



**APZU PetaLinux**

# **USER'S MANUAL**

**ACROMAG INCORPORATED**

**30765 South Wixom Road**

**Wixom, MI 48393-2417 U.S.A.**

**Tel: (248) 295-0310**

**Email: [solutions@acromag.com](mailto:solutions@acromag.com)**

Copyright 2021, Acromag, Inc., Printed in the USA.  
Data and specifications are subject to change without notice.

**8501175A**

# Table of Contents

---

## Table of Contents

---

<b>1.0</b>	<b>GENERAL INFORMATION.....</b>	<b>3</b>
1.1	Intended Audience .....	3
1.2	Preface .....	3
1.3	Trademark, Trade Name and Copyright Information .....	3
<b>2.0</b>	<b>SETUP VIRTUALBOX VIRTUAL MACHINE .....</b>	<b>4</b>
2.1	Host PC Installation Requirements for PetaLinux.....	4
2.2	Installation of Oracle VirtualBox .....	4
2.3	Installation of Ubuntu Linux 16.04.5 .....	7
2.4	Installation of Petalinux.....	12
<b>3.0</b>	<b>CREATE PETALINUX APPLICATION .....</b>	<b>14</b>
3.1	petalinux-create .....	14
3.2	petalinux-config.....	14
3.3	petalinux-create and config with BSP .....	15
3.4	petalinux-build .....	19
3.5	petalinux-package .....	20
3.6	Adding Files to SD Card .....	20
3.7	Boot Process.....	21
3.8	Create FAT Partition SD Card.....	22
3.9	Create the PetaLinux helloworld application .....	25
3.10	Running the helloworld application .....	29

---

---

3.11	Linux GPIO.....	31
3.12	Access to PL Digital I/O Ports .....	33
3.13	Test of Ethernet.....	38
3.14	Test of USB .....	39
3.15	Test of I2C .....	42
REVISION HISTORY .....		43

## 1.0 GENERAL INFORMATION

---

### 1.1 Intended Audience

This users' manual was written for technically qualified personnel who will be working with I/O devices using the AcroPack module. It is not intended for a general, non-technical audience that is unfamiliar with I/O devices and their application.

### 1.2 Preface

The information contained in this manual is subject to change without notice, and Acromag, Inc. (Acromag) does not guarantee its accuracy. Acromag makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Further, Acromag assumes no responsibility for any errors that may appear in this manual and makes no commitment to update, or keep current, the information contained in this manual. No part of this manual may be copied or reproduced in any form, without the prior written consent of Acromag,

### 1.3 Trademark, Trade Name and Copyright Information

© 2021 by Acromag Incorporated.

All rights reserved. Acromag and Xembedded are registered trademarks of Acromag Incorporated. All other trademarks, registered trademarks, trade names, and service marks are the property of their respective owners.

## 2.0 Setup VirtualBox Virtual Machine

### 2.1 Host PC Installation Requirements for PetaLinux

#### Installation Requirements

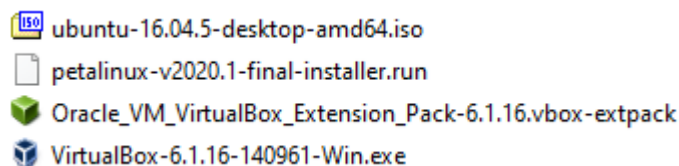
The PetaLinux tools installation requirements are:

- Minimum workstation requirements:
  - 8 GB RAM (recommended minimum for Xilinx® tools)
  - 2 GHz CPU clock or equivalent (minimum of eight cores)
  - 100 GB free HDD space
  - Supported OS:
    - Red Hat Enterprise Workstation/Server 7.4, 7.5, 7.6, 7.7, 7.8 (64-bit)
    - CentOS Workstation/Server 7.4, 7.5, 7.6, 7.7, 7.8 (64-bit)
    - Ubuntu Linux Workstation/Server 16.04.5, 16.04.6, 18.04.1, 18.04.2, 18.04.3, 18.04.4 (64-bit)

Recommended:

Ubuntu Linux 16.04.5

The following applications will need to be downloaded to your computer:



**Note:** You can have Vivado tools set up on a different machine; it is not necessary to have PetaLinux and Vivado tools set up on the same machine.

### 2.2 Installation of Oracle VirtualBox

Make sure the Extension Pack you download is the same version as your VirtualBox installer.

#### Acromag used VirtualBox 6.1.16 platform packages

VirtualBox-6.1.16-140961-Win.exe

Also downloaded

#### VirtualBox 6.1.16 Oracle VM VirtualBox Extension Pack

Oracle\_VM\_VirtualBox\_Extension\_Pack-6.1.16.vbox-extpack

Downloads <https://www.virtualbox.org/wiki/Downloads>

See VirtualBoxUserManual.pdf for detail installation instructions.

To enable your CPU virtualization extensions, you will need to reboot your PC and enter the (BIOS or UEFI) configuration menu.

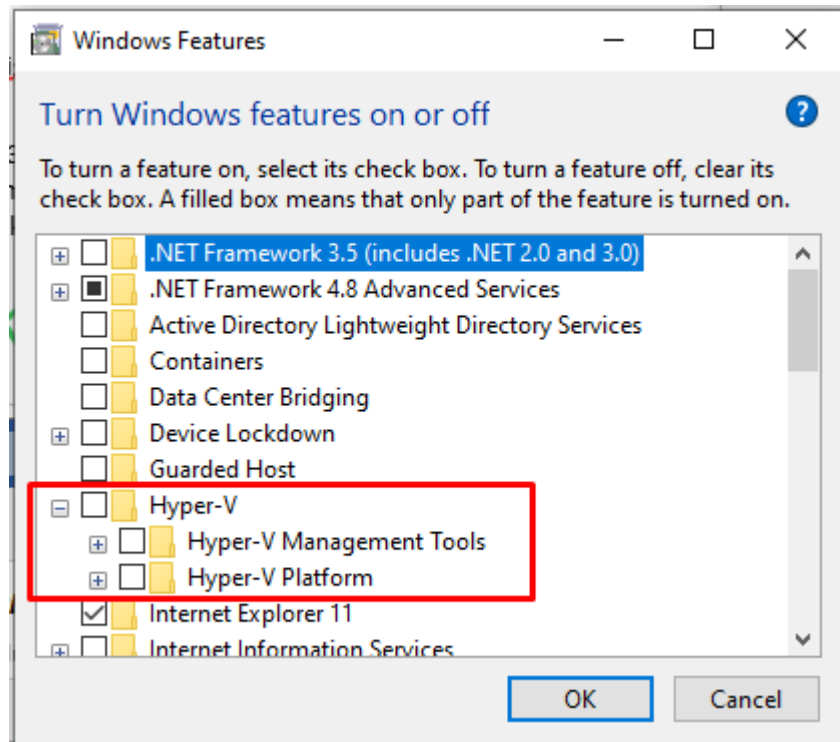
Once in the configuration screen, look for the **Intel VT-x**, **Intel Virtualization Technology**, or **Virtualization Extensions** options and make sure that they are enabled. Could be under Security -> Virtualization.

#### Disabling Windows 10 Hyper-V:

Windows 10 Pro includes Microsoft provided VM enablement called Hyper-

V. However, Hyper-V and VirtualBox cannot be used concurrently. The Hyper-V feature of Windows needs to be disabled to use VirtualBox.

The following dialog is where many Windows 10 features can be enabled/disabled, you can find it from the Win 10 search bar with the “Windows Features” search string and Hyper-V can be disabled by removing the checkmarks next to these features:



Launch the VirtualBox installer from Windows Explorer by double-clicking the self-extracting executable (VirtualBox-6.1.a6-140961-Win.exe).

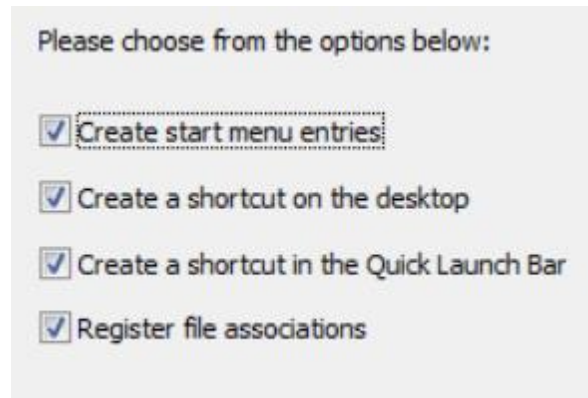
Allow the installer to make changes to your computer, if so prompted.

The installation Welcome dialog is displayed and enables you to choose where to install Oracle VM VirtualBox, and which components to install. In addition to the Oracle VM VirtualBox application, the following components are available: USB support, Networking, and Python support.

You may accept all the installation defaults.

If the options are acceptable, select the **Next** button.

Accept the default options shown below and select the **Next** button.



Select **Yes** button to continue with the installation.

Select **Install** to begin the installation.

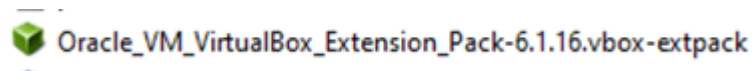
If prompted, allow the installer to make changes to your system, including installation of the USB interface and Network adapters.

Select the **Finish** button to complete the installation.

Once VirtualBox starts (you can also start it from the Desktop shortcut, or the Windows Start button), the Extension Pack must be added. From the main menu, select **File > Preferences**.

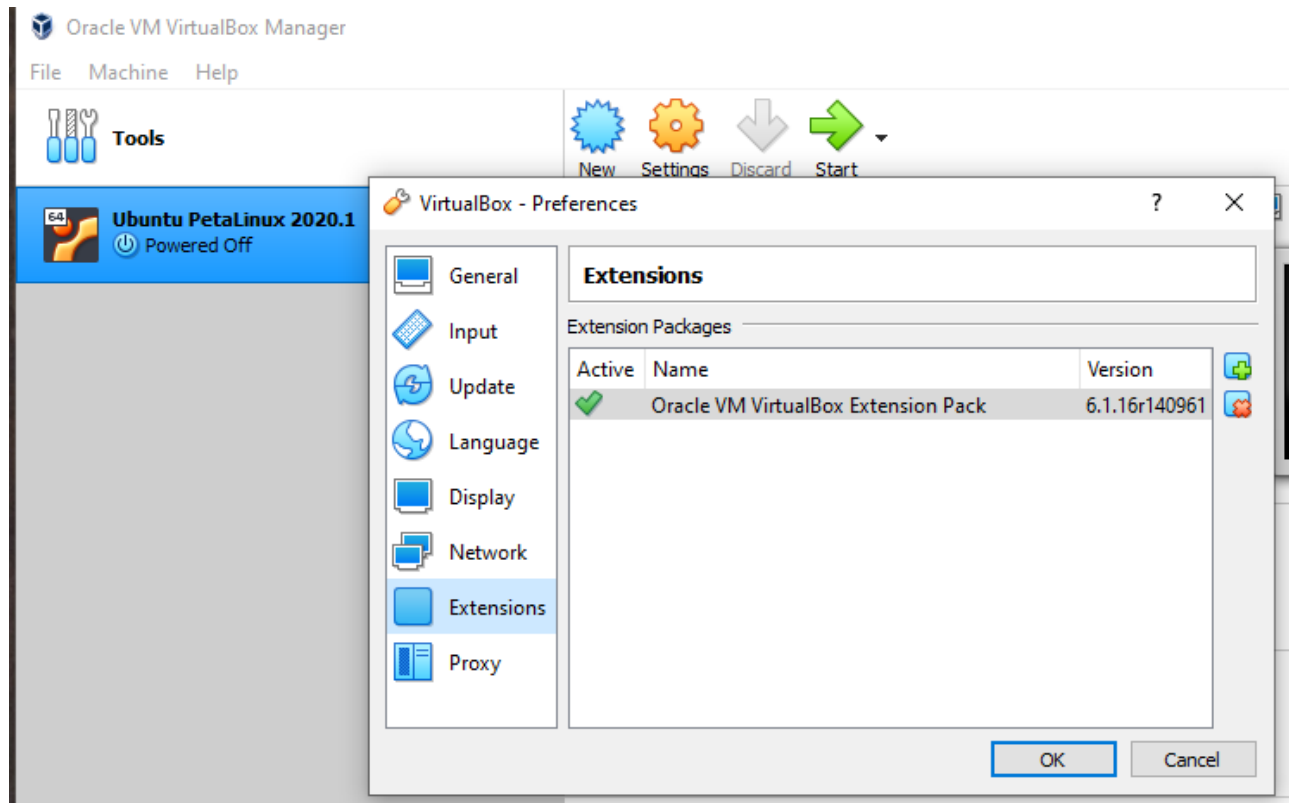
Select **Extensions**. Right-select in the *Extension Packages* whitespace box and select **Add Package**.

Browse to the location where you downloaded the VirtualBox Extension Pack compatible with your VirtualBox version. Select the Extension Pack and click the **Open** button.



Select the **Install** button to add the VirtualBox Extension Pack.

Then select the **OK** button to return to VirtualBox.

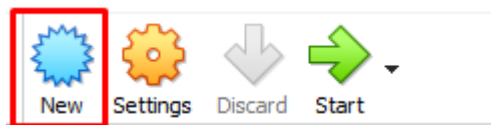


This completes the installation of VirtualBox on your host development system. VirtualBox is now ready to accept a new Virtual Machine.

## 2.3 Installation of Ubuntu Linux 16.04.5

Create a New Virtual Machine

Launch Oracle VM VirtualBox Manager and select the **New** icon at the upper left.



Select a descriptive name like Ubuntu PetaLinux 2020.1 for the VM. Set the *Type* to **Linux** and the *Version* to **Ubuntu (64-bit)**.

Select the amount of memory to be allocated to the Virtual Machine. Allocating more memory will improve the VM performance, but you must leave enough memory available for your host system for all other concurrent processes. For a host system with 16 GB of RAM, a value of **8192 MB** is recommended for the Virtual Machine.



Select the **Create a virtual hard disk now** radio button.

Select the **Create** button.

Select the **VDI (VirtualBox Disk Image)** radio button.

Select the **Next** button.

Select the **Fixed size** radio button.

Select the **Next** button.

Select the name and location for the Virtual Machine within your host file system. The recommended size is at least **300.00** GB.

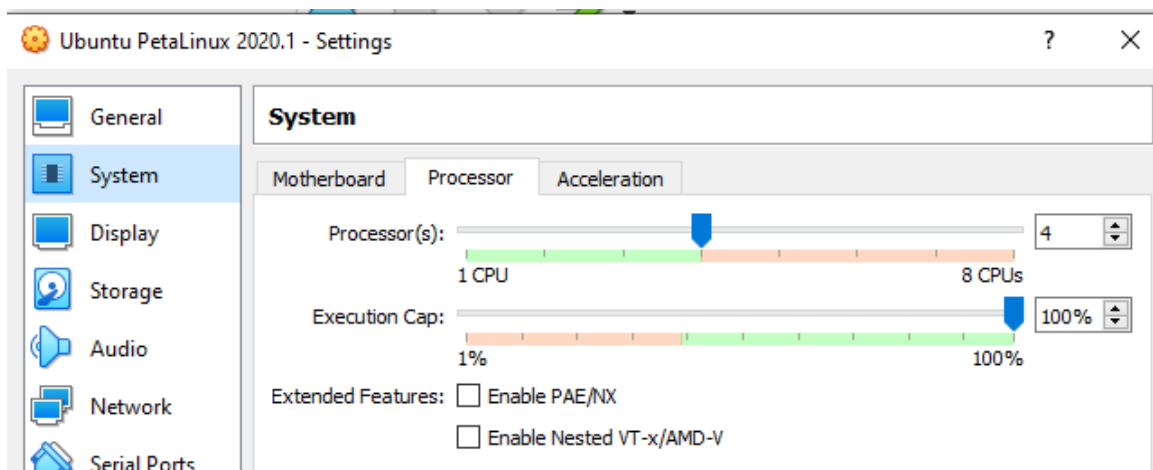
Select the **Create** button.

After the Virtual Disk completes, select the newly created VM.

Select **Settings** button.

Under the System blade and Processor tab, select the number of logical CPU cores to be allocated to the Virtual Machine.

For a host system with 8 CPU cores, a value of 4 CPU cores is recommended for the Virtual Machine.



The VM is now ready to accept an operating system.

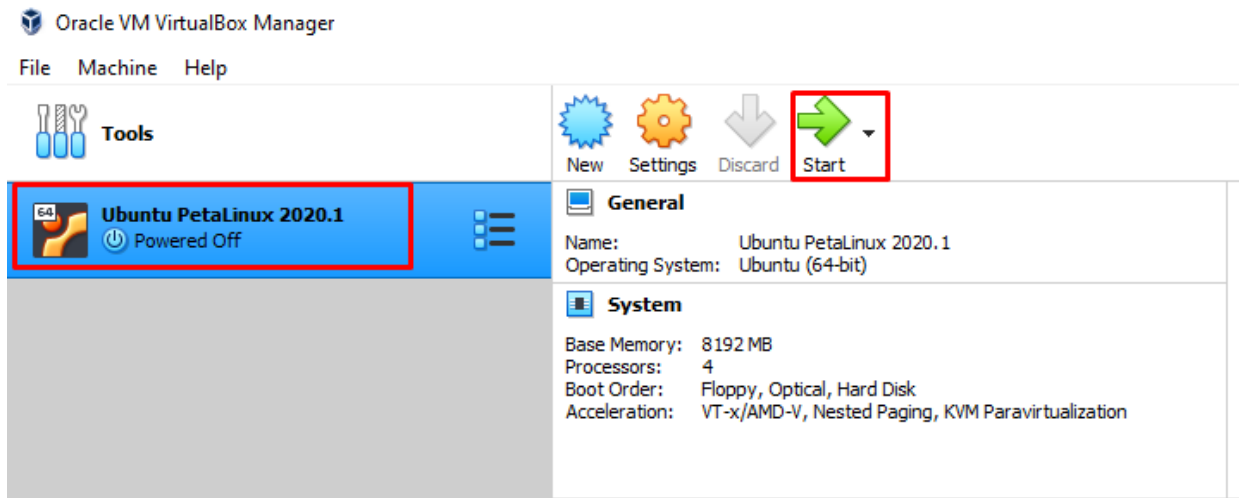
Download Ubuntu 16.04.5-desktop-amd64.iso from the following site:

[http://old-releases.ubuntu.com/releases/16.04.5/?\\_ga=2.219481497.1684163097.1608763228-1979948035.1604077425](http://old-releases.ubuntu.com/releases/16.04.5/?_ga=2.219481497.1684163097.1608763228-1979948035.1604077425)

Note that the Ubuntu image is in .iso format as needed.

Launch VirtualBox and select the VM (Ubuntu PetaLinux 2020.1) as shown below.

Select the **Start** arrow.



Select the Browse icon (folder with green up arrow) and locate the Ubuntu 16.04.5-desktop-amd64.iso image.

Select the **Start** button.

At the install screen select **English**.

Select **install Ubuntu** button.

At the Preparing to install Ubuntu dialog leave the two options unchecked.

Select the **Continue** button.

At the Installation type dialog select **Erase disk and install Ubuntu**.

Select the **Install Now** button.

At the Write the changes to disks? dialog select **Continue** button.

Select your time zone.

Select **Continue** button.

Select Keyboard layout option.

Select **Continue** button.

Enter your primary user name for the Virtual Machine in field "Your name".

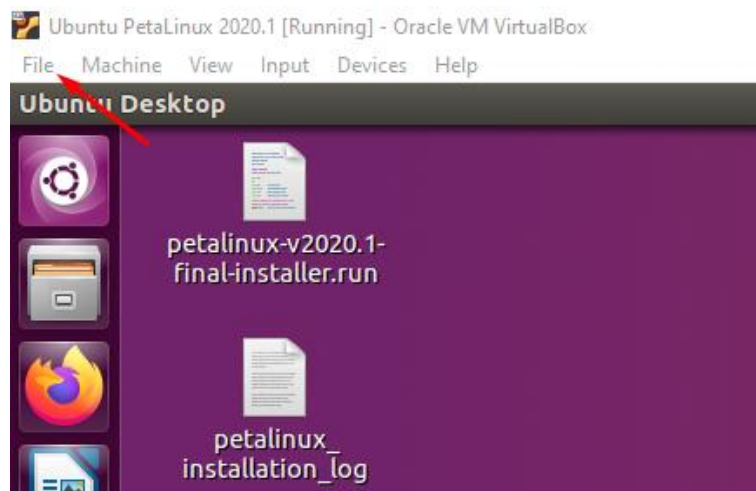
Enter and confirm your password.

Select **Continue** button.

When the installation is complete, select **Restart Now** button.

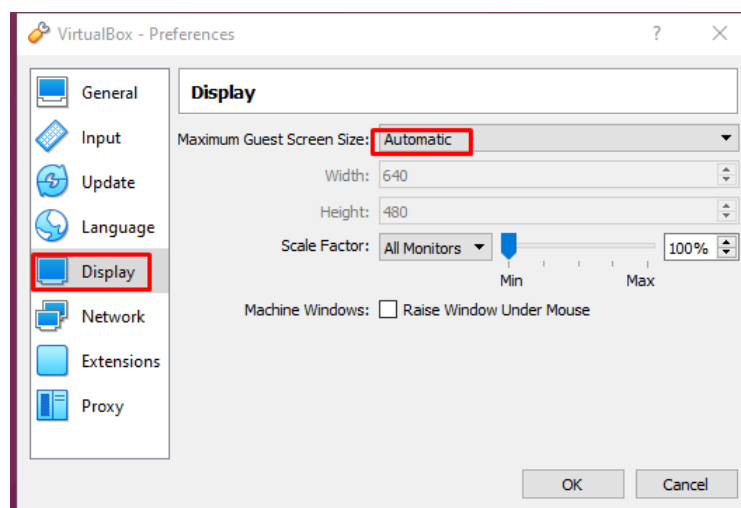
Open VM VirtualBox Manager and select Ubuntu PetaLinux 2020.1 and then select Start arrow.

Sign in and select File -> Preferences at the Ubuntu PetaLinux 2020.1 dialog

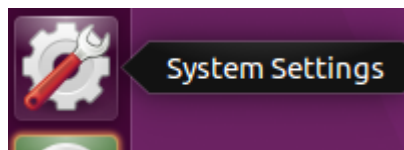


Select the **Display** blade and set the Maximum Guest Screen size to **Automatic**.

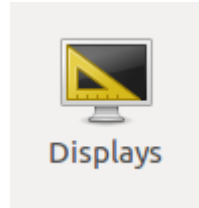
Select **OK**.



Set the display resolution by selecting the system Settings icon.



Then select the Displays Icon

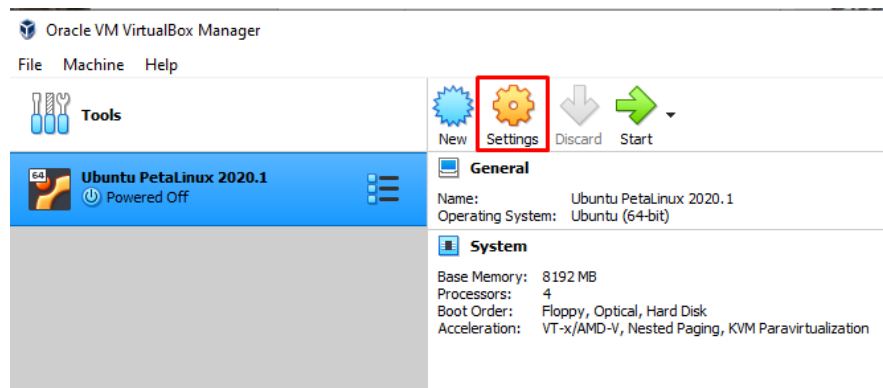


Adjust the resolution as desired.

## Network Bridging

To allow your Virtual Machine to accept an address from a local DHCP server, you can change the default network type to Bridge.

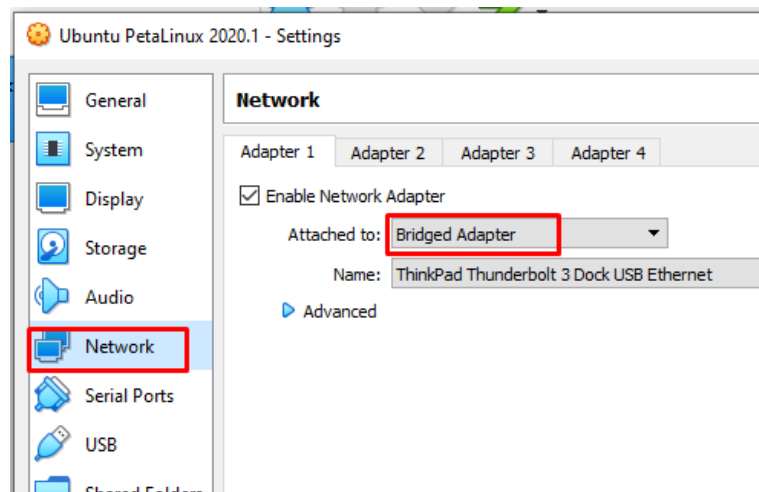
Select the **Settings** Icon or **Machine -> Settings**.



Select the **Network** entry.

Select **Bridge Adapter** from the dropdown menu.

Select **OK**.



## Set a root user password

By default, Ubuntu does not set a password for the root user. You can do this by simply invoking the **sudo passwd** command. You supply your own user password, then set the root user password.

```
sudo passwd
<Enter user password>
<Enter new root password>
<Confirm new root password>
```

From this point forward, you will be able to precede a command with `sudo` to obtain root authority.

## 2.4 Installation of PetaLinux

PetaLinux requires a number of standard development tools and libraries to be installed on your Linux host workstation. PetaLinux 2020.1 require some additional system tools and libraries to be installed on the host system. For the list of required packages for the PetaLinux, refer to Xilinx UG1144 document.

See Xilinx Master Answer record 73296 PetaLinux: How to install the required packages for the PetaLinux Build Host?

The script attached to Answer Record 73296 will install the required packages for the PetaLinux Build Host.

Change the default shell to  
bash for PetaLinux

With Ubuntu, PetaLinux tools require that your host system `/bin/sh` is 'bash'.

Determine the current shell setting for your Ubuntu environment.

Enter **\$ ls -l /bin/sh**

Change the default shell for all terminal window with the following.

Enter **\$ sudo dpkg-reconfigure dash**

Select **No** to remove dash as the default system shell.

Verify set to bash by entering

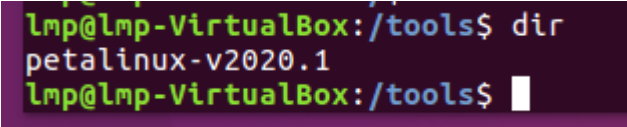
**\$ ls -l /bin/sh**

Adjusting Permissions

The PetaLinux tools require the installation as a non-root user so the permissions of these directories under the `/opt` folder must be adjusted accordingly.

This can be accomplished in a terminal window with the following commands:

```
$ sudo chmod ugo+w /tools/
$ sudo mkdir /tools/petalinux-v2020.1
$ sudo chmod -R ugo+w /tools/petalinux-v2020.1
```



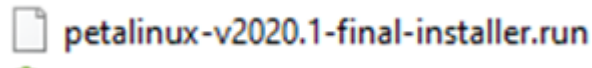
A terminal window screenshot showing the execution of the `dir` command in the `/tools` directory. The output shows the directory `petalinux-v2020.1` has been created. The prompt is `lmp@lmp-VirtualBox:/tools$`.

## Install PetaLinux

The Imp:Imp is unique to our system setup. Please change as needed.

**\$ sudo chown -R Imp:Imp /tools/petalinux-v2020.1**

PetaLinux 2020.1 can be downloaded from Xilinx's website.



Copy the PetaLinux installer to your VM desktop.

Open a terminal window.

Press Ctrl + Alt + t simultaneously to open a new terminal.

Launch the installer.

**\$ sudo chmod ugo+rw Desktop**

**\$ cd ~/Desktop**

**\$ .sudo chmod 755 petalinux-v2020.1-final-installer.run**

**\$ ./petalinux-v2020.1-final-installer.run --dir /tools/petalinuxv2020.1**

Accept the license conditions.

Source the **/tools/petalinux-v2020.1/settings.sh** script prior to attempting to use the PetaLinux tools for development.

**\$ source /tools/petalinux-v2020.1/settings.sh**

To permanently set this environment variable for the all future terminal sessions, include the above command near the top of the **~/bashrc** file using your favorite editor.

### 3.0 Create PetaLinux Application

PetaLinux includes and manages the Linux kernel sources and libraries. There are eight PetaLinux commands:

- **petalinux-create** (create hardware platform)
- **petalinux-config** (configure kernel, applications, system settings)
- **petalinux-build** (build kernel, file image system)
- petalinux-util
- **petalinux-package** (package boot file with fsbl, bit file etc.)
- petalinux-upgrade
- petalinux-devtool
- petalinux-boot

The petalinux commands in bold will be used in the build of the example application that follows. If you are creating a new project from the Acromag provided BSP go to section 3.3 petalinux-create and config with BSP else continue to section 3.1 petalinux-create.

#### Xilinx PetaLinux User Guide

The complete Xilinx PetaLinux User Guide (ug1144) is available from Xilinx <http://www.xilinx.com/cgibin/docs/rdoc?v=latest;d=ug1144-petalinux-tools-reference-guide.pdf>

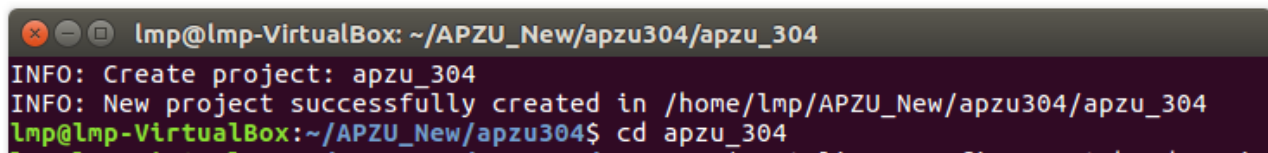
### 3.1 petalinux-create

#### Create New PetaLinux Project

Creating a project from scratch and not using the Acromag provided BSP will require also running through the steps given in section 3.9 PetaLinux helloworld application. When creating a project from scratch sections 3.1, 3.2 are followed. When creating a project from the Acromag BSP sections 3.1 and 3.2 are skipped and the project creation process is started in section 3.3.

The following will create an apzu\_304 directory under the current project directory.

**\$ petalinux-create --type project --name apzu\_304 --template zynqMP**

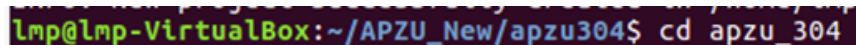


```
lmp@lmp-VirtualBox: ~/APZU_New/apzu304/apzu_304
INFO: Create project: apzu_304
INFO: New project successfully created in /home/lmp/APZU_New/apzu304/apzu_304
lmp@lmp-VirtualBox:~/APZU_New/apzu304$ cd apzu_304
```

Continue with petalinux-config as given in the next section.

### 3.2 petalinux-config

The following will configure the design with the hardware xsa file generated by vivado. The APZU\_top.xsa must be one directory above the apzu\_304 directory as seen below. In this case the APZU\_top.xsa must reside in the ~/APZU\_NEW/apzu304 directory.



```
lmp@lmp-VirtualBox:~/APZU_New/apzu304$ cd apzu_304
```

```
$ cd apzu_304
```

```
$ petalinux-config --get-hw-description=../
```

After running petalinux-config, the PetaLinux project has been created based on the .xsa file.

PetaLinux includes an automated device tree generator. The device tree based on hardware design file APZU\_top.xsa.

By inspecting device-tree you can look at what PetaLinux has detected in the .xsa file.

PetaLinux includes and manages the Linux kernel sources and libraries.

Kernel build process compiles DTS and DTB and links into kernel image.

The petalinux generated DTS files in the following directory.

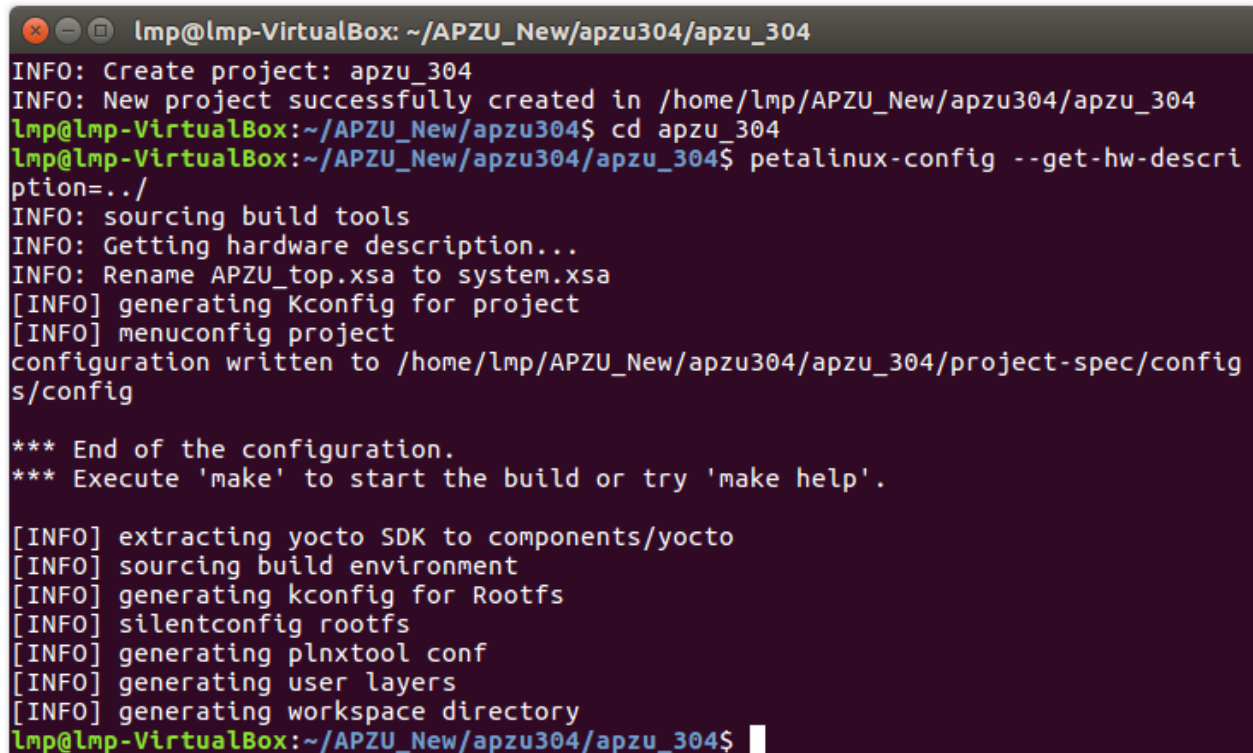
<project>/components/plnx\_workspace/device-tree/device-tree/

Do not edit the auto generated device-tree files.

User edits are allowed to the system-user.dtsi DTS file in

<project>/project-spec/meta-user/recipes-bsp/device-tree/files/

This allows user overwrite of default parameters and addition of new parameters. Changes to the system-user.dtsi will not be overwritten by subsequent petalinux-config. Skip to section 3.4 petalinux-build after completing this petalinux-config step.



```
lmp@lmp-VirtualBox: ~/APZU_New/apzu304/apzu_304
INFO: Create project: apzu_304
INFO: New project successfully created in /home/lmp/APZU_New/apzu304/apzu_304
lmp@lmp-VirtualBox:~/APZU_New/apzu304$ cd apzu_304
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304$ petalinux-config --get-hw-description=../
INFO: sourcing build tools
INFO: Getting hardware description...
INFO: Rename APZU_top.xsa to system.xsa
[INFO] generating Kconfig for project
[INFO] menuconfig project
configuration written to /home/lmp/APZU_New/apzu304/apzu_304/project-spec/configs/config
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] extracting yocto SDK to components/yocto
[INFO] sourcing build environment
[INFO] generating kconfig for Rootfs
[INFO] silentconfig rootfs
[INFO] generating plnxtool conf
[INFO] generating user layers
[INFO] generating workspace directory
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304$
```

Continue with petalinux-build as given in section 3.4.

### 3.3 petalinux-create and config with BSP

Acromag provided a BSP (board support package) file for each of the APZU boards the APZU 301, 303 and 304. The BSP contains the project application helloworld as provided by Acromag. In order to generate the petalinux boot



files with this helloworld application, the steps given here should be followed.

Start by creating a new project folder. For this example, the apzu301t folder will be created in our existing APZU\_New folder. Copy the apzu301\_BSP.bsp and APZU\_top.xsa corresponding to the APZU 301 to the apzu301t folder. Then run the following petalinux-create command.

```
$ petalinux-create -t project -n apzu301 -s ~/APZU_New/apzu301t/apzu301_BSP.bsp
```

```
lmp@lmp-VirtualBox:~/APZU_New/apzu301t$ dir
apzu301_BSP.bsp  APZU_top.xsa
lmp@lmp-VirtualBox:~/APZU_New/apzu301t$ petalinux-create -t project -n apzu301 -
s ~/APZU_New/apzu301t/apzu301_BSP.bsp
INFO: Create project: apzu301
INFO: New project successfully created in /home/lmp/APZU_New/apzu301t/apzu301
lmp@lmp-VirtualBox:~/APZU_New/apzu301t$
```

This will create a project referencing the original BSP Vivado design in this case for the apzu301 board. The helloworld application will also be provided as can be seen in the following path:

```
~/APZU_New/apzu301t/apzu301/project-spec/meta-user/recipes-apps/helloworld
```

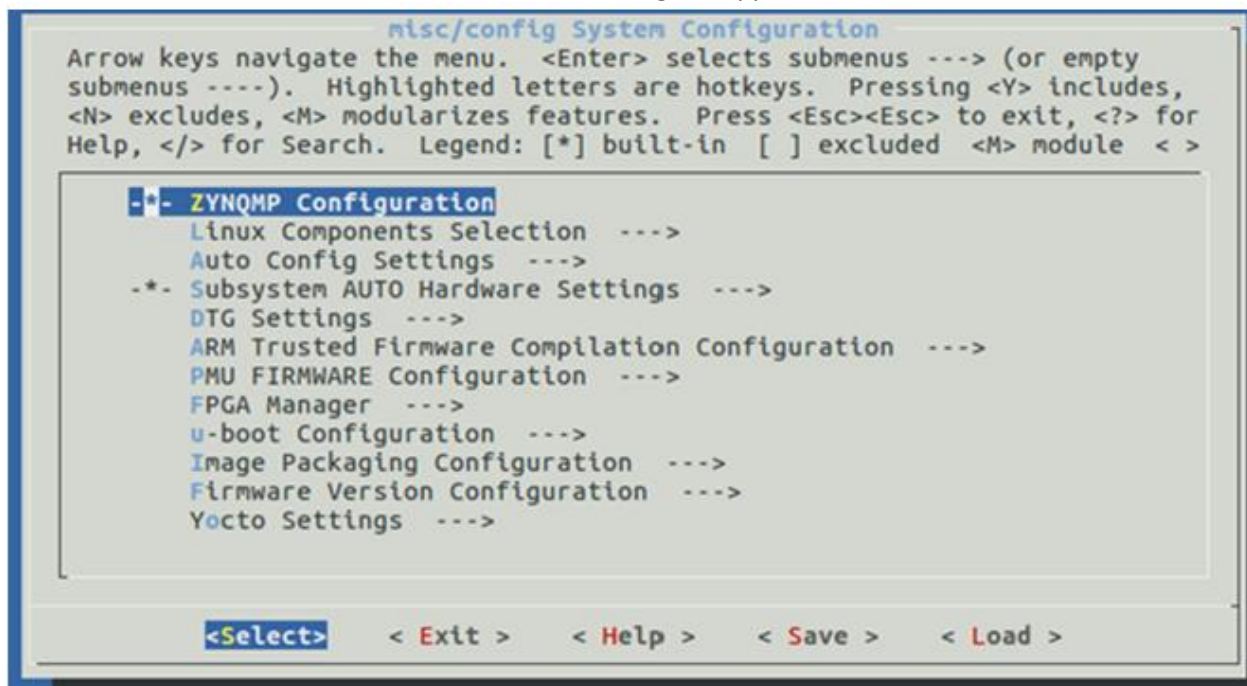
The step to update the design with the xsa file is still needed. This will be done in petalinux-config. Change directories to the apzu301 folder that was created in the previous step.

```
$ cd apzu301
```

Run the following petalinux-config command:

```
$ petalinux-config --get-hw-description=~/APZU_New/apzu301t/APZU_top.xsa
```

Running the following petalinux-config commands will take time and cause your primary system to lag. After running this first petalinux-config command, the following will appear on the screen.



Hit the Esc key two times. Then, if prompted select Yes. Info statements similar to the following will appear in the terminal.

```
lmp@lmp-VirtualBox:~/APZU_New/apzu301t/apzu301$ petalinux-config --get-hw-description=../
INFO: sourcing build tools
INFO: Getting hardware description...
INFO: Rename APZU_top.xsa to system.xsa
[INFO] generating Kconfig for project
[INFO] menuconfig project

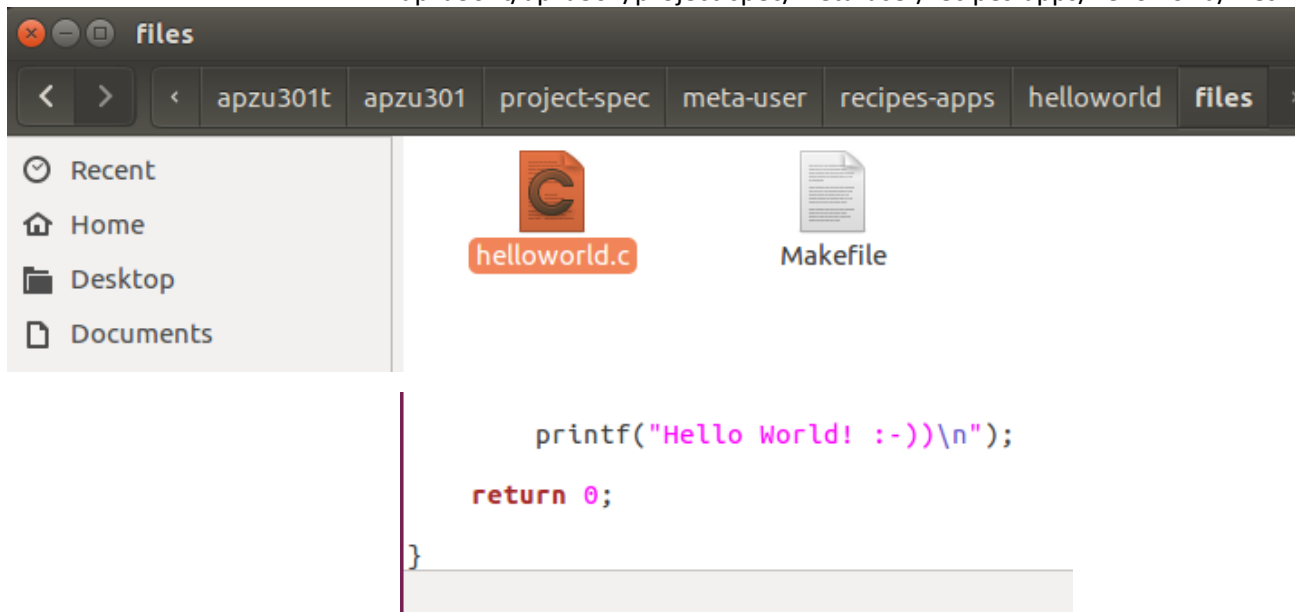
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] extracting yocto SDK to components/yocto
[INFO] sourcing build environment
[INFO] generating kconfig for Rootfs
[INFO] silentconfig rootfs
[INFO] generating plnxtool conf
[INFO] generating user layers
[INFO] generating workspace directory
lmp@lmp-VirtualBox:~/APZU_New/apzu301t/apzu301$
```

It should be noted that the created helloworld application already contains the Acromag generated c-file. This will be the application that we run. If desired, the file can be opened and modified. For this example, we will modify it by adding another right bracket “:-))” to it at the end of the file as seen below.

You can find the helloworld.c file in the following directory:

apzu301t/apzu301/project-spec/meta-user/recipes-apps/helloworld/files



Save the changes made to the helloworld.c file.

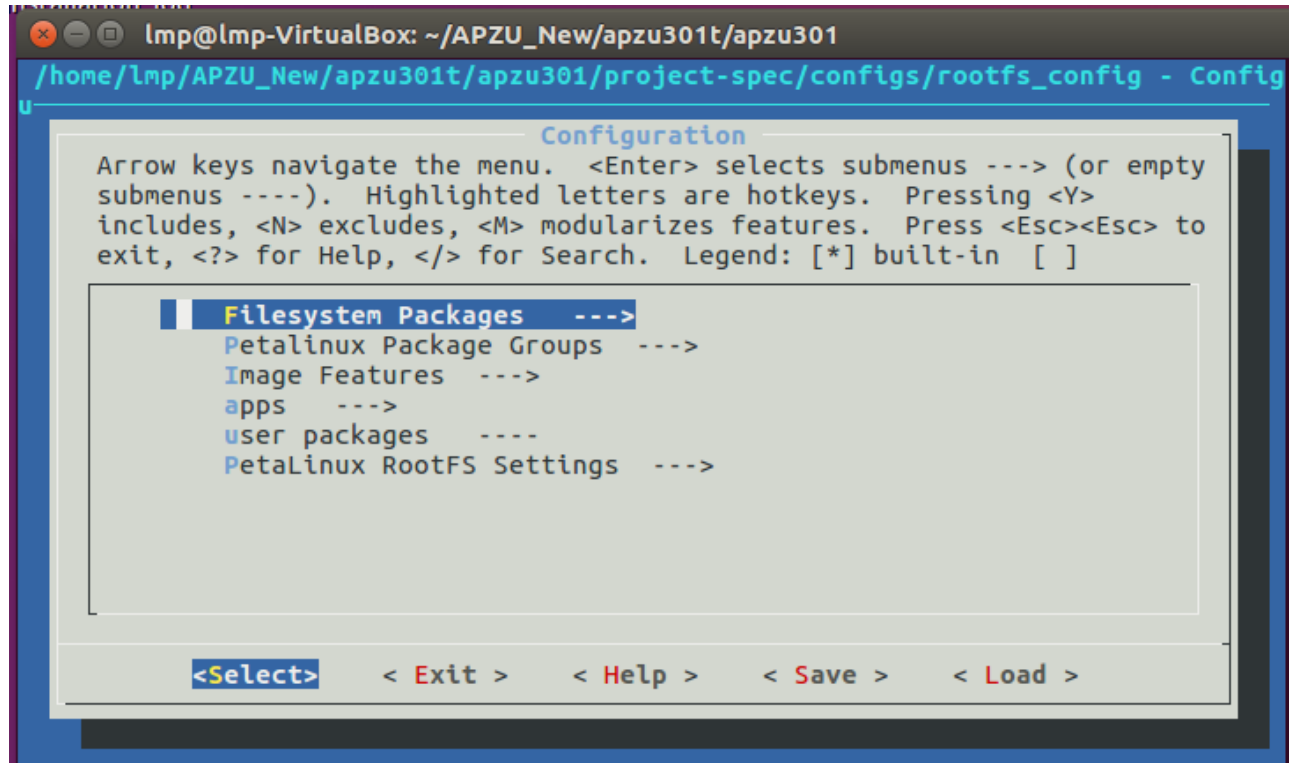
In the terminal window verify that you are in the following directory location:

```
~/APZU_New/apzu301t/apzu301
```

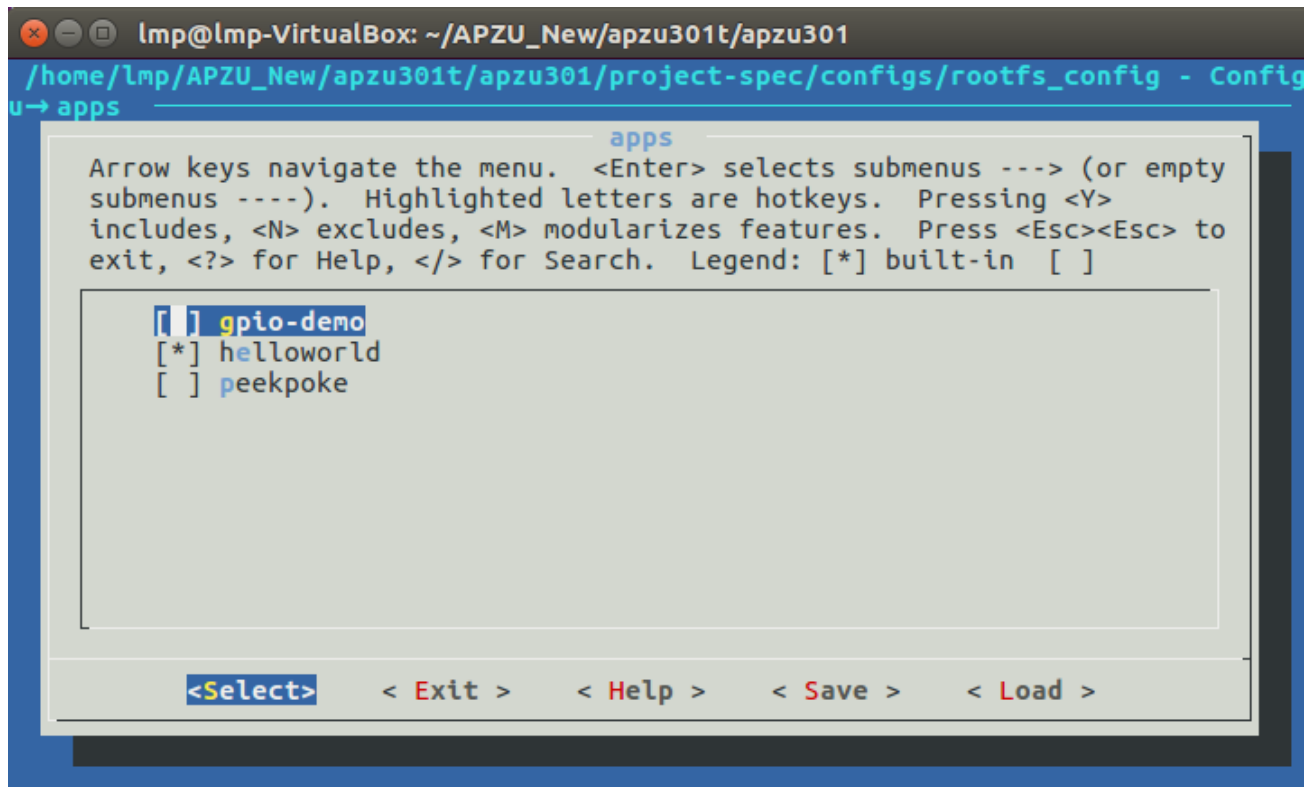
Run the following petalinux-config commands in order to compile the changes of the helloworld.c file into the boot files.

**\$ petalinux-config -c rootfs**

A configuration screen similar to the following will appear.



Navigate to apps and verify that the helloworld is included.



Hit Esc 4 times to exit out. Next, execute:

**\$ petalinux-config -c kernel**

This may take a while to run, depending on your system.

If desired, you can modify the kernel configurations at this time. Otherwise, select Esc twice to exit out of the Kernel Configuration.

A screen similar to what is given below should appear, letting you know that the kernel was successfully configured.

```
NOTE: Setscene tasks completed
NOTE: Tasks Summary: Attempted 396 tasks of which 388 didn't need to be rerun and all succeeded.
INFO: Updating config fragment /home/lmp/APZU_New/apzu301t/apzu301/components/yocto/workspace/sources/linux-xlnx/oe-local-files/devtool-fragment.cfg
[INFO] successfully configured kernel
lmp@lmp-VirtualBox:~/APZU_New/apzu301t/apzu301$
```

Continue with petalinux-build as given in the next section.

### 3.4 petalinux-build

This step generates a device tree (DTB) file, a first stage boot loader, Zynq UltraScale+ Arm Trusted Firmware (ATF), U-Boot, Linux kernel, and a root file system image. This step can take up to 45 minutes to complete.

**\$ petalinux-build**

Continue with petalinux-package as given in the next section.

### 3.5 petalinux-package

This step generates a boot image. The boot image can be loaded into the SD card. When the APZU board is powered up it will boot from the boot image. The boot image contains the first stage boot loader image, FPGA bitstream, PMU firmware, ATF, and U-Boot.

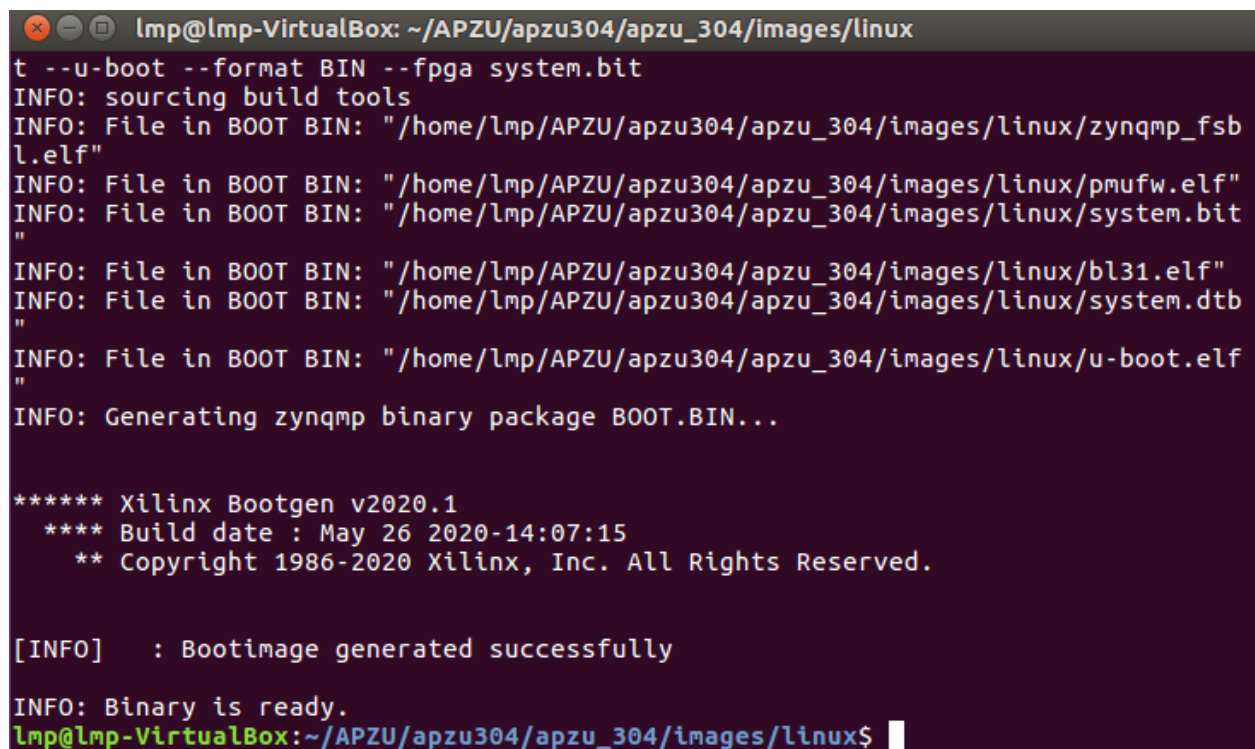
Change directory to location of images with following command.

**\$ cd images/linux**

Execute the following command to generate the boot image in .BIN format.

**\$ petalinux-package --boot --u-boot --format BIN --fpga system.bit**

If the image already exists add the `--force` to allow overwrite of previous version.

**\$ petalinux-package --boot --u-boot --format BIN --fpga system.bit --force**

```
lmp@lmp-VirtualBox: ~/APZU/apzu304/apzu_304/images/linux
$ petalinux-package --boot --u-boot --format BIN --fpga system.bit
INFO: sourcing build tools
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/zynqmp_fsb
l.elf"
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/pmufw.elf"
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/system.bit
"
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/bl31.elf"
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/system.dtb
"
INFO: File in BOOT BIN: "/home/lmp/APZU/apzu304/apzu_304/images/linux/u-boot.elf
"
INFO: Generating zynqmp binary package BOOT.BIN...

***** Xilinx Bootgen v2020.1
**** Build date : May 26 2020-14:07:15
** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

[INFO] : Bootimage generated successfully

INFO: Binary is ready.
lmp@lmp-VirtualBox:~/APZU/apzu304/apzu_304/images/linux$
```

### 3.6 Adding Files to SD Card

See section 3.8 for the steps needed to FAT partition an SD card.

In the VirtualBox pull downs for the Virtual Machine, select **Devices -> USB -> SanDisk MobileMate Micro [9407]**

Copy the following files from `/pre-built/linux/images/` into the root directory of the first FAT partition of your SD card:

- BOOT.BIN
- image.ub
- boot.scr

Note **Imp** must be replaced with your system directory

If your current SD card currently has data in the root partition, run the following command in your terminal.

```
$ sudo rm -r /media/Imp/root/*
```

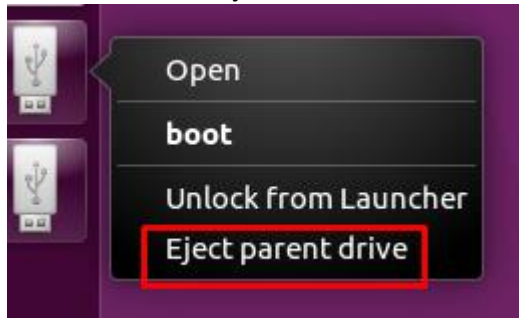
The following updates the root partition files on the SD card. Execute the following from the `./images/linux` directory.

```
$ sudo tar -C /media/Imp/root -xvf rootfs.tar.gz
```

Run the sync command to flush the Virtual Machine's cache to the SD card.

```
$ sudo sync
```

Close folders and eject the SD card.



### 3.7 Boot Process

Open a console on your workstation and then start your preferred serial communication program (e.g., PuTTY, Tera Term, kermit, etc.) with the baud rate set to 115200 on that console.

Set the boot mode of the board to SD boot.

Plug the SD card into the boards SD socket.

Power on the board.

At the login prompt, use the login **root** and password **root** to log into the Linux system.

```
[ 8.286434] random: crng init done
Public key portion is:
ssh-rsa AAAAB3NzaClyc2EAAAADAQABAAQACidXcsNAh9rlKS6eHPOUN2PljhPdjiDELWR4VUWpd+
FZVTOpb67db5QX4txlac+GkhwNuCAG4Ss6xhByoZbDqJdTwA3+F2vDXRVtKZJ6eKDQRQcBG6MYiHT7Bf
p7h9oKDPQrhmICCAJwEI00EASqkZlC5fk7bhgFeGGAJHrxMLae7W6cpbzoGSRQ5lgBN28/JC9QK3s4k
zbv7CfHHDlHF2XfBWJG77RMtIfmq6aYd0ltsFetlVeIVtOu/nhAft+AOlshu4nIHxIytrNXqSZ4WpBoX
faCv2uwSPed9BNixmzikJT/j8eLanSCWGkfBQSVXG5R5bzuVmt5kf7p0NQsS1 root@apzu_303
Fingerprint: sha1!! b9:51:2a:01:cf:14:3d:86:4a:99:5d:3a:90:b6:f8:d9:03:6d:23:26
dropbear.
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

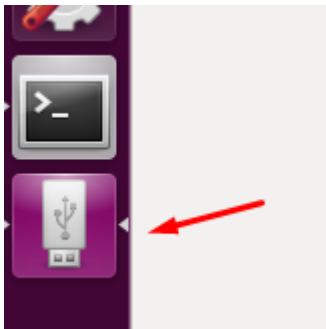
PetaLinux 2020.1 apzu_303 /dev/ttyPS0

apzu_303 login: root
Password:
root@apzu_303:~#
```

At this point the First Stage Boot Loader has initialized the processor. U-Boot has passed the device tree to boot the Linux system. The Linux Kernel initializes system hardware and mount root file system based on device tree. Custom startup scripts can be used to launch application specific processes.

### 3.8 Create FAT Partition SD Card

See Xilinx ug1144 for additional details. Follow these steps to prepare an SD card for you boot files.



- 1) Insert the SD card into your USB SD card reader of your computer.
- 2) Ignore any file explorer pop ups and messages that appears related to the card being inserted.
- 3) In the VirtualBox pull downs for the Virtual Machine, select Devices -> USB -> -> SanDisk MobileMate Micro [9407].
- 4) The contents of the card will pop up in File Explorer Window as well as register to the side bar of your Ubuntu Virtual Machine.
- 5) Close any File Explorer pop ups.
- 6) Open a terminal window. => Press Ctrl + Alt + t simultaneously to open a new terminal.
- 7) In terminal window type
 

```
$ ls /dev/sd*
```
- 8) As you can see there are only two devices shown sda and sdb



```
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$ ls /dev/sd*  
/dev/sda /dev/sda1 /dev/sda2 /dev/sda5 /dev/sdb /dev/sdb1  
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$
```

- 9) If your Virtual Machine process configured a SINGLE hard drive is should be /sda; thus, we can assume that /sdb is the SDCARD.  
If you are unsure or your configuration has more partitions, remove the SDCARD and re-run the ls command to determine which is the SDCARD.

- 10) In terminal window type

```
$ sudo fdisk /dev/sdb
```

- 11) Type **p** to see a list of existing partitions. Your card might show something different then that shown below.

```
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$ sudo fdisk /dev/sdb  
[sudo] password for lmp:  
  
Welcome to fdisk (util-linux 2.27.1).  
Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.  
  
Command (m for help): p  
Disk /dev/sdb: 14.9 GiB, 16022241280 bytes, 31293440 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x00000000  
  
Device      Boot Start      End  Sectors  Size Id Type  
/dev/sdb1             8192 31293439 31285248 14.9G  c W95 FAT32 (LBA)  
  
Command (m for help):
```

- 12) If you have partitions, delete them.  
press **d** and select each as seen below.

```
Command (m for help): d  
Selected partition 1  
Partition 1 has been deleted.  
  
Command (m for help):
```

- 13) Make selection **n**, then type **p** for primary, **1** for the first partition, **2048** (or just enter), finally **+1G**



```

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-31293439, default 2048): 2048
Last sector, +sectors or +size[K,M,G,T,P] (2048-31293439, default 31293439): +1G

Created a new partition 1 of type 'Linux' and of size 1 GiB.

Command (m for help): █

```

- 14) As this will become our BOOT partition, we will need to mark it as such by typing **a**
- 15) Create the second partition that will be used by the ROOTFS. Make selection **n**, then type **p** for primary, **2** then press **ENTER** twice. This selects the rest of the card's memory for the second partition.
- 16) Type **p** to check the new partitions. It should look similar to the following.

```

Command (m for help): p
Disk /dev/sdb: 14.9 GiB, 16022241280 bytes, 31293440 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device      Boot  Start      End  Sectors  Size Id Type
/dev/sdb1   *      2048   2099199   2097152    1G 83 Linux
/dev/sdb2                2099200 31293439 29194240 13.9G 83 Linux

Command (m for help): █

```

- 17) Type **w** to quit. This will write the new table configuration.
- 18) If you get the following error, continue to step 19.

```

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8).

```

- 19) **\$ umount /dev/sdb1**  
**\$ umount /dev/sdb2**  
**\$ sudo fdisk /dev/sdb**  
Type **p** (see list of existing partitions)  
Type **w** (quit fdisk)

20) Format the boot partition as FAT32 and ROOTFS partition as EXT4

**\$ sudo mkfs.vfat -F 32 -n boot /dev/sdb1**

**\$ sudo mkfs.ext4 -L root /dev/sdb2**

```
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$ sudo mkfs.vfat -F 32 -n boot /dev/sdb1
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$ sudo mkfs.ext4 -L root /dev/sdb2
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 3649280 4k blocks and 913920 inodes
Filesystem UUID: 022e8d77-fec7-4d36-b137-55c250fd26ce
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

lmp@lmp-VirtualBox:~/APZU_New/apzu304/apzu_304/images/linux$
```

### 3.9 Create the PetaLinux helloworld application

To add the helloworld application the PetaLinux create command is used.

The application is located: <project-root>/project-spec/meta-user/recipes-apps/myapp

Change directories to the location of the build, components, image, and project-spec folders (as seen below).

**\$ cd apzu\_303**

```
lmp@lmp-VirtualBox:~/APZU_New/apzu303$ dir
apzu_303  apzu303_BSP.bsp  APZU_top.xsa
lmp@lmp-VirtualBox:~/APZU_New/apzu303$ cd apzu_303
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$ dir
build  components  config.project  images  project-spec
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$
```

To create a new software application, that will also be enabled in the root filesystem configuration, so that it will become part of the installed PetaLinux boot image run the following.

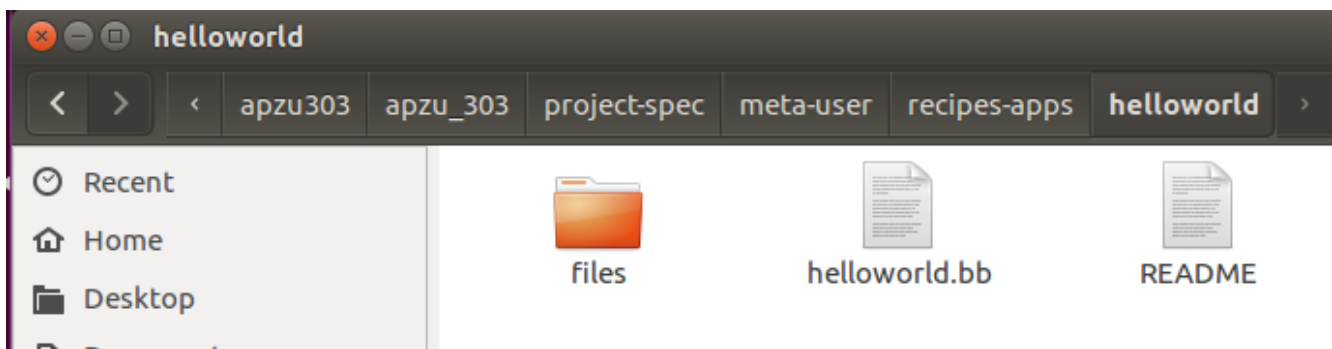
**\$ petalinux-create -t apps --template c --name helloworld --enable**

(Note: If you decide to change the name of the application, make sure to use lower case letters.)

```
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$ petalinux-create -t apps --templ
ate c --name helloworld --enable
INFO: Create apps: helloworld
INFO: New apps successfully created in /home/lmp/APZU_New/apzu303/apzu_303/proje
ct-spec/meta-user/recipes-apps/helloworld
INFO: Enabling created component...
INFO: sourcing build environment
INFO: silentconfig rootfs
INFO: helloworld has been enabled
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$
```

The new custom application can be found with its created files in the following directory:

apzu303/apzu\_303/project-spec/meta-user/recipes-apps/helloworld



The created application already contains a template-generated c-file. If desired, the file can be opened and modified. This program has been modified to include the PCIe bus ingress setup. It also includes code to write the first 16 locations to the PL BRAM. Additionally, it includes code to activate the two LED present on the board. Finally, it ends by printing "Hello World! :-))".

Change directories to

apzu303/apzu\_303/project-spec/meta-user/recipes-apps/helloworld/files

Open the helloworld.c file using gedit. The completed helloworld.c code is given in the petalinux directory of the EDK design.

To compile and helloworld application and build into the PetaLinux boot system you need to perform the following.

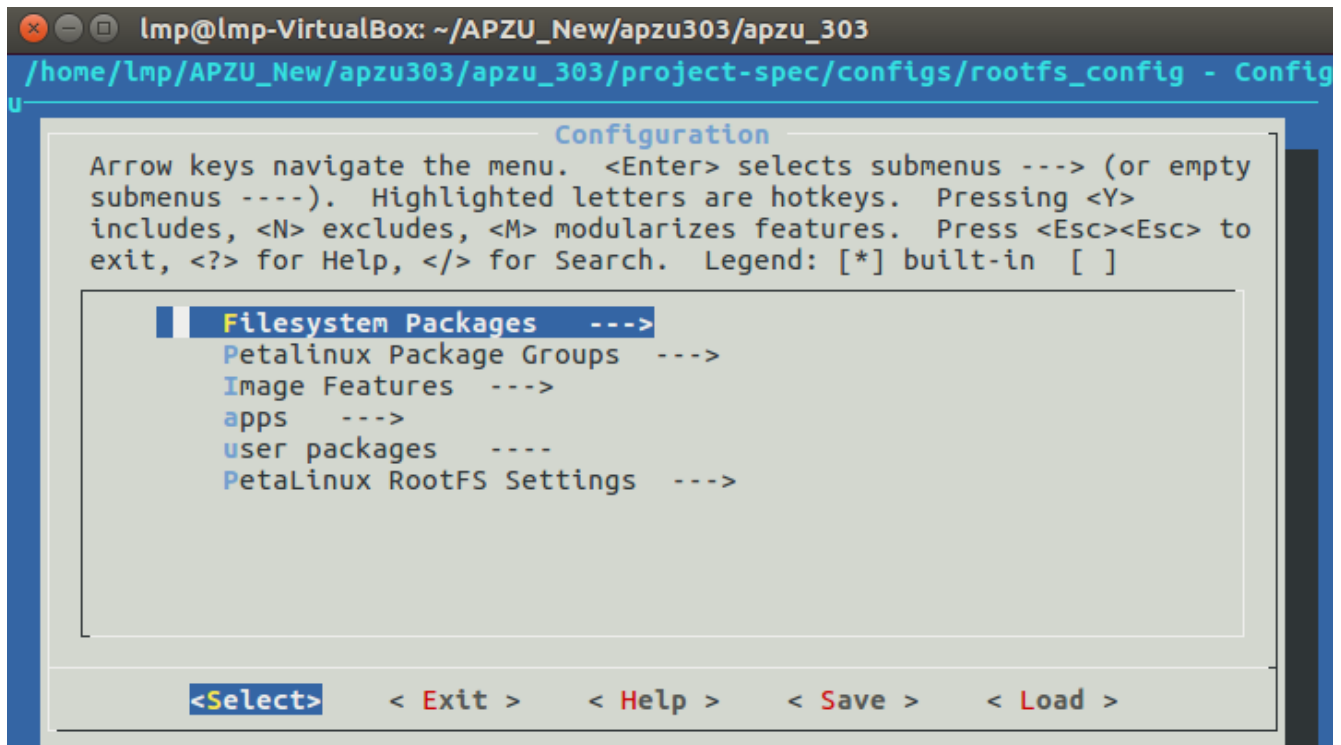
Type the following into the terminal window.

```
$ cd ~/APZU_New/apzu303/apzu_303
```

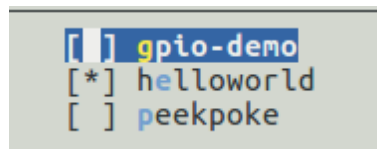
Then type the following command.

```
$ petalinux-config -c rootfs
```

The following configuration screen should appear.



Navigate to apps and verify that the helloworld is included and selected.



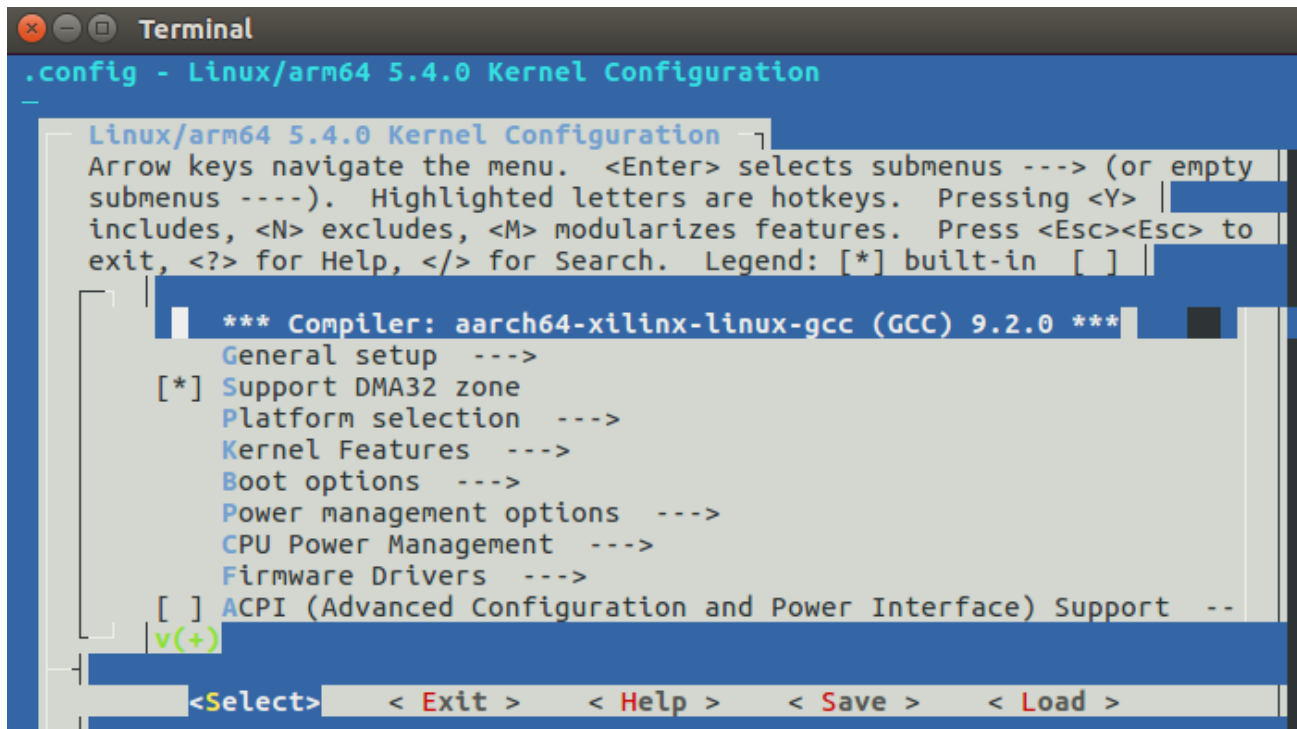
Press the Esc key four times to exit out.

Next, type the following into the terminal window.

**\$ petalinux-config -c kernel**

This may take a while to run, depending on your system.

If desired, modify the kernel configurations. Hit Esc twice to exit out of the Kernel Configuration.



[INFO] **successfully configured kernel** should appear, letting you know that the kernel was successfully configured.

Now, type the following in the terminal window.

**\$ petalinux-build**

This could take more than ten minutes depending on your system. If a warning (such as is shown below) occurs, it can be ignored. This is where compile errors can show up in your helloworld.c program. The errors cannot be ignored and must be corrected.

```

lmp@lmp-VirtualBox: ~/APZU_New/apzu303/apzu_303
[INFO] successfully configured kernel
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$ petalinux-build
INFO: sourcing build tools
[INFO] building project
[INFO] sourcing build environment
[INFO] generating user layers
[INFO] generating workspace directory
INFO: bitbake petalinux-image-minimal
Loading cache: 100% |#####| Time: 0:00:01
Loaded 4230 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:00:05
Parsing of 2962 .bb files complete (2960 cached, 2 parsed). 4231 targets, 169 sk
ipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
WARNING: /home/lmp/APZU_New/apzu303/apzu_303/components/yocto/layers/meta-xilinx
/meta-xilinx-bsp/recipes-kernel/linux/linux-xlnx_2020.1.bb:do_compile is tainted
from a forced run
Initialising tasks: 100% |#####| Time: 0:00:04
Checking sstate mirror object availability: 100% |#####| Time: 0:00:06
Sstate summary: Wanted 158 Found 18 Missed 140 Current 856 (11% match, 86% compl
ete)
NOTE: Executing Tasks
NOTE: Setscene tasks completed
NOTE: linux-xlnx: compiling from external source tree /home/lmp/APZU_New/apzu303
/apzu_303/components/yocto/workspace/sources/linux-xlnx
NOTE: Tasks Summary: Attempted 3626 tasks of which 3157 didn't need to be rerun
and all succeeded.

Summary: There was 1 WARNING message shown.
INFO: Successfully copied built images to tftp dir: /tftpboot
[INFO] successfully built project
lmp@lmp-VirtualBox:~/APZU_New/apzu303/apzu_303$

```

Complete steps given in section 3.5 to run petalinux-package.

Complete steps given in section 3.6 to add boot files to SD card.

Complete steps given in section 3.7 to test boot using the SD card.

### 3.10 Running the helloworld application

After booting to PetaLinux and signing in with user **root** and password **root**,. Type the cd to /usr/bin and ls commands to the terminal window as shown below.

```

apzu_303 login: root
Password:
root@apzu_303:~#

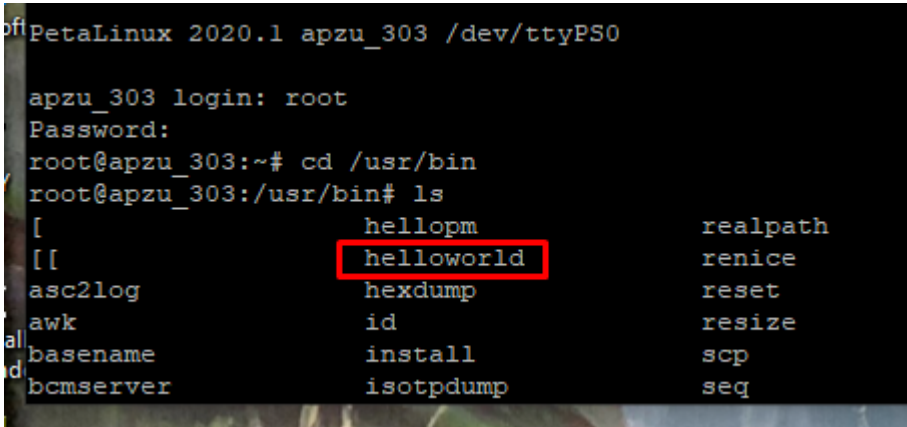
```

```
$ cd /usr/bin
```

```
$ ls
```



Identify the helloworld application as seen below.



```
PetaLinux 2020.1 apzu_303 /dev/ttyPS0

apzu_303 login: root
Password:
root@apzu_303:~# cd /usr/bin
root@apzu_303:/usr/bin# ls
[          hellopm          realpath
[[          helloworld      renice
asc2log     hexdump      reset
awk         id          resize
all         basename    scp
d           bcmserver   isotpdump   seq
```

Below is a screen capture of the Windows OS APZU demonstration program.

The windows OS example program will print the following to the screen.

**Has the PetaLinux ingress program already been run for this board since system start [Y/N]?:**

Answer n and select return.

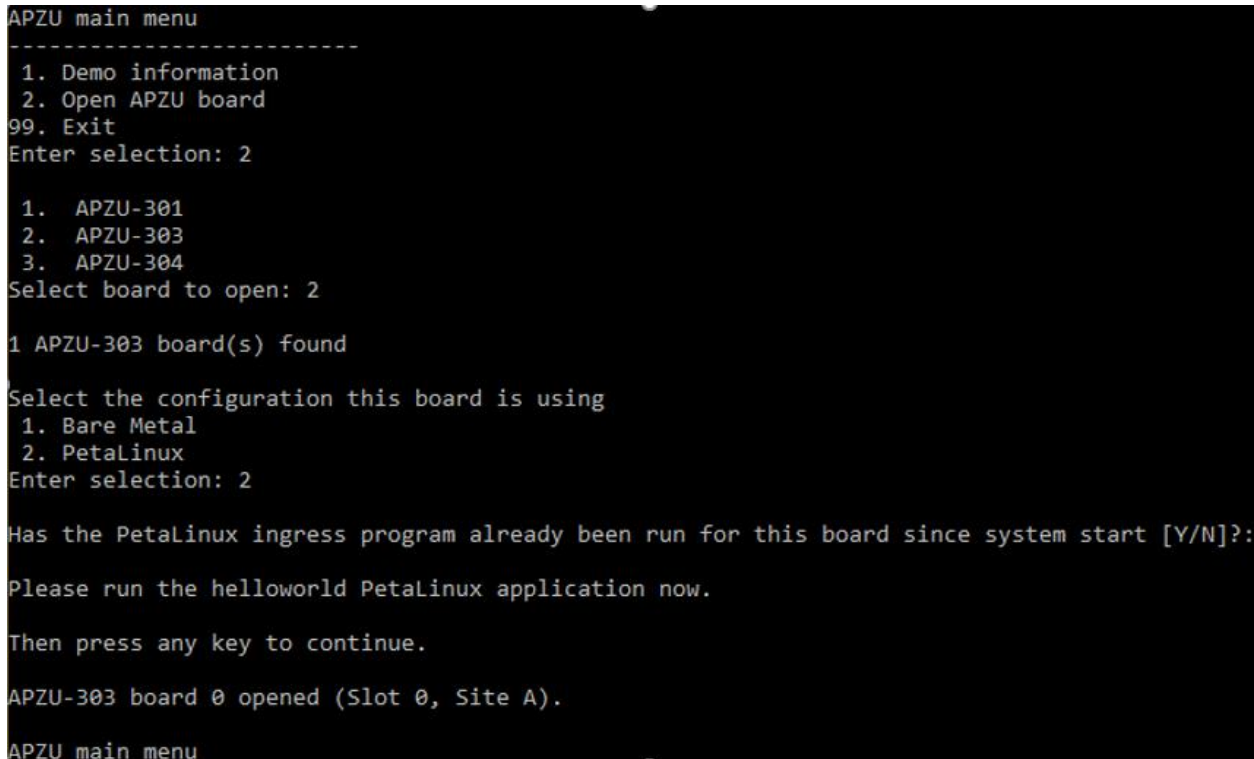
You will be directed to run the helloworld application at this time.

At the petalinux putty terminal enter the following:

**\$ cd /usr/bin**

**\$ helloworld**

Then press any key to continue back at the windows OS example design.



```
APZU main menu
-----
1. Demo information
2. Open APZU board
99. Exit
Enter selection: 2

1. APZU-301
2. APZU-303
3. APZU-304
Select board to open: 2

1 APZU-303 board(s) found

Select the configuration this board is using
1. Bare Metal
2. PetaLinux
Enter selection: 2

Has the PetaLinux ingress program already been run for this board since system start [Y/N]? :
Please run the helloworld PetaLinux application now.

Then press any key to continue.

APZU-303 board 0 opened (Slot 0, Site A).

APZU main menu
```

The helloworld program will run and display the output to the screen.

```

root@apzu_304:/usr/bin# helloworld
ptr_pcielink = *0xB8837000*
Check if PCIe Link up
00000003
PCIe Link up...
Hello World! :-))
root@apzu_304:/usr/bin#

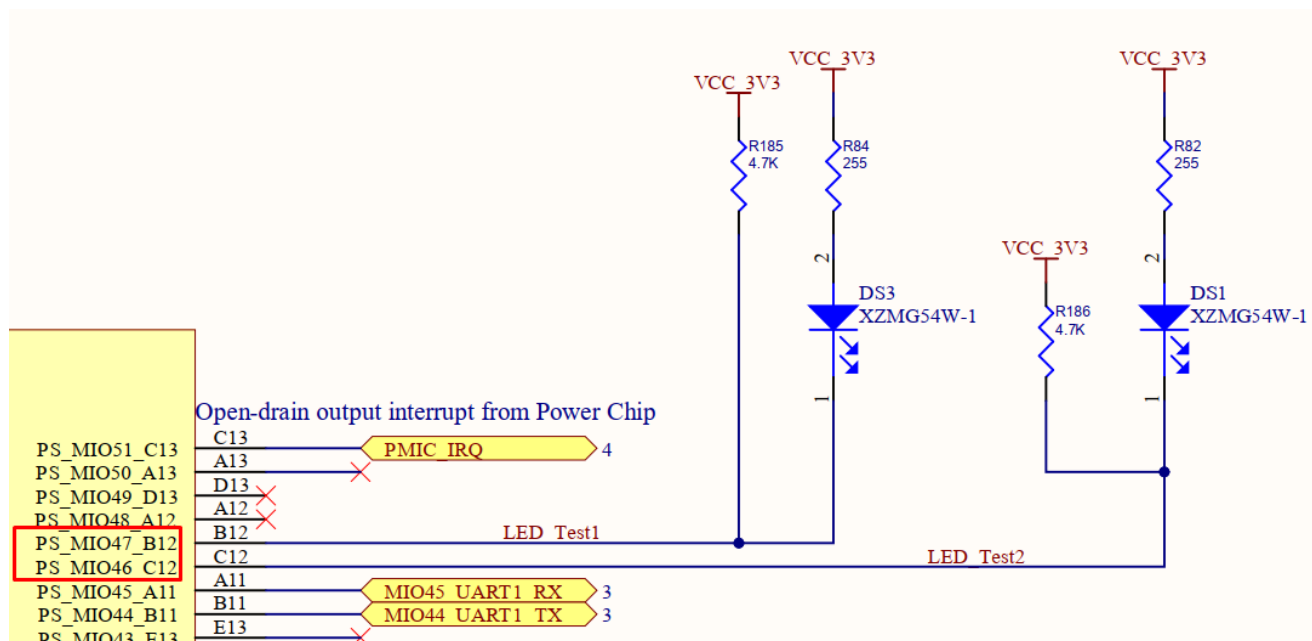
```

At this point you can continue with running the windows OS demonstration program. For example, see section 3.12 which illustrates read of the BRAM.

### 3.11 Linux GPIO

Demonstration of access to APZU LEDs is demonstrated using a few Linux base methods.

Notice in schematic image below that the LEDs are controlled with Zynq MIO46 and MIO47 pins.



The pins are access at base\_gpio + offset, or 338 + 46 = 384 (LED DS1), or 338 + 47 = 385 (LED DS3).

A value of "1" will turn LED off and a value of "0" will turn LED on.

Test LED control using Linux echo command as shown below.

```

echo 384 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio384/direction
echo 0 > /sys/class/gpio/gpio384/value
echo 1 > /sys/class/gpio/gpio384/value
echo 384 > /sys/class/gpio/unexport echo 384 >

```



```
echo 385 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio385/direction
echo 0 > /sys/class/gpio/gpio385/value
echo 1 > /sys/class/gpio/gpio385/value
echo 385 > /sys/class/gpio/unexport
```

Another method is direct access to GPIO device registers with /dev/mem

The ZYNQMP GPIO controller is at address 0xFF0A0000 in the device tree.

LED pin 46 is controlled via bit 20 and LED pin 47 is controlled via bit 21.

**\$ cd /usr/sbin**

Enable ports

**\$ devmem 0xFF0A0248 w 0x300000**

Set Direction to output

**\$ devmem 0xFF0A0244 w 0x300000**

Write date 0 to turn LED on.

**\$ devmem 0xFF0A0044 w 0x0**

Write date 1 to turn LED off.

**\$ devmem 0xFF0A0044 w 0x300000**

Also the following code is included in the helloworld.c file provided by Acromag. This code also activates the LEDs.

```
// activate LED DS3 with at 385 and LED DS1 at 384
fd = open("/sys/class/gpio/export", O_WRONLY);
write(fd, "384", 4);
close(fd);

// write to the direction register so all the GPIOs are on output to drive LEDs
fd = open("/sys/class/gpio/gpio384/direction", O_WRONLY);
write(fd, "out", 4);
close(fd);

fd = open("/sys/class/gpio/gpio384/value", O_WRONLY);
write(fd, "0", 2);
close(fd);

fd = open("/sys/class/gpio/export", O_WRONLY);
write(fd, "385", 4);
close(fd);

fd = open("/sys/class/gpio/gpio385/direction", O_WRONLY);
write(fd, "out", 4);
close(fd);

fd = open("/sys/class/gpio/gpio385/value", O_WRONLY);
write(fd, "0", 2);
close(fd);
```

### 3.12 Access to PL Digital I/O Ports

The digital I/O ports present in the PL programmable logic of the FPGA can be accessed from petalinux user space using the devmen and mmap functions.

#### BRAM and mmap

The helloworld application illustrates use of the mmap function to access the BRAM. The first 16 bytes are written and then verified via read of the same location using the PCIe bus.

```
ptr_AXI =
    mmap(NULL, pgsz, PROT_READ | PROT_WRITE | PROT_EXEC, MAP_SHARED, fdm, (off_t)ba_AXI );
if (ptr_AXI == MAP_FAILED) {
    printf("mmap: %s (%d)\n", strerror(errno), errno);
    close(fdm);
    return -1;
}

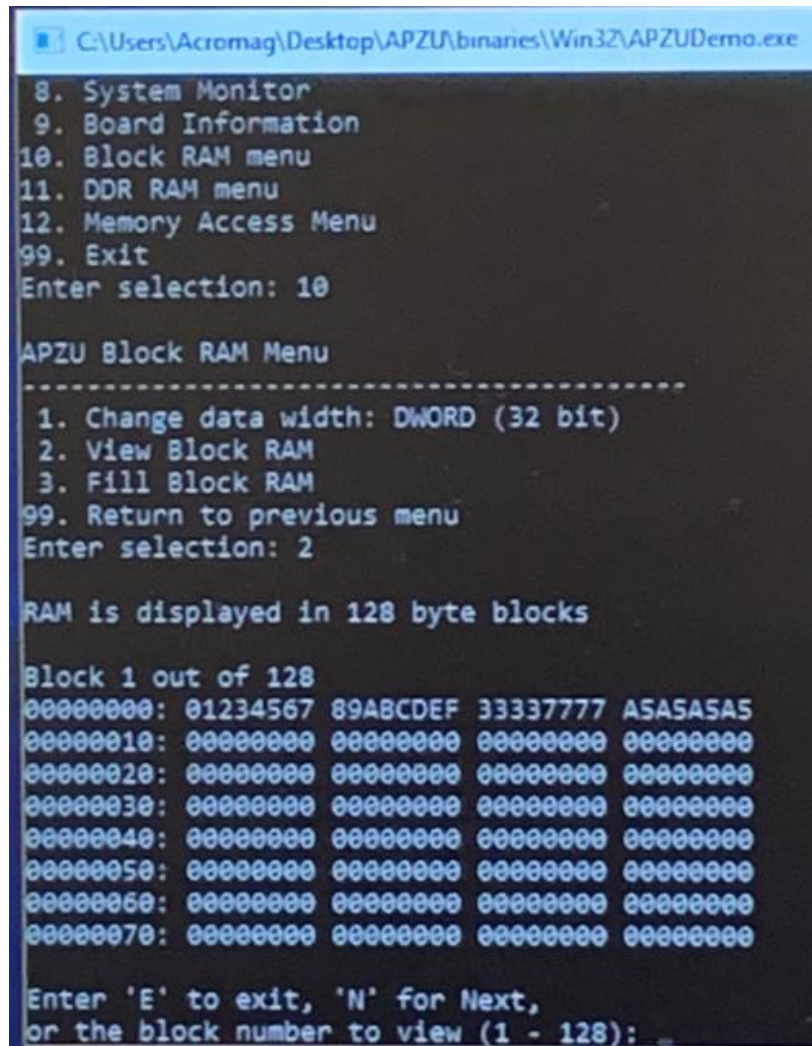
plmp = (uint32_t *) (ptr_AXI + offset_AXI + 0x4000);
*plmp = 0x1234567;

plmp = (uint32_t *) (ptr_AXI + offset_AXI + 0x4004);
*plmp = 0x89ABCDEF;

plmp = (uint32_t *) (ptr_AXI + offset_AXI + 0x4008);
*plmp = 0x33337777;

plmp = (uint32_t *) (ptr_AXI + offset_AXI + 0x400C);
*plmp = 0xA5A5A5A5;
```

The 16 bytes written to BRAM by the helloworld application can be verified using the PCIe interface to the BRAM. As seen below the view of the BRAM verifies that these values have been correctly written. To view the BRAM select option 10 followed by option 2.



```

C:\Users\Acromag\Desktop\APZU\binaries\Win32\APZUDemo.exe
8. System Monitor
9. Board Information
10. Block RAM menu
11. DDR RAM menu
12. Memory Access Menu
99. Exit
Enter selection: 10

APZU Block RAM Menu
-----
1. Change data width: DWORD (32 bit)
2. View Block RAM
3. Fill Block RAM
99. Return to previous menu
Enter selection: 2

RAM is displayed in 128 byte blocks

Block 1 out of 128
00000000: 01234567 89ABCDEF 33337777 ASASASAS
00000010: 00000000 00000000 00000000 00000000
00000020: 00000000 00000000 00000000 00000000
00000030: 00000000 00000000 00000000 00000000
00000040: 00000000 00000000 00000000 00000000
00000050: 00000000 00000000 00000000 00000000
00000060: 00000000 00000000 00000000 00000000
00000070: 00000000 00000000 00000000 00000000

Enter 'E' to exit, 'N' for Next,
or the block number to view (1 - 128):

```

### BRAM and devmem

The following illustrates use of devmem when reading the contents of the BRAM in the PL of the FPGA.

The BRAM is accessed at AXI address 0xA0004000 to 0xA0007FFF. This test also utilized the PCIe interface to the APZU to load the first seven locations of BRAM with 0, 1, 2, 3, 4, 5, and 6 as seen below.

```
C:\Users\Acromag\Desktop\APZU\binaries\Win32\APZUDemo.exe
Enter selection: 1
1. Fill all Block RAM
2. Specify start of block and value count
Enter selection: 2
Enter offset from beginning of buffer (0 - 3FFC): 0
Enter number of values to modify (0 - 4096): 7
APZU Block RAM Menu
-----
1. Change data width: DWORD (32 bit)
2. View Block RAM
3. Fill Block RAM
99. Return to previous menu
Enter selection: 2
RAM is displayed in 128 byte blocks
Block 1 out of 128
00000000: 00000000 00000001 00000002 00000003
00000010: 00000004 00000005 00000006 00000000
00000020: 00000000 00000000 00000000 00000000
00000030: 00000000 00000000 00000000 00000000
00000040: 00000000 00000000 00000000 00000000
00000050: 00000000 00000000 00000000 00000000
00000060: 00000000 00000000 00000000 00000000
00000070: 00000000 00000000 00000000 00000000
Enter 'E' to exit, 'N' for Next,
or the block number to view (1 - 128):
```

The devmem command is then used to read these same memory locations, from the petalinux command prompt, as seen below.

```
root@apzu_303:/usr/bin# devmem 0xA0004000
0x00000000
root@apzu_303:/usr/bin# devmem 0xA0004004
0x00000001
root@apzu_303:/usr/bin# devmem 0xA0004008
0x00000002
root@apzu_303:/usr/bin# devmem 0xA000400c
0x00000003
root@apzu_303:/usr/bin# devmem 0xA0004010
0x00000004
root@apzu_303:/usr/bin# devmem 0xA0004014
0x00000005
root@apzu_303:/usr/bin# devmem 0xA0004018
0x00000006
root@apzu_303:/usr/bin# devmem 0xA000401c
0x00000000
root@apzu_303:/usr/bin#
```

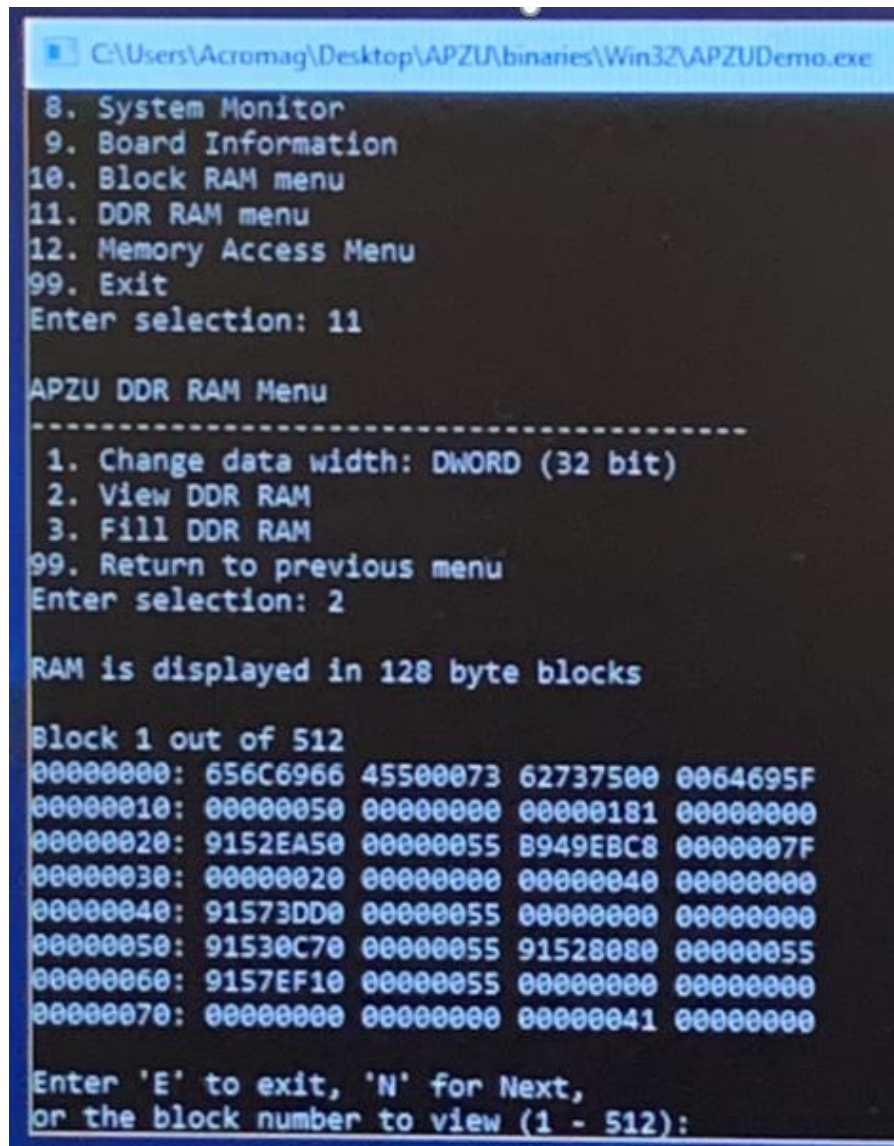
### DDR Memory

The following illustrates use of devmem when reading the contents of the DDR memory.

The DDR memory is accessed at AXI address 0x01000000 to 0x0100FFFF. A 64K byte memory space of the DDR is accessible starting at address 0x01000000. This test also utilized the PCIe interface to the APZU to verify the contents of the DDR memory.

The contents of the DDR memory are read using the PCIe interface to the APZU as seen below.





```

C:\Users\Acromag\Desktop\APZU\binaries\Win32\APZUDemo.exe
8. System Monitor
9. Board Information
10. Block RAM menu
11. DDR RAM menu
12. Memory Access Menu
99. Exit
Enter selection: 11

APZU DDR RAM Menu
-----
1. Change data width: DWORD (32 bit)
2. View DDR RAM
3. Fill DDR RAM
99. Return to previous menu
Enter selection: 2

RAM is displayed in 128 byte blocks

Block 1 out of 512
00000000: 656C6966 45500073 62737500 0064695F
00000010: 00000050 00000000 00000181 00000000
00000020: 9152EA50 00000055 B949EBC8 0000007F
00000030: 00000020 00000000 00000040 00000000
00000040: 91573DD0 00000055 00000000 00000000
00000050: 91530C70 00000055 91528080 00000055
00000060: 9157EF10 00000055 00000000 00000000
00000070: 00000000 00000000 00000041 00000000

Enter 'E' to exit, 'N' for Next,
or the block number to view (1 - 512):

```

The devmem command is then used to read the first four memory locations as seen below.

```

root@apzu_303:/usr/bin# devmem 0x01000000
0x656C6966
root@apzu_303:/usr/bin# devmem 0x01000004
0x45500073
root@apzu_303:/usr/bin# devmem 0x01000008
0x62737500
root@apzu_303:/usr/bin# devmem 0x0100000c
0x0064695F

```

### Digital I/O

The digital I/O ports can be access using devmem as demonstrated above at the addresses given below.

See the APZU user manual for the usable bits of these registers.

AXI Base Address	Bit(s)	Description
<b>0xA000 0000</b>	31:0	Global Interrupt Register
<b>0xA000 0004</b>	31:0	Location in System Register
<b>0xA000 0008</b>	31:0	Digital Input/Output Register
<b>0xA000 000C</b>	31:0	Digital Direction Control Register
<b>0xA000 0010</b>	31:0	Interrupt Enable Register
<b>0xA000 0014</b>	31:0	Interrupt Type Register
<b>0xA000 0018</b>	31:0	Interrupt Polarity Register
<b>0xA000 001C</b>	31:0	Interrupt Status Register
<b>0xA000 0020</b>	31:0	NOT USED
<b>0xA000 0024</b>	31:0	NOT USED
<b>0xA000 0028</b>	31:0	NOT USED
<b>0xA000 002C</b>	31:0	RS485 Data Register <sup>1</sup>
<b>0xA000 0030</b>	31:0	RS485 Direction Control Register <sup>1</sup>
<b>0xA000 0034→ 0xA000 01FF</b>	31:0	NOT USED
<b>0xA000 0200</b>	31:0	Firmware Revision Register
<b>0xA000 0204→ 0xA000 1FFF</b>	31:0	NOT USED <sup>2</sup>

**Note 1:** RS485 is only available on the APZU-303 model.

### 3.13 Test of Ethernet

To demonstration functional operation of the Ethernet port, the ping test can be used as illustrated below.

Running a ping test will report the time it takes for a small data set to be transmitted from the APZU to a server on the Internet and back to your device again. Ping performs a to two-way latency round-trip delay test.

Below is image of results testing ping from APZU to the Acromag website.

```
root@apzu_304:~# ping www.acromag.com -c5
PING www.acromag.com (104.24.120.215): 56 data bytes
64 bytes from 104.24.120.215: seq=0 ttl=59 time=3.251 ms
64 bytes from 104.24.120.215: seq=1 ttl=59 time=2.718 ms
64 bytes from 104.24.120.215: seq=2 ttl=59 time=2.711 ms
64 bytes from 104.24.120.215: seq=3 ttl=59 time=4.799 ms
64 bytes from 104.24.120.215: seq=4 ttl=59 time=2.710 ms

--- www.acromag.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 2.710/3.237/4.799 ms
root@apzu_304:~#
```

### 3.14 Test of USB

Demonstration functional operation of the USB port is illustrated below with use of cat command. The cat (short for “concatenate”) command is one of the most frequently used command in Linux like operating systems. The command cat allows us to create single or multiple files, view contents of file, concatenate files and redirect output in terminal or files.

Run lsusb without a USB thumb drive in the breakout panels USB port.

#### lsusb

Place USB thumb drive into the breakout panels USB port and then run lsusb again. Notice a new device is detected as seen below.

```
root@apzu_303:~# lsusb
Bus 001 Device 001: ID 1d6b:0002
Bus 002 Device 001: ID 1d6b:0003
root@apzu_303:~# [ 27.765354] usb 1-1: new high-speed USB device number 2 using xhci-hcd
[ 27.920694] usb 1-1: New USB device found, idVendor=05e3, idProduct=0751, bcdDevice=14.04
[ 27.928868] usb 1-1: New USB device strings: Mfr=3, Product=4, SerialNumber=0
[ 27.936008] usb 1-1: Product: USB Storage
[ 27.940011] usb 1-1: Manufacturer: USB Storage
[ 27.946115] usb-storage 1-1:1.0: USB Mass Storage device detected
[ 27.952532] scsi host0: usb-storage 1-1:1.0
[ 28.982660] scsi 0:0:0:0: Direct-Access Generic STORAGE DEVICE 1404 PQ: 0 ANSI: 6
[ 29.232373] sd 0:0:0:0: [sda] 31116288 512-byte logical blocks: (15.9 GB/14.8 GiB)
[ 29.241089] sd 0:0:0:0: [sda] Write Protect is off
[ 29.246997] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 29.275495] sda: sdal
[ 29.285038] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 29.526000] cramfs: Unknown parameter 'umask'
[ 29.535761] FAT-fs (sdal): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
root@apzu_303:~# lsusb
Bus 001 Device 001: ID 1d6b:0002
Bus 001 Device 002: ID 05e3:0751
Bus 002 Device 001: ID 1d6b:0003
root@apzu_303:~#
```



Run the mount command as seen below and notice the media/sda1 device.

#### mount

```
root@apzu_303:~# mount
none on / type rootfs (rw)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
devtmpfs on /dev type devtmpfs (rw,relatime,size=884012k,nr_inodes=221003,mode=755)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /var/volatile type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=000)
/dev/sda1 on /run/media/sda1 type vfat (rw,relatime,gid=6,fmask=0007,dmask=0007,allow_utime=0020,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
```

cd to the USB device and list the files present on the device as shown below.

#### cd /run/media/sda1

#### ls

```
root@apzu_303:~# cd /run/media/sda1
root@apzu_303:/run/media/sda1# ls
ACR5533
APZU_HardwareFiles
MPSoC_HW
MPSoC_PetaLinux_2018_3_student_complete_v01.zip
MPSoC_SW_2018_3_student_complete_v01
MPSoC_TTC_2018_3_lab_0_v01_u96.pdf
System Volume Information
VirtualBox_Installation_Guide_2018_3_v1_l2.pdf
file_write.txt
helloworld.c
mymodule
mymodule.c
mymodule_old.c
newMEM.c
out.dts
petalinux-v2020.1-final-installer.run
system.dtb
ubuntu-16.04.5-desktop-amd64.iso
xilinx-zcu102-v2020.1-final.bsp
root@apzu_303:/run/media/sda1#
```

Using the cat command create a new text file named file\_write3.txt.

Run the following:

#### cat > file\_write3.txt

The file\_write3.txt will take input from the user. Type the following text into file\_write3.txt.

This is the first line in file\_write3.txt!

This is the second line.

To exit use.

**CTRL+d**

You can display the contents of the file\_write3.txt file with  
**cat file\_write3.txt**

```
root@apzu_303:/run/media/sdal# cat > file_write3.txt
This is the first line in file_write3.txt!
This is the second line.
root@apzu_303:/run/media/sdal# cat file_write3.txt
This is the first line in file_write3.txt!
This is the second line.
root@apzu_303:/run/media/sdal#
```

Running ls again will display the content of the USB drive with the new file\_write3.txt file included as seen below.

```
root@apzu_303:/run/media/sdal# ls
ACR5533
APZU_HardwareFiles
MPSoC_HW
MPSoC_PetaLinux_2018_3_student_complete_v01.zip
MPSoC_SW_2018_3_student_complete_v01
MPSoC_TTC_2018_3_lab_0_v01_u96.pdf
System Volume Information
VirtualBox_Installation_Guide_2018_3_v1_12.pdf
file_write.txt
file_write3.txt
helloworld.c
mymodule
mymodule.c
mymodule_old.c
newMEM.c
out.dts
petalinux-v2020.1-final-installer.run
system.dtb
ubuntu-16.04.5-desktop-amd64.iso
xilinx-zcu102-v2020.1-final.bsp
root@apzu_303:/run/media/sdal#
```

### 3.15 Test of I2C

The I2C interface is used to communicate with the ethernet MAC and Power Chip PMIC device.

The IIC link to the ethernet MAC can be used to read the ethernet MAC ID. The Linux provided i2ctransfer function can be used to read the ethernet MAC ID. In addition, the i2ctransfer function can be used to write the IIC EEPROM device. The default device address of the ethernet MAC is **0x50**.

The default device address for PMIC is **0x5E**.

An example of read and write to the ethernet MAC using i2ctransfer is given below. For example, the following command can be used to read the IIC EEPROM MAC at 0x50.

```
$ cd /usr/sbin
```

```
$ i2ctransfer 0 w1@0x50 0x0 r8
```

```
root@apzu_303:/usr/sbin# i2ctransfer 0 w1@0x50 0x0 r8
i2ctransfer: WARNING! This program can confuse your I2C bus
Continue? [y/N] y
0x00 0x00 0x00 0xc3 0x01 0x00 0x00 0x00
root@apzu_303:/usr/sbin#
```

The i2ctransfer program provides the following usage instructions:

```
fprintf(stderr,
    "Usage: i2ctransfer [-f] [-y] [-v] [-V] [-a] I2CBUS DESC [DATA] [DESC [DATA]]...\n"
    "  I2CBUS is an integer or an I2C bus name\n"
    "  DESC describes the transfer in the form: {r|w}LENGTH[@address]\n"
    "    1) read/write-flag 2) LENGTH (range 0-65535, or '?')\n"
    "    3) I2C address (use last one if omitted)\n"
    "  DATA are LENGTH bytes for a write message. They can be shortened by a suffix:\n"
    "    = (keep value constant until LENGTH)\n"
    "    + (increase value by 1 until LENGTH)\n"
    "    - (decrease value by 1 until LENGTH)\n"
    "    p (use pseudo random generator until LENGTH with value as seed)\n"
    "Example (bus 0, read 8 byte at offset 0x64 from EEPROM at 0x50):\n"
    " # i2ctransfer 0 w1@0x50 0x64 r8\n"
    "Example (same EEPROM, at offset 0x42 write 0xff 0xfe ... 0xf0):\n"
    " # i2ctransfer 0 w17@0x50 0x42 0xff-0xf0");
```

See results given below after running the provided example given in the instructions above.

```
root@apzu_303:/usr/sbin# i2ctransfer 0 w17@0x50 0x42 0xff-
i2ctransfer: WARNING! This program can confuse your I2C bus
Continue? [y/N] y
root@apzu_303:/usr/sbin# i2ctransfer 0 w1@0x50 0x42 r8
i2ctransfer: WARNING! This program can confuse your I2C bus
Continue? [y/N] y
0xff 0xfe 0xfd 0xfc 0xfb 0xfa 0xf9 0xf8
```

## Revision History

---

Release Date	Version	EGR/DOC	Description of Revision
05-MAR-2021	A	LMP/ARP	Release with product introduction.