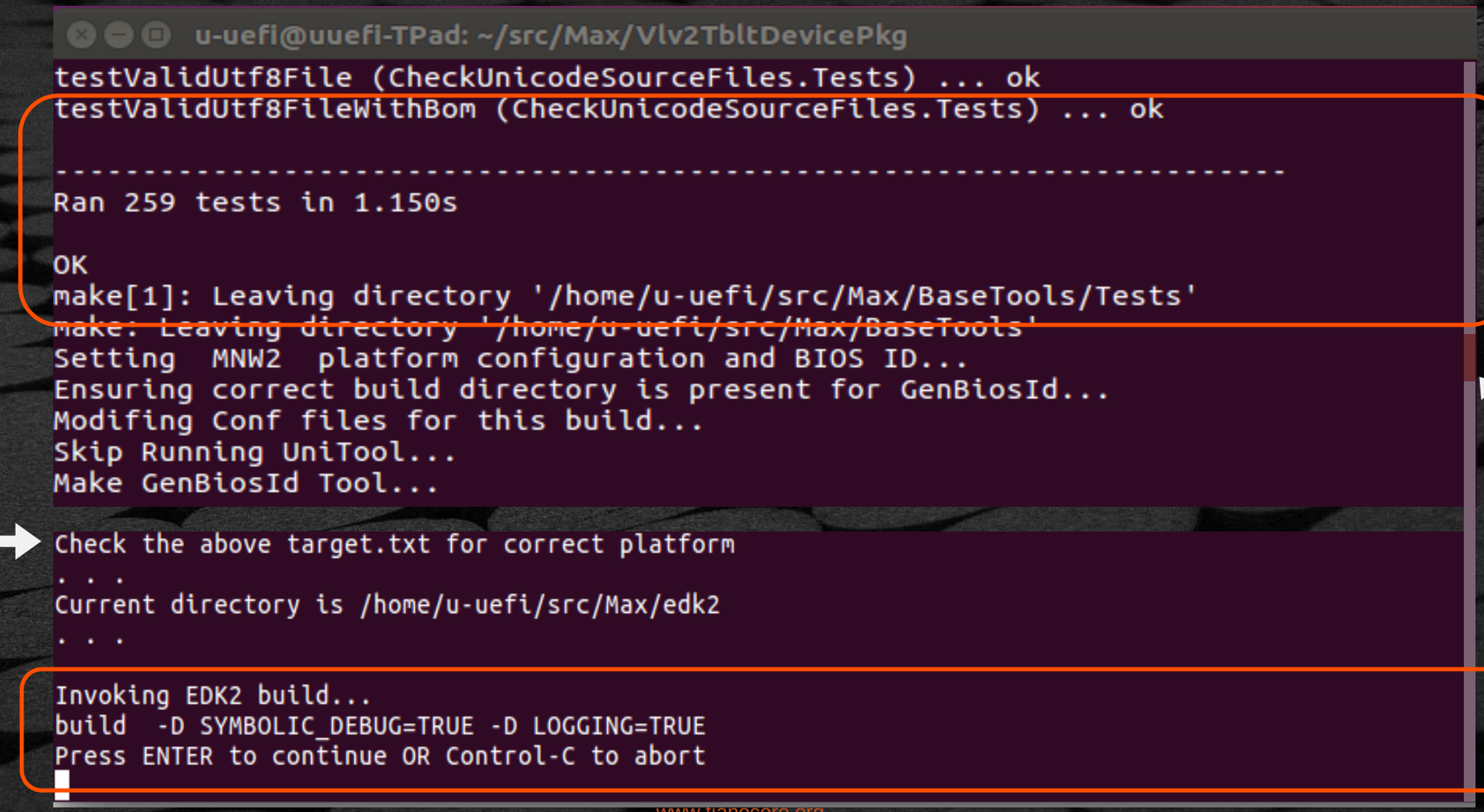
```
bash$ cd ~src/Max/edk2
bash$ chmod +x edksetup.sh
bash$ cd ~src/Max/edk2-platforms/
bash$ chmod +x Vlv2TbltDevicePkg/bld_vlv.sh
bash$ chmod +x Vlv2TbltDevicePkg/Build_IFWI.sh
bash$ chmod +x Vlv2TbltDevicePkg/GenBiosId
```

BUILD PROCESS FOR DEBUG

From Terminal Prompt enter:

Note: *the Build will Pause*

```
bash$ cd Vlv2TbltDevicePkg
bash$ . Build_IFWI.sh MNW2 Debug
```



1. Scroll up

2. Check MAKE for BaseTools OK

3. Check config…

4. Press ENTER to Continue

EXAMINE BUILD PARAMETERS

```
build
```

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

TARGET = DEBUG

Build mode

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

```
TARGET                = DEBUG
TARGET_ARCH           = IA32   X64
```

Build mode

CPU architecture

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

```
TARGET                = DEBUG
TARGET_ARCH           = IA32   X64
TOOL_CHAIN_TAG        = GCC5
```

Build mode

CPU architecture

Tool Chain GCC #

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

```
TARGET                = DEBUG
TARGET_ARCH           = IA32   X64
TOOL_CHAIN_TAG        = GCC5
ACTIVE_PLATFORM       =
      Vlv2TbltDevicePkg/PlatformPkgGccX64.dsc
```

Build mode
CPU architecture
Tool Chain GCC #
Platform (.DSC file)

EXAMINE BUILD PARAMETERS

```
build  -D SYMBOLIC_DEBUG=TRUE -D LOGGING=TRUE
          . . .   -D Option (n)
```

MACROS

Logging

Symbolic Debug

Properties from conf\Target.txt

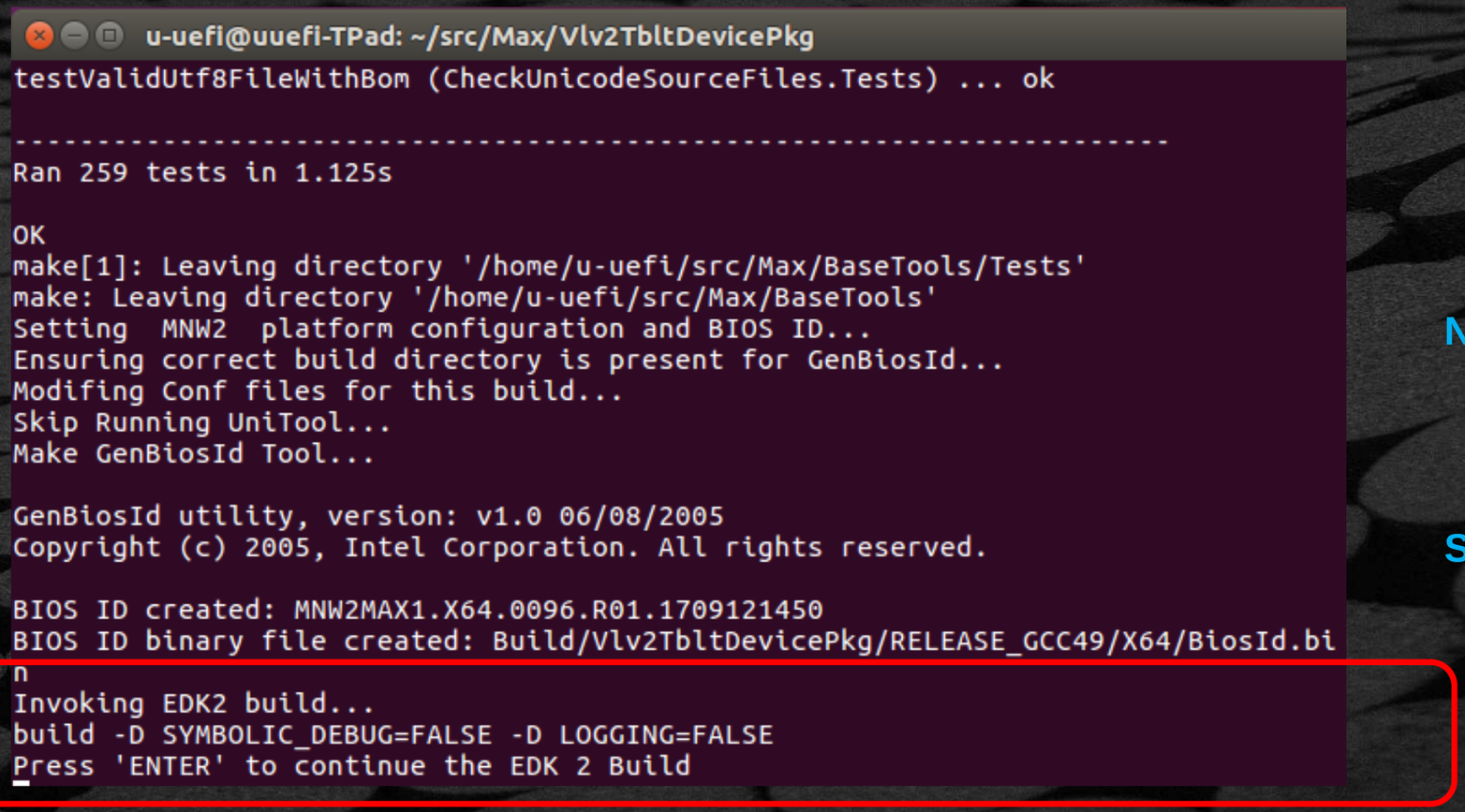
```
TARGET                = DEBUG
TARGET_ARCH           = IA32   X64
TOOL_CHAIN_TAG        = GCC5
ACTIVE_PLATFORM       =
      Vlv2TbltDevicePkg/PlatformPkgGccX64.dsc

MAX_CONCURRENT_THREAD_NUMBER = 1
```

Build mode
CPU architecture
Tool Chain GCC #
Platform (.DSC file)
Thread Count

BUILD PROCESS FOR RELEASE

From Terminal Prompt enter:

```
bash$ cd Vlv2TbltDevicePkg
bash$ . Build_IFWI.sh MNW2 Release
```



NOTE: MACROS

Logging

Symbolic Debug

Set to False

DEBUG & RELEASE DIFFERENCES

DEBUG has a slower boot than RELEASE
because of time it takes to display debug info

DEBUG & RELEASE DIFFERENCES

DEBUG has a slower boot than RELEASE
because of time it takes to display debug info

DEBUG has a larger image than RELEASE
because the embedded debug info

DEBUG & RELEASE DIFFERENCES

DEBUG has a slower boot than RELEASE
because of time it takes to display debug info

DEBUG has a larger image than RELEASE
because the embedded debug info

DEBUG uses the serial port for debug string output

DEBUG & RELEASE DIFFERENCES

DEBUG has a slower boot than RELEASE
because of time it takes to display debug info

DEBUG has a larger image than RELEASE
because the embedded debug info

DEBUG uses the serial port for debug string output

DEBUG contains the debug strings

tianocore

DEBUG & RELEASE DIFFERENCES

DEBUG has a slower boot than RELEASE
because of time it takes to display debug info

DEBUG has a larger image than RELEASE
because the embedded debug info

DEBUG uses the serial port for debug string output

DEBUG contains the debug strings

DEBUG contains detailed debug strings that show
the boot process and various ASSERT/TRACE errors

BUILD PROCESS COMPLETED

```
u-uefi@uuefi-TPad: ~/src/Max
- Done -
Build end time: 14:54:13, Sep.12 2017
Build total time: 00:03:43

./Vlv2TbltDevicePkg/Stitch/IFWIHeader/IFWI_HEADER.bin  is HEADER FILE
Skip Running fce...
Skip Running KeyEnroll...
Skip Running BIOS_Signing ...

Build location: Build/Vlv2TbltDevicePkg/RELEASE_GCC49
BIOS ROM Created: MNW2MAX_X64_F_0006_01.ROM

------------------- The EDKII BIOS build has successfully completed. ----------
----------


Finished Building BIOS.
================================================================================
Skip Build_IFWI:  Calling IFWI Stitching Script...

Build_IFWI is finished.
The final IFWI file is located in Stitch
================================================================================
u-uefi@uuefi-TPad:~/src/Max$
```

Build Process Completed

The EDK II build generates multiple firmware volumes, which are combined in the .BIN image.

FLASING THE NEW BIOS

1. Access Max Binary image file from build folder
   - **~src/Max/Vlv2TbltDevicePkg/Stitch**
   - **DEBUG MNW2MAX1_X64_D_0097_01_GCC.bin**
   - **RELEASE MNW2MAX1_X64_R_0097_01_GCC.bin**
2. Copy BIN files to a USB Thumb drive
3. Copy **MinnowBoard.MAX.FirmwareUpdateX64.efi** to a USB thumb drive from /FW/PlatformBuildLab
4. Use “Screen” and Boot into the UEFI Shell on MAX then type **“FS0:”**

```
bash$ screen /dev/ttyUSBn 115200
```

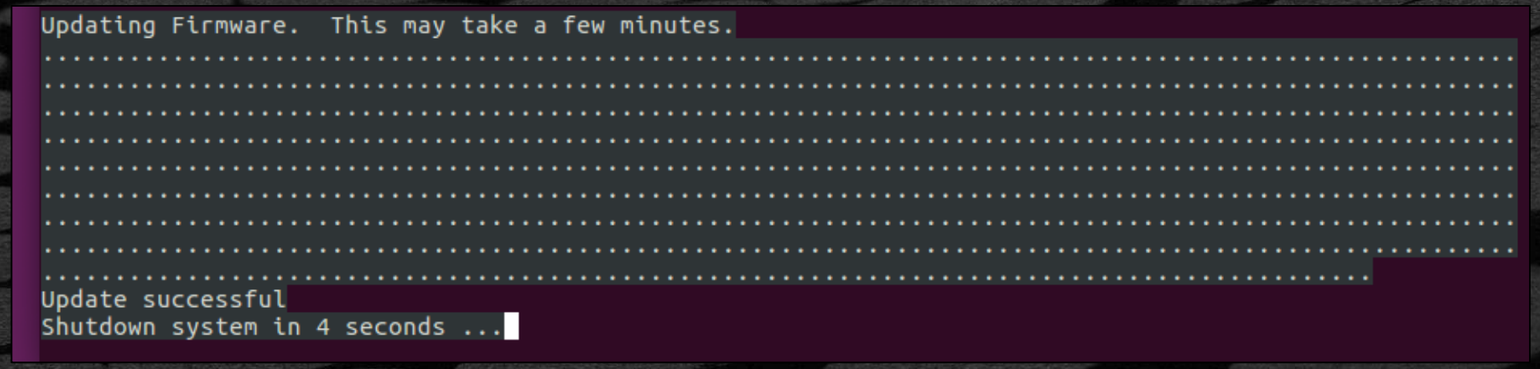
```
      FS0: Alias(s):HD8b0b::BLK1:
           PciRoot(0x0)/Pci(0x14,0x0)/USB(0x1,0x0)/HD(1,MBR,0x007EA5EF,0x50,0x3983AB)
     BLK0: Alias(s):
           PciRoot(0x0)/Pci(0x14,0x0)/USB(0x1,0x0)
Press ESC in 3 seconds to skip startup.nsh or any other key to continue.
Shell> fs0:
FS0:\>
```

FLASING THE NEW BIOS

5 Run update .efi utility with either BIN file
(Note the "*TAB*" Key will fill out the command line for you)

```
FS0:\> MinnowBoard.MAX.FirmwareUpdateX64.efi MNW2MAX1_X64_D_0096_01_GCC.bin
```

WAIT for the new firmware update to finish



6 Reset and boot new firmware

NOTE for Ubuntu Screen terminal
Control –H for backspace
Control A then D to return to Ubuntu
Screen –r to return to UEFI Shell Console

VERIFY AFTER FIRMWARE UPDATE

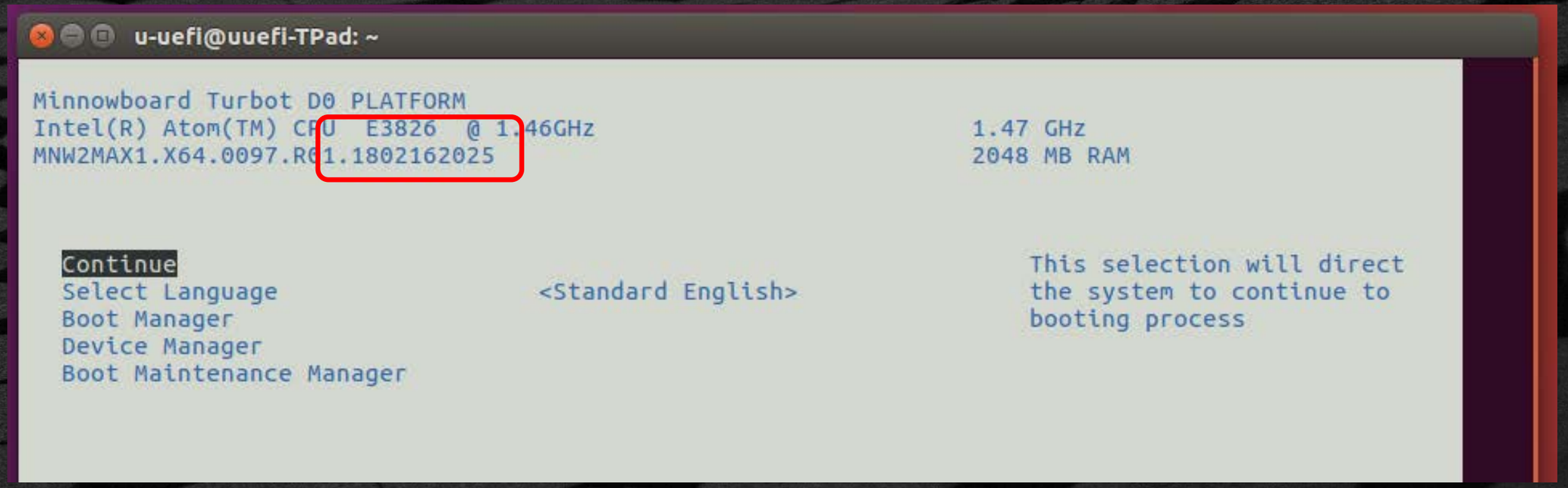
7

Verify that the Firmware was updated by checking the Date

At the shell prompt type “exit”

```
Shell>
Shell> exit
```

The EDK II front page will show the BIOS ID with Date/time stamp



tianocore

SUMMARY

- Lab 1: Build a EDK II Platform using OVMF package
- Lab 2: Hardware Setup for Minnowboard Max/Turbot
- Lab 3: Build a EDK II Platform using Minnowboard Max/Turbot

tianocore

Questions?

tianocore