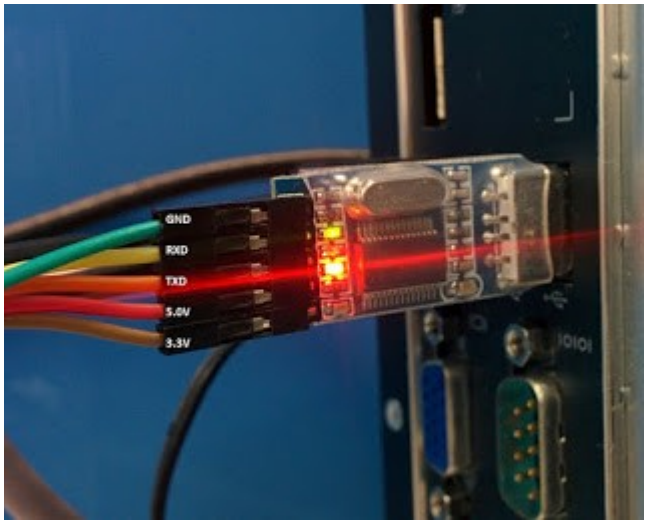
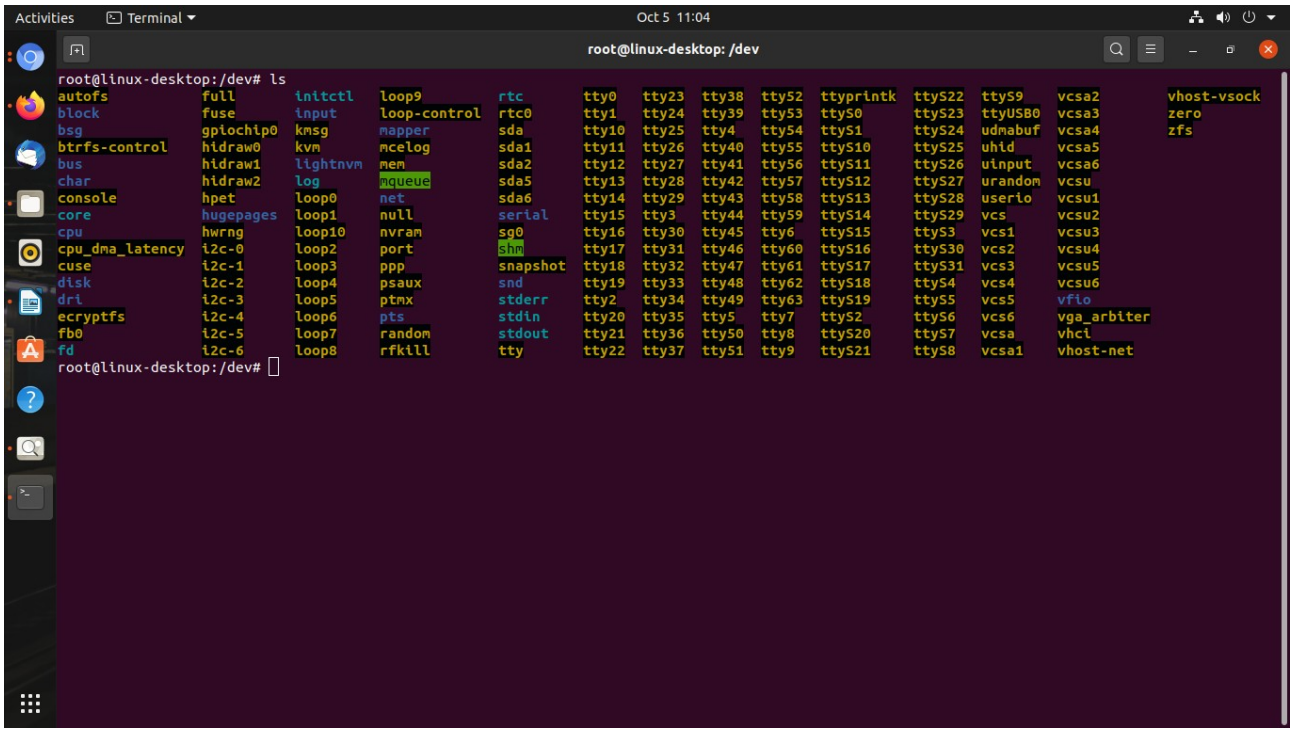
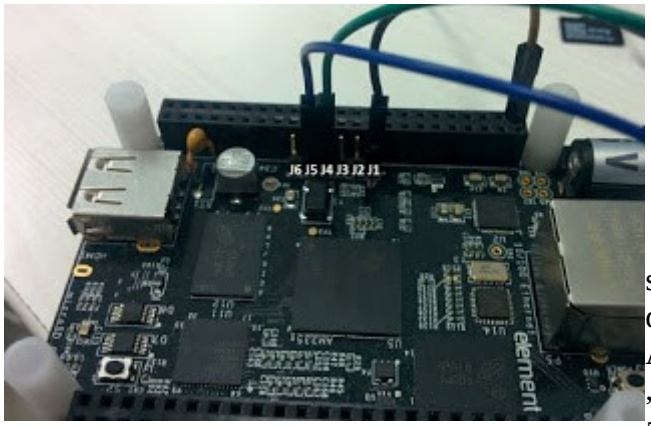
STARTING BEAGLEBONE BLACK U-BOOT WITHOUT SDCARD

step1. Connect uart with the pc and check the /dev/ttyUSB0 port is visible or not





step2. Connect the uart pins with the bbb with proper configuration(*cross check TX & RX pins)



step3. Then we have to download latest u-boot and ARM Cross Compiler ,follow the below links accordingly

a. (https://www.digikey.com/eewiki/display/linuxonarm/BeagleBone+Black)

b. (http://linuxkernel51.blogspot.com/2015/08/booting-beagle-bone-black-over-uart.html)
c. (https://www.youtube.com/watch?v=3y1LMNPoaJI)
d. (https://gist.github.com/eepp/6056325) ******
e. (http://www.blackpeppertech.com/pepper/tech-tree/boot-your-beaglebone-black-in-60-minutes/) *********(very important link)

f. *(http://linuxkernel51.blogspot.com/2015/08/beagleboneblack-boot-from-uart.html)-for u-boot
g. *(http://linuxkernel51.blogspot.com/2015/08/boot-beaglebone-black-with-nfs.html) for nfs

step4. Make a New project directory (p1) here we clone all essential tools

1.ARM CROSS COMPILER-GCC

==> wget -c https://releases.linaro.org/components/toolchain/binaries/6.5-2018.12/arm-linux-gnueabihf/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf.tar.xz

==> tar xf gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf.tar.xz
==> export CC=`pwd`/gcc-linaro-6.5.0-2018.12-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

** pwd** where you have extracted the above tool

2.U-BOOT

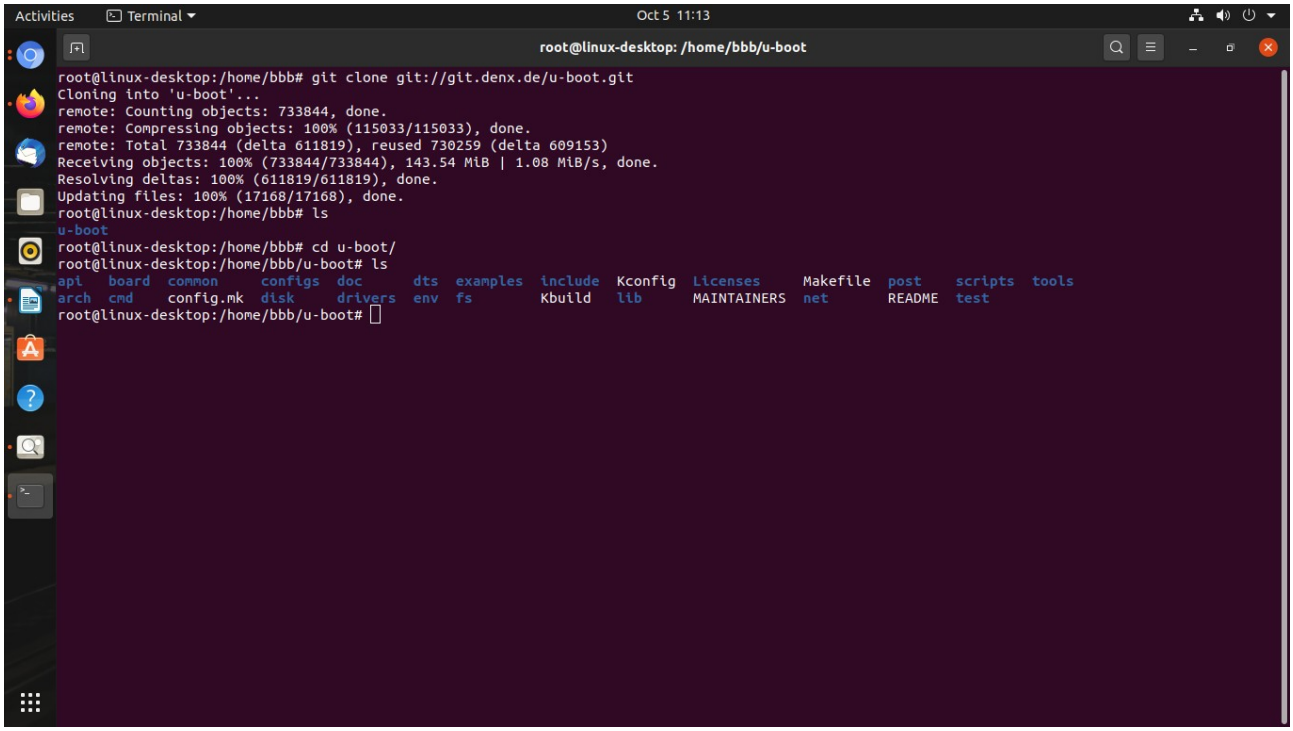
==> git clone -b v2019.04 https://github.com/u-boot/u-boot –depth=1
==> cd/u-boot
==> git checkout -b p1

PATCHES

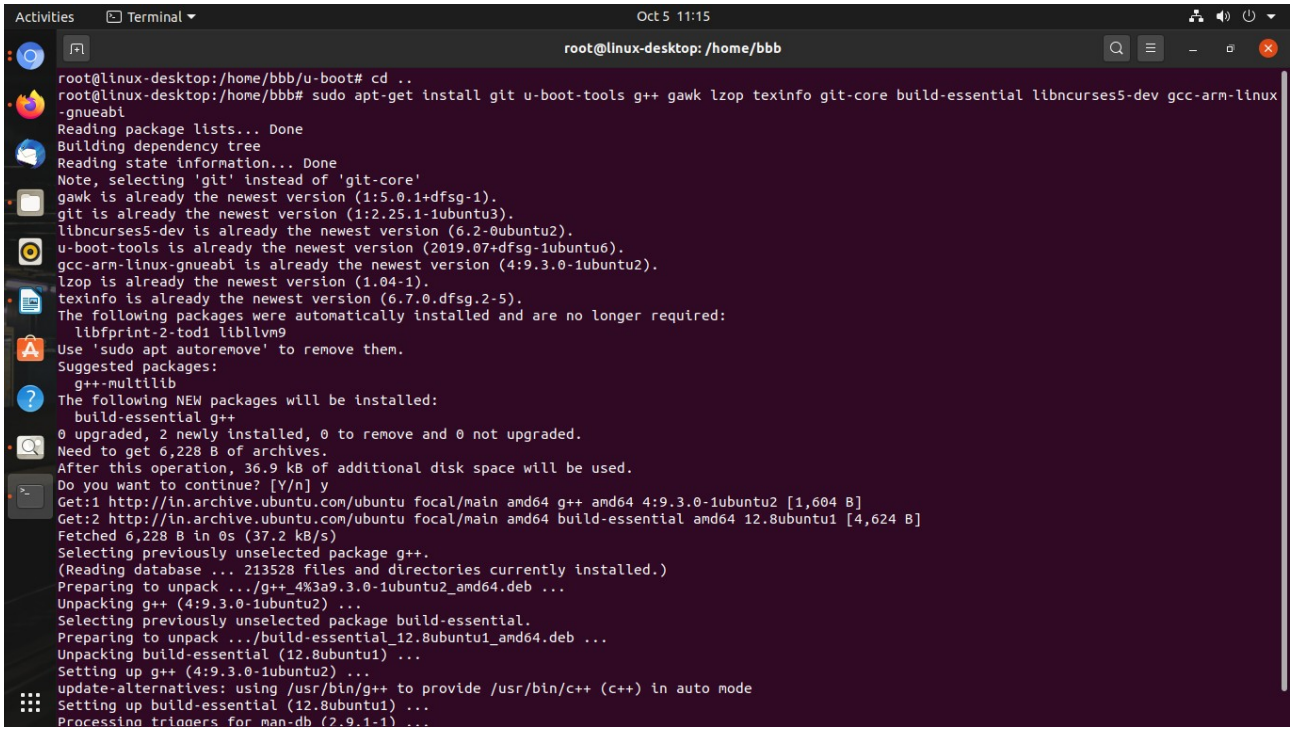
==> wget -c https://github.com/eewiki/u-boot-patches/raw/master/v2019.04/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch

==> wget -c https://github.com/eewiki/u-boot-patches/raw/master/v2019.04/0002-U-Boot-BeagleBone-Cape-Manager.patch

==> patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
==> patch -p1 < 0002-U-Boot-BeagleBone-Cape-Manager.patch



step5. ==> some essential tools that we need “sudo apt-get install git u-boot-tools g++ gawk lzop texinfo git-core build-essential libncurses5-dev gcc-arm-linux-gnueabi”



step6. Compile u-boot ,commands are available in the above links

** in some cases if the compiler is not properly exported then you will find error in that case you can locate the compiler's path like:-

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

**before compiling we distclean old .config files

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- distclean

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- am335x_evm_defconfig

** the above command will create a new .config file that will have all default configuration of am335x soc, if you want to edit these configuration type the following command

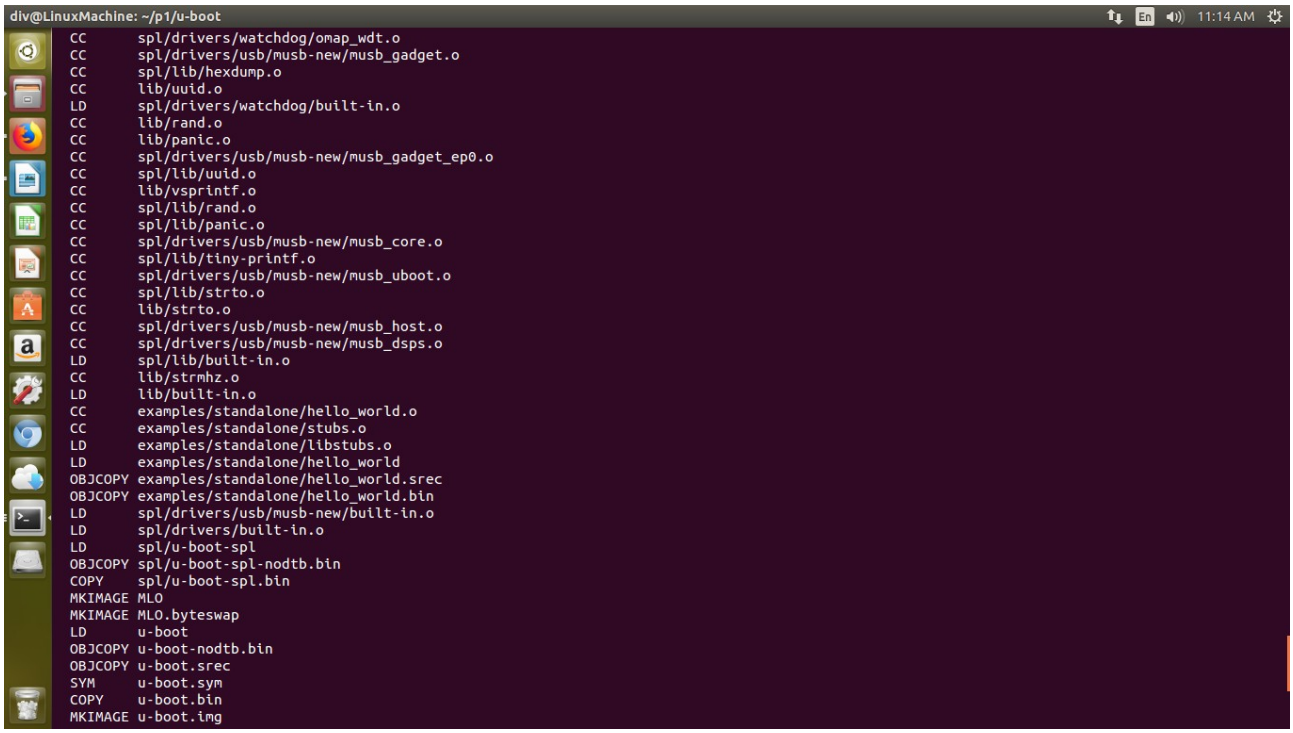
==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- menuconfig

**a new window will open here you can customize u-boot 's parameter

a. (3) delay in seconds before automatically booting

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- -j4

** the above command will start building the u-boot , after completion 3 main files will generate MLO, u-boot.img, u-boot-spl.bin inside u-boot folder

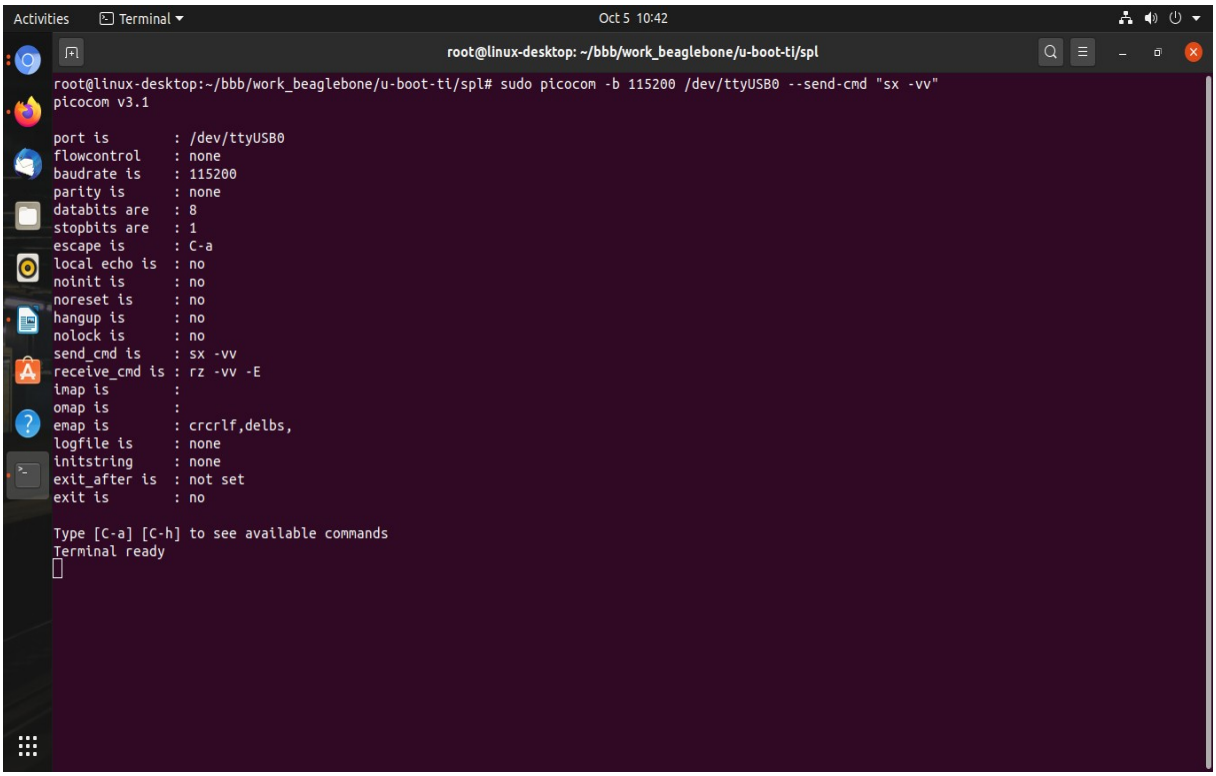


NOTE

* (before compiling we have to be inside the u-boot folder)
* (distclean every time before compiling)
* (on a new linux pc you will find many errors in the terminal i.e gnu,bison,m4 are missing or unable to compile something like that so just copy the error and search in google you will find the solution)
* (after compiling am335x_evm_defconfig or am335x_boneblack_vboot_defconfig , the ARM Cross compiler will create many new files inside the uboot folder . We need only 2 file right now for the bbb
    1. u-boot-spl.bin (this file you will find inside /*u-boot*/spl)
    2. u-boot.img (this file you will find inside /u-boot))
* (After these two files are generated i will suggest you to keep these files inside a same folder that will help you later on , copy u-boot.img file and paste it into spl folder)

step7. Install picocom or minicom any one of your choice
step8. I am using picocom

a. now change the folder to where the two files are located (/u-boot/spl) and open terminal from that folder
b. Now run " sudo picocom -b 115200 /dev/**ttyUSB0** --send-cmd "sx -vv" " (tty/USB0 is where my uart is connected)



c. Now press ctrl+a then ctrl+s to send the file (we are already inside the folder where the two files are located so type the first file name "u-boot-spl.bin")

```
root@linux-desktop:~/bbb/work_beaglebone/u-boot-ti/spl# sudo picocom -b 115200 /dev/ttyUSB0 --send-cmd "sx -vv"
picocom v3.1

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sx -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready

*** file: u-boot-spl.bin
$ sx -vv u-boot-spl.bin
Sending u-boot-spl.bin, 839 blocks: Give your local XMODEM receive command now.
```

now picocom is waiting for the board

***** NOW POWER ON THE BOARD BY PRESSING THE S2 SWITCH*****

```
baudrate is    : 115200
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sx -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready

*** file: u-boot-spl.bin
$ sx -vv u-boot-spl.bin
Sending u-boot-spl.bin, 839 blocks: Give your local XMODEM receive command now.
Xmodem sectors/kbytes sent:   0/ 0kRetry 0: Timeout on sector ACK
Retry 0: Got 00 for sector ACK
Retry 0: NAK on sector
Bytes Sent: 107520   BPS:771

Transfer complete

*** exit status: 0 ***

U-Boot SPL 2020.01-00516-g9d5d74c3cc (Oct 01 2020 - 17:10:46 +0530)
Trying to boot from UART
CC
```

now it should say transfer complete with exit status “0”, After successful transfer you will see the character 'C' as acknowledgement from BBB.

d. now again press ctrl+a then ctrl+s and send the 2nd file “u-boot.img”

```
Activities   Terminal   Oct 5 10:57
root@linux-desktop: ~/bbb/work_beaglebone/u-boot-ti/spl
nolock is      : no
send_cmd is    : sx -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready

*** file: u-boot-spl.bin
$ sx -vv u-boot-spl.bin
Sending u-boot-spl.bin, 839 blocks: Give your local XMODEM receive command now.
Bytes Sent: 107520   BPS:5944

Transfer complete

*** exit status: 0 ***

U-Boot SPL 2020.01-00516-g9d5d74c3cc (Oct 01 2020 - 17:10:46 +0530)
Trying to boot from UART
CC
*** file: u-boot.img
$ sx -vv u-boot.img
Sending u-boot.img, 5977 blocks: Give your local XMODEM receive command now.
Xmodem sectors/kbytes sent:   0/ 0kRetry 0: NAK on sector
Retry 0: NAK on sector
Retry 0: NAK on sector
Bytes Sent: 765184   BPS:9029

Transfer complete

*** exit status: 0 ***
```

now for the 2nd file it should also say transfer complete with exit status “0”

e. right after now the board will restart automatically , press space bar when it ask to stop the u-boot

```
Activities   Terminal   Oct 5 11:01
root@linux-desktop: ~/bbb/work_beaglebone/u-boot-ti/spl
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Loading Environment from FAT... <ethaddr> not set. Validating first E-fuse MAC
Net:   eth0: ethernet@4a100000
Warning: usb_ether MAC addresses don't match:
Address in ROM is          de:ad:be:ef:00:01
Address in environment is  b0:d5:cc:fc:67:4d
, eth1: usb_ether
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1(part 0) is current device
Scanning mmc 1:1...
switch to partitions #0, OK
mmc1(part 0) is current device
SD/MMC found on device 1
** Invalid partition 2 **
## Error: "bootcmd_nand0" not defined
starting USB...
Bus usb@47401800: Port not available.
ethernet@4a100000 Waiting for PHY auto negotiation to complete.............................CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC◆◆◆◆◆◆◆◆◆◆◆◆◆◆C
U-Boot SPL 2016.11-rc3-00002-g73df7f7 (Nov 04 2016 - 15:20:36)
Trying to boot from MMC2


U-Boot 2016.11-rc3-00002-g73df7f7 (Nov 04 2016 - 15:20:36 -0500), Build: jenkins-github_Bootloader-Builder-479

CPU  : AM335X-GP rev 2.1
I2C:   ready
DRAM:  512 MiB
Reset Source: Power-on reset has occurred.
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
Net:   eth0: MII MODE
cpsw
Press SPACE to abort autoboot in 2 seconds
=>
```

now our u-boot is ready..

==========>
AT PAGE 12

3.KERNEL

STEP1. Clone the source code inside p1

==> git clone https://github.com/beagleboard/linux.git
==> cd linux
==> git checkout 4.1

STEP2. clean previous builds/.config/binaries

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- distclean

STEP4. build .config with default configuration

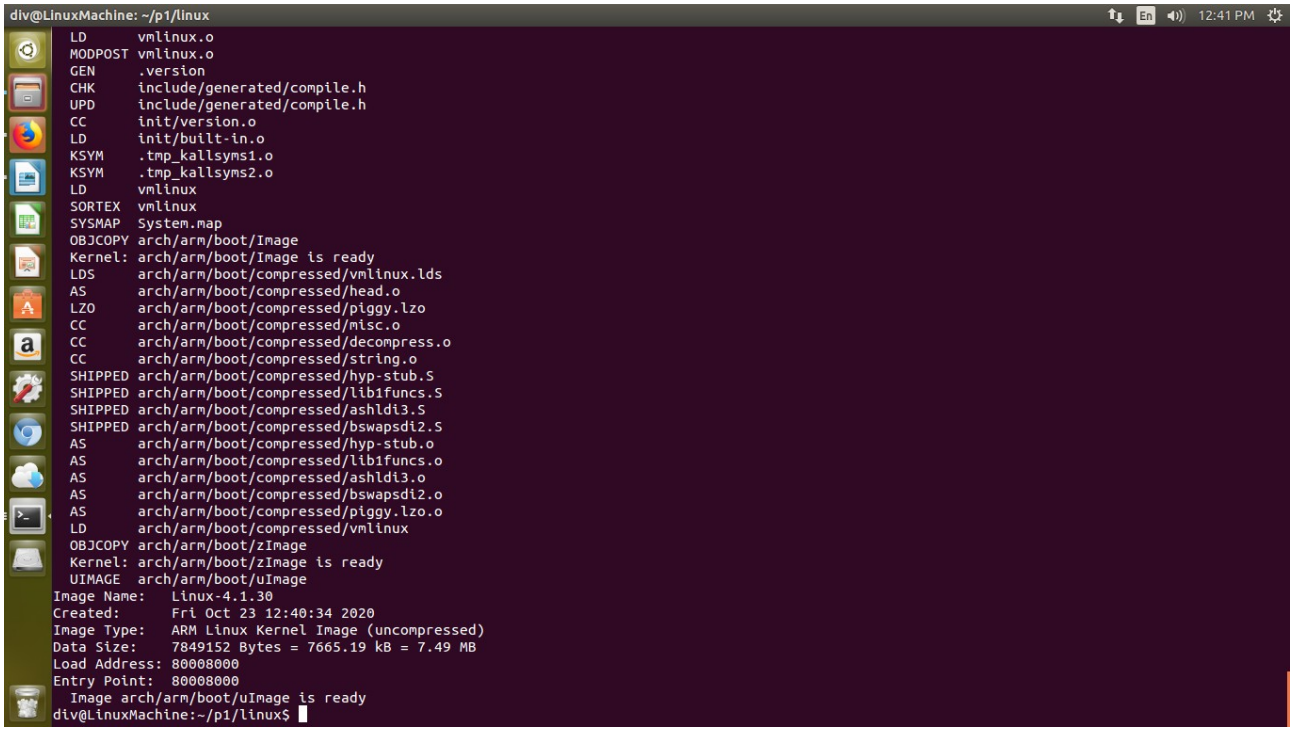
==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- bb.org_defconfig

STEP5. [Optional] – to change default configuration settings

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- menuconfig

STEP6. build kernel and device tree

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- uImage dtbs LOADADDR=0x80008000 -j4



now images are ready i.e (uImage,zImage,am335x-boneblack.dtb)
uImage, zImage ==(inside linux/arch/arm/boot/)
am335x-boneblack.dtb==(inside linux/arch/arm/boot/dts/)

** step7 & step8 are only required if you want to make your own Root File System in our case we will be using Debian 10.4 so no need to perform step7 and step8

STEP7. build kernel modules

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- -j4 modules

STEP8. install modules into RFS (busybox in our case) – **to be done after building Busybox**

==> make ARCH=arm CROSS_COMPILE=/home/div/p2/soft/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf- INSTALL_MOD_PATH=*~/custom_beaglebone/busybox/custom_rfs/lib/modules* modules_install

4.Root File System

STEP1. Download debian 10 from the link inside p1

==> wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-10.4-minimal-armhf-2020-05-10.tar.xz

(you can also use other root file system like Ansgtrom , Buildroot etc)

STEP2. Extract the file

==> tar xf debian-10.4-minimal-armhf-2020-05-10.tar.xz

**Now all images are ready in order to boot the beaglebone , now we setup the host machine with TFTP and NFS to transfer the images to beaglebone' RAM and boot it

**then from RAM we will partition the internal emmc of beaglebone and transfer the bootloader ,kernel , rootfile system into emmc

Setting up the host pc

5.TFTP

STEP1. Install a TFTP server on host machine

==> sudo apt-get install tftpd
==> sudo apt-get install tftpd-hda
==> sudo apt-get install tftpd-server

STEP2. Now create a file in /etc/xinetd.d/tftp and add the following and save it

```
service tftp
{
disable       = no
protocol      = udp
port          = 69
socket_type   = dgram
wait          = yes
user          = root
server        = /usr/sbin/in.tftpd
server_args   = /var/lib/tftpboot -s
per_source    = 11
cps           = 100 2
flags         = IPv4
}
```

STEP3. Now we create a folder in which we will keep the files which we want to transfer through TFTP Protocol

==> mkdir /var/lib/tftpboot

*The above path should match with “server_args” in the above file
*give permission to the folder so that anybody can read and write

==>chmod 777 /var/lib/tftpboot

STEP4. Restart the server

==> sudo service xinetd restart

STEP5. Now we copy ““uImage, am335x-boneblack.dtb, u-boot-spl.bin, MLO, u-boot.img”” to /var/lib/tftpboot folder

STEP6. Restart host machine

6.NFS

STEP1. Install a NFS server on your host machine

==> sudo apt-get install nfs-kernel-server

STEP2. Create a directory that you want to share as your Network Root File System

==> mkdir -p /home/div/p2/bbb-work/rootfs

*give permission to the folder

==> chmod 777 /home/div/p2/bbb-work/rootfs

STEP3. To export a directory by NFS (Network File System) as your root filesystem, edit your /etc/exports configuration file and add this line

/home/div/p2/bbb-work/rootfs 192.168.0.21(rw,sync,no_subtree_check,no_root_squash,insecure)

STEP4. Now restart the server

==> sudo service nfs-kernel-server restart

*in the above line 192.168.0.21 denotes ip address of beaglebone which we will going to assign

STEP5. Now extract the debian-10.4-minimal-armhf-2020-05-10.tar.xz into rootfs folder

CONTINUED WITH U-BOOT.....

==> setenv ipaddr 192.168.0.21

==>setenv serverip 192.168.0.29 *(check your host machine ip address)

==>setenv gw_ip 192.168.0.1 *(check your host machine gateway address)

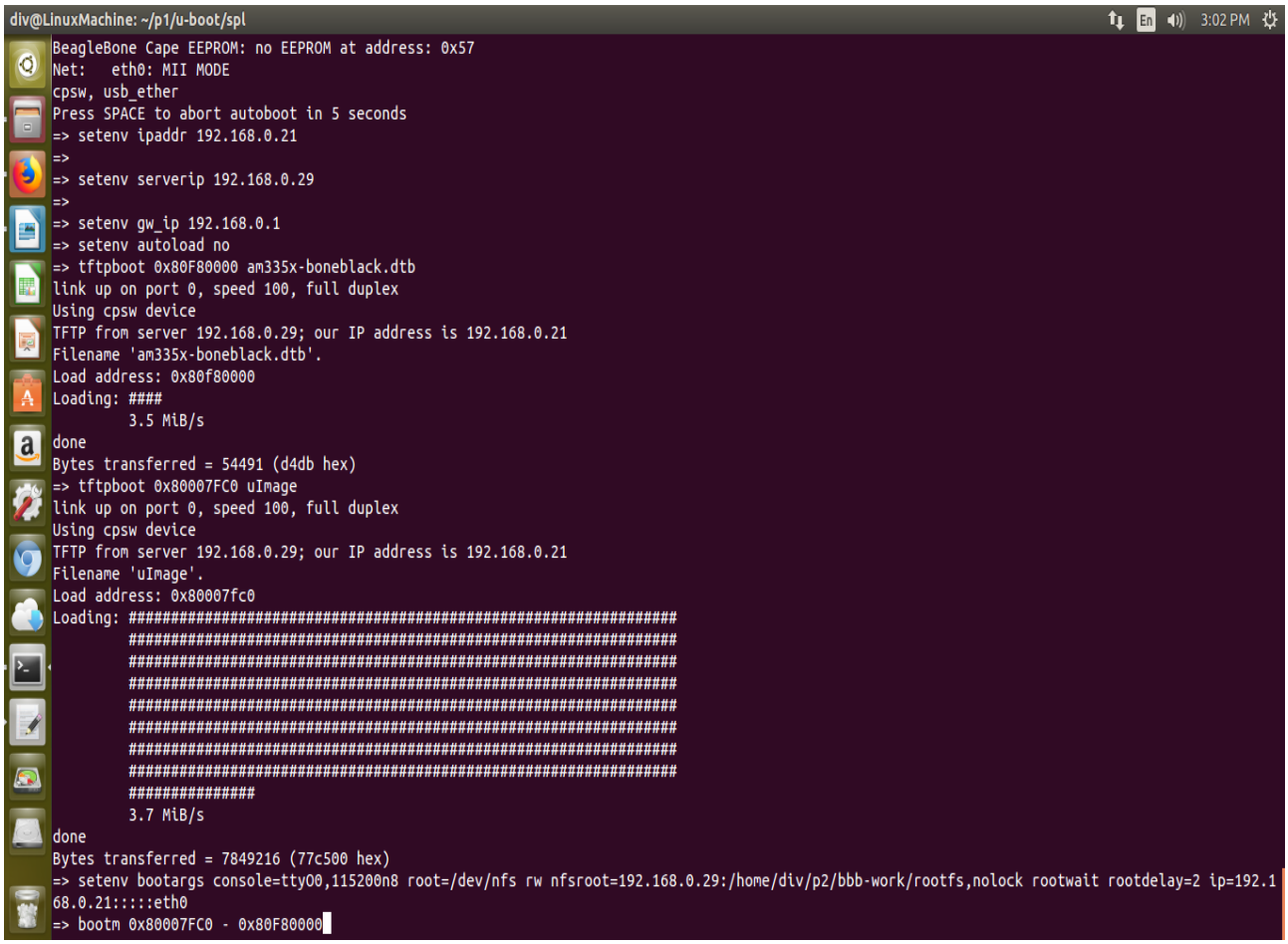
==>setenv autoload no

==>tftpboot 0x80F80000 am335x-boneblack.dtb

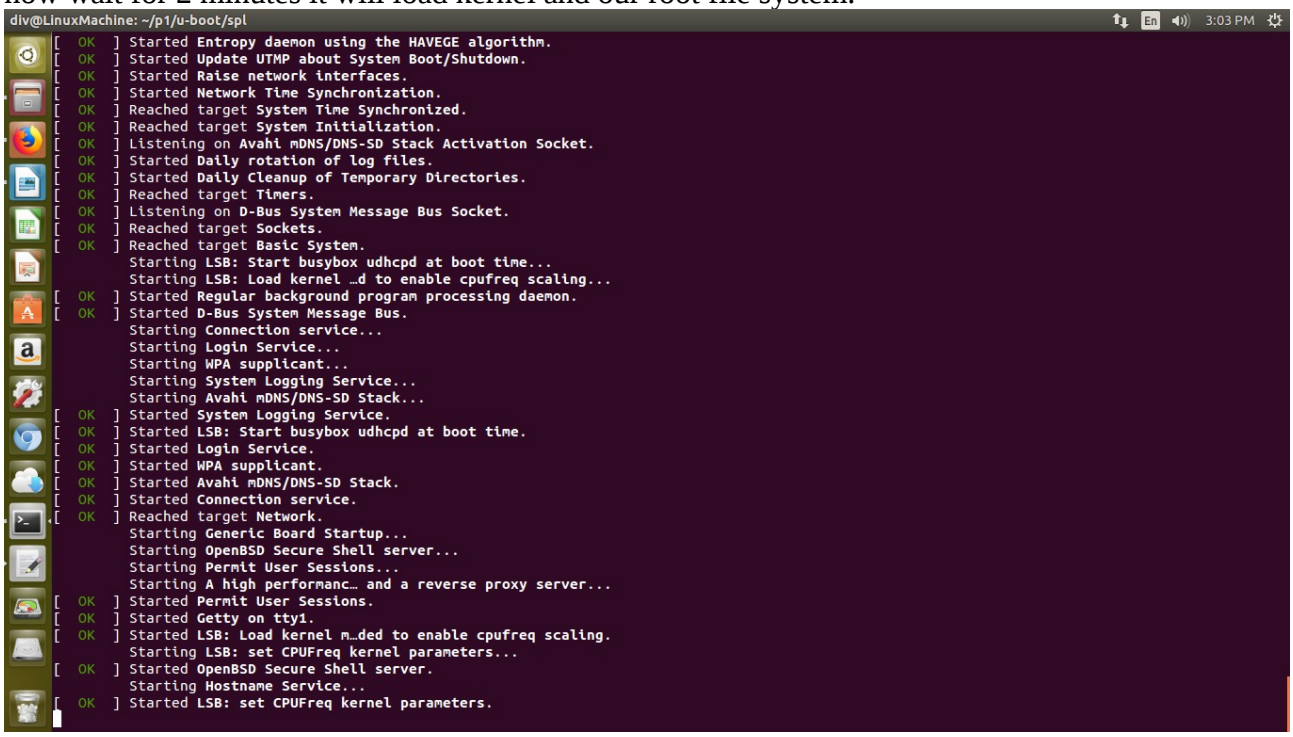
==>tftpboot 0x80007FC0 uImage

==>setenv bootargs console=ttyO0,115200n8 root=/dev/nfs rw nfsroot=192.168.0.29:/home/div/p2/bbb-work/rootfs,nolock rootwait rootdelay=2 ip=192.168.0.21:::::eth0

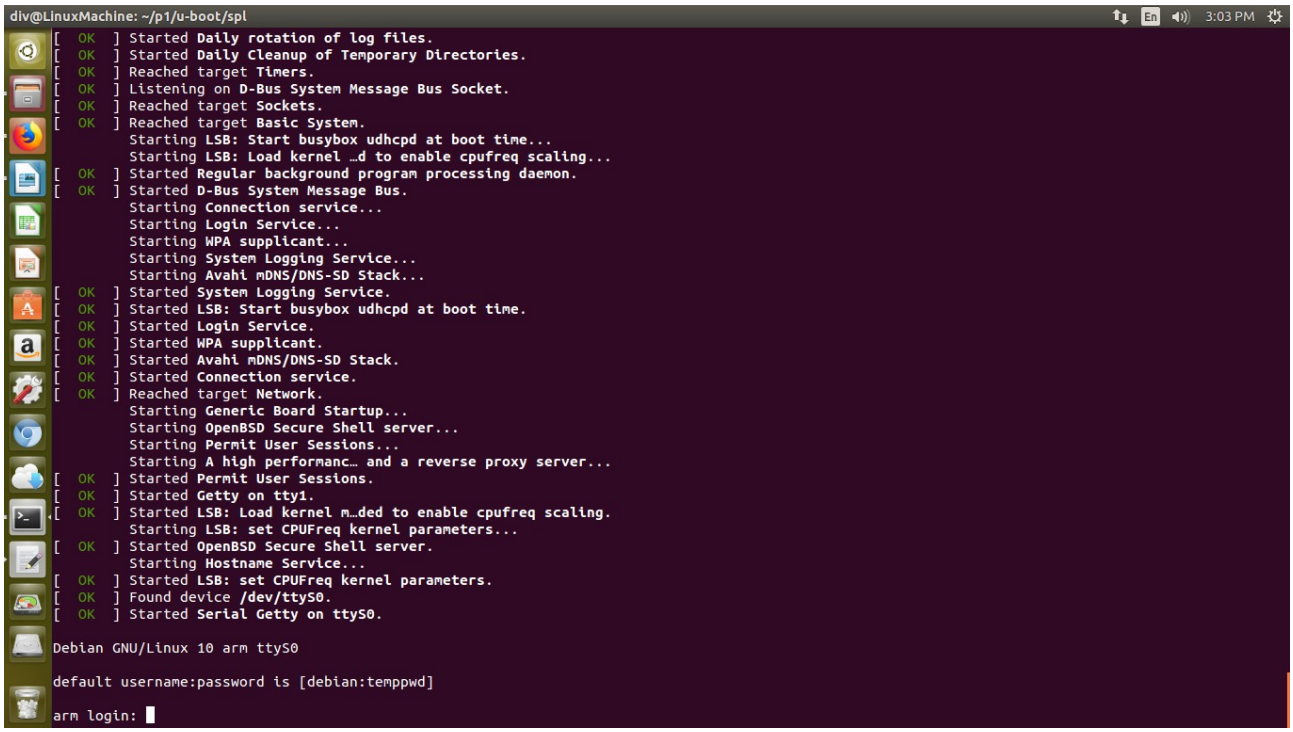
==>bootm 0x80007FC0 – 0x80F80000



now wait for 2 minutes it will load kernel and our root file system.



if everything goes well then you will be able to see the login screen like this



arm login: root
password: root

```
div@LinuxMachine: ~/p1/u-boot/spl                                   3:07 PM
         Starting Connection service...
         Starting Login Service...
         Starting WPA supplicant...
         Starting System Logging Service...
         Starting Avahi mDNS/DNS-SD Stack...
[  OK  ] Started System Logging Service.
[  OK  ] Started LSB: Start busybox udhcpd at boot time.
[  OK  ] Started Login Service.
[  OK  ] Started WPA supplicant.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Started Connection service.
[  OK  ] Reached target Network.
         Starting Generic Board Startup...
         Starting OpenBSD Secure Shell server...
         Starting Permit User Sessions...
         Starting A high performanc… and a reverse proxy server...
[  OK  ] Started Permit User Sessions.
[  OK  ] Started Getty on tty1.
[  OK  ] Started LSB: Load kernel m…ded to enable cpufreq scaling.
         Starting LSB: set CPUFreq kernel parameters...
[  OK  ] Started OpenBSD Secure Shell server.
         Starting Hostname Service...
[  OK  ] Started LSB: set CPUFreq kernel parameters.
[  OK  ] Found device /dev/ttyS0.
[  OK  ] Started Serial Getty on ttyS0.

Debian GNU/Linux 10 arm ttyS0

default username:password is [debian:temppwd]

arm login: root
Password:
Last login: Mon May 11 00:24:46 UTC 2020 from 192.168.0.29 on pts/0

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@arm:~#
```

Here we have successfully booted the beaglebone, at this point all the images are sitting inside RAM

From here we will transfer the images to emmc so that next time beaglebone will be able to boot itself

CREATE PARTITION IN EMMC

STEP1.first we Check the partition table present in emmc

==>fdisk -l

```
root@arm:~# fdisk -l
Disk /dev/mmcblk0: 3.6 GiB, 3825205248 bytes, 7471104 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: 0x647f0d99

Device         Boot  Start     End Sectors  Size Id Type
/dev/mmcblk0p1 *      2048  133119  131072   64M  c W95 FAT32 (LBA)
/dev/mmcblk0p2      133120 7471103 7337984  3.5G 83 Linux


Disk /dev/mmcblk0boot1: 4 MiB, 4194304 bytes, 8192 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


Disk /dev/mmcblk0boot0: 4 MiB, 4194304 bytes, 8192 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
root@arm:~#
```

here i have already created 2 partitions in emmc for you it may varry so we will start fresh by erasing everything

STEP2.

==> fdisk /dev/mmcblk0 *(listed above)
==> o - this clears the existing partitions
==> p - this lists all partition tables on the card (there should be none)

==> n - create a new partition
==> p - primary partition
==> 1 - partition number
==> 2048 - default value for the first sector
==> +64M - last sector / partition size
==> t - change the partition type (select partition 1)
==> c - change tha partition type to "W95 FAT32 (LBA)"
==> a - set the bootable flag for the selected partition (1)
==> n - create a new partition
==> p - primary partition
==> 2 - partition number
==> hit Enter to choose the default (next available) value for the first sector
==> hit Enter to choose the default (last) value for the last sector
==> p - this lists all partition tables on the card (there should be two)
==> w - write all the above changes to disk
==> umount /dev/mmcblk0p1; mkfs.vfat -F 32 /dev/mmcblk0p1 - format the first partition
==> umount /dev/mmcblk0p2; mkfs.ext4 /dev/mmcblk0p2 - format the second partition

STEP3.

Now we mount mmcblk0p1 to a folder boot so that we can copy the files (MLO, u-boot.img, am335x-boneblack.dtb, uImage)

==> mkdir -p /p1/boot
==> mount /dev/mmcblk0p1 boot

STEP4.

Now we mount mmcblk0p2 to a folder root so that we can copy the root file system (debian-10.4-minimal-armhf-2020-05-10.tar.xz)

==> mkdir -p/p1/root
==> mount /dev/mmcblk0p2 root

MOVE TO HOST MACHINE

HOST MACHINE

STEP1. Open terminal and install ssh

==> sudo apt install ssh

STEP2. Copy files into a folder (MLO, u-boot.img, am335x-boneblack.dtb, uImage,uEnv.txt) now being inside the folder run following commands

==>scp uImage root@192.168.0.21:/p1/boot
==>scp am335x-boneblack.dtb root@192.168.0.21:/p1/boot
==>scp MLO root@192.168.0.21:/p1/boot
==>scp u-boot.img root@192.168.0.21:/p1/boot
==>scp uEnv.txt root@192.168.0.21:/p1/boot

STEP3. Extract the tar and Move the folder content to root folder

==>scp (content inside debian-10.4-minimal-armhf-2020-05-10.tar.xz) root@192.168.0.21:/p1/root

MOVE BACK TO BOARD

STEP5. Now unmount boot
==> umount boot

STEP6. Now unmount root
==> umount root

STEP7. Reboot the board
==> reboot

From here the board will restart and u-boot will come up then after waiting for 3 seconds u-boot will read the text file called uEnv.txt (copied by us in the 1st partition of emmc) and initialize the parameters to load kernel image , device tree blob , and our root file system wait for 2 – 3 minutes and the board will boot up on its own.

How to make uEnv.txt file

STEP1. Make a text file (using nano or vi) name as uEnv.txt
==> nano uEnv.txt

STEP2. Copy the below content inside the file and save it

```
console=ttyO0,115200n8
ipaddr=192.168.0.21
serverip=192.168.0.35
gw_ip=192.168.0.1
loadaddr=0x82000000
fdtaddr=0x88000000
mmcdev=0
mmcpart=1
loadfrommmc=load mmc 1:1 ${loadaddr} uImage;load mmc 1:1 ${fdtaddr} am335x-boneblack.dtb
linuxbootargs=setenv bootargs console=${console} root=/dev/mmcblk0p2 rw rootfstype=ext4
uenvcmd=setenv autoload no; run loadfrommmc; run linuxbootargs; bootm ${loadaddr} - $
{fdtaddr}
```