

Linux Kernel Training: Lecture 2

Booting the BEAGLEBONE BLACK

Sam Protsenko
<joe.skb7@gmail.com>

June 1, 2020

Agenda

1. U-Boot Basics
2. IDE Considerations
3. Workshop: Bringing up the BBB

U-Boot Basics

Embedded Board

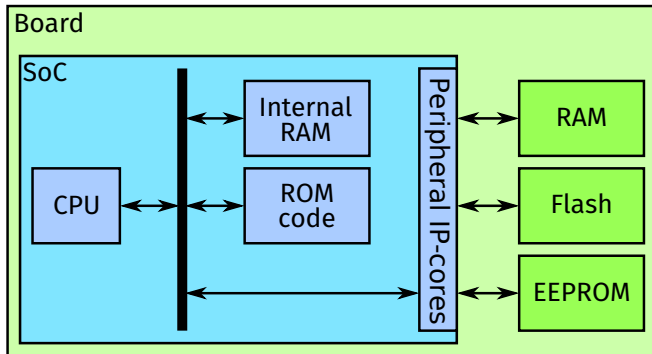


Figure 1: Simplified view of Embedded board

What is Bootloader?

ROM code has limitations:

- Doesn't know about RAM
- Doesn't know board name
- Not flexible enough

ROM code

?

kernel

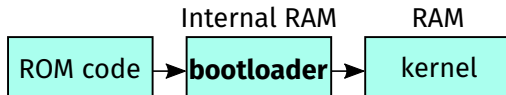
What is Bootloader?

ROM code has limitations:

- Doesn't know about RAM
- Doesn't know board name
- Not flexible enough

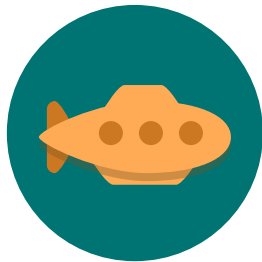
Bootloader to the rescue:

- Resides in flash (can be upgraded)
- Able to configure RAM
- Knows boot procedure
- Convenient features



Why U-Boot?

- Bootloader for Embedded boards
 - Popular for Android devices
 - Adoption in automotive
- GPLv2
- 13 architectures (consider ARM)
- ~300 boards
- Device drivers, lib routines
- Resembles Linux kernel a lot
- Scripting, extensive command set



U-Boot Features

- Boots from various sources
- Boots various OSs
- Monitor (U-Boot shell)
 - Commands
 - Environment
- 2 stage boot (SPL + U-Boot)
- “Falcon” mode (SPL only)

Two Stage Boot

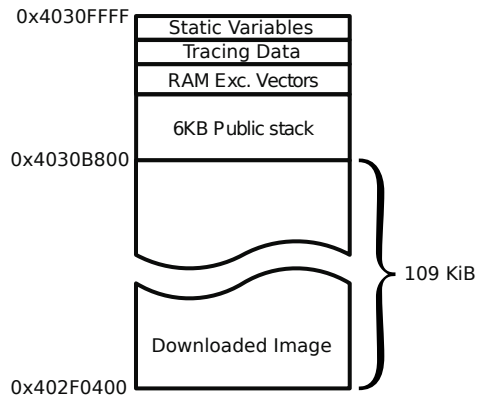
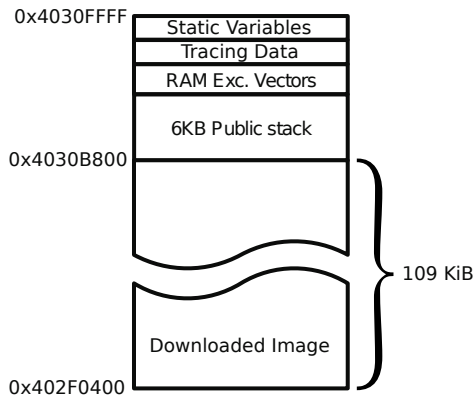


Figure 2: SRAM layout (from AM335x TRM)

Two Stage Boot



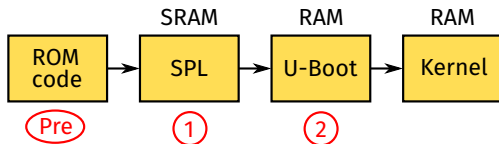
But bootloader is bigger!

For **BEAGLEBONE BLACK**, `u-boot.img` is 391 KiB.

Figure 2: SRAM layout (from AM335x TRM)

Two Stage Boot (cont'd)

Let's add an intermediate stage (SPL):

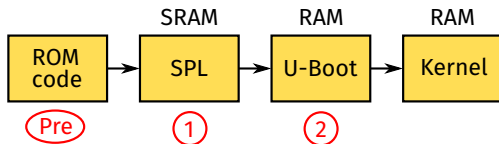


For **BEAGLEBONE BLACK**:

Stage	Size
SPL	75 KiB
U-Boot	391 KiB

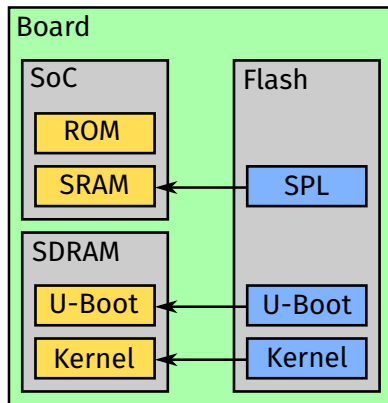
Two Stage Boot (cont'd)

Let's add an intermediate stage (SPL):



For **BEAGLEBONE BLACK**:

Stage	Size
SPL	75 KiB
U-Boot	391 KiB



Boot Sources

ROM code boot source is usually selected:

- Via **SYSBOOT** DIP switch
- By hard-wiring SYSBOOT lines (pull-up/pull-down)
- Using some USER button
- By inserting SD card (“Card Detect” pin)

Boot Sources

ROM code boot source is usually selected:

- Via **SYSBOOT** DIP switch
- By hard-wiring SYSBOOT lines (pull-up/pull-down)
- Using some USER button
- By inserting SD card (“Card Detect” pin)

U-Boot can boot kernel from:

- Flash devices (eMMC, SD card)
- Network boot (TFTP, NFS)
- Peripheral boot (USB, serial console)

Flashing Methods

Most commonly used flashing methods:

- Via USB:
 - fastboot (Android)
 - DFU
- Via SD card

Flashing Methods

Most commonly used flashing methods:

- Via USB:
 - fastboot (Android)
 - DFU
- Via SD card

To unbrick the board (bad U-Boot on eMMC):

- Boot from SD card and re-flash
- YMODEM boot (via UART)
- USB peripheral boot
- Use JTAG

U-Boot Shell

- U-Boot has a “monitor” program
- Command line interface
- A set of commands is implemented
- Resembles Bash (has scripting capabilities)
- Press “Space” to get into U-Boot shell

U-Boot Shell

- U-Boot has a “monitor” program
- Command line interface
- A set of commands is implemented
- Resembles Bash (has scripting capabilities)
- Press “Space” to get into U-Boot shell

Most commonly used commands:

```
bdinfo, bootm, bootz, crc32, dfu, dhcp, env,  
fastboot, fatload, fdt, gpt, i2c, md, mmc, mw,  
part, ping, reset, run, tftpboot, version, ...
```

U-Boot Environment

- Keeps all U-Boot shell variables
- Just a set of strings in **key=value** format

U-Boot Environment

- Keeps all U-Boot shell variables
- Just a set of strings in **key=value** format

Set default environment:

```
=> env default -f -a
```

```
=> env save
```

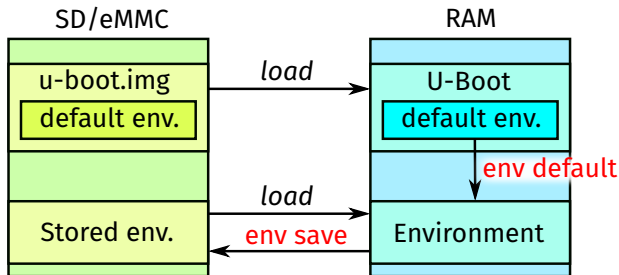
U-Boot Environment

- Keeps all U-Boot shell variables
- Just a set of strings in **key=value** format

Set default environment:

```
=> env default -f -a
```

```
=> env save
```



```
=> env print
```

```
board_name=A335BNLT
```

```
board_rev=0A5A
```

```
board_serial=1813BBBK4642
```

```
bootargs=
```

```
bootcmd=if test ${boot_fit} -eq 1; then ...
```

```
fdtfile=am335x-boneblack.dtb
```

```
findfdt=if test $board_name = A335BONE; then ...
```

```
partitions=uuid_disk=${uuid_gpt_disk}; ...
```

Use-case: Format eMMC

Using **gpt** command:

```
=> gpt write mmc 1 $partitions
```

Use-case: Format eMMC

Using **gpt** command:

```
=> gpt write mmc 1 $partitions
```

or Android way:

```
=> fastboot 0
```

```
$ fastboot oem format
```


Use-case: Format eMMC

Using **gpt** command:

```
=> gpt write mmc 1 $partitions
```

or Android way:

```
=> fastboot 0  
$ fastboot oem format
```

- Consider **\$partitions** variable...
- Check with **part list mmc 1**

Use-case: Flashing eMMC

Via DFU:

```
=> setenv dfu_alt_info $dfu_alt_info_emmc  
=> dfu 0 mmc 1  
$ dfu-util -D MLO -a MLO.raw
```

Use-case: Flashing eMMC

Via DFU:

```
=> setenv dfu_alt_info $dfu_alt_info_emmc  
=> dfu 0 mmc 1  
$ dfu-util -D MLO -a MLO.raw
```

Via fastboot (Android way):

```
=> fastboot 0  
$ fastboot flash rootfs rootfs.img
```

Use-case: Boot Linux

```
=> setenv mmcdev 1  
=> setenv bootpart 1:2  
=> run findfdt  
=> run mmcboot
```

Use-case: Boot Linux

```
=> setenv mmcdev 1
=> setenv bootpart 1:2
=> run findfdt
=> run mmcboot
```

mmcboot=

```
mmc dev ${mmcdev}; mmc rescan;
load mmc ${bootpart} ${loadaddr} /boot/zImage
load mmc ${bootpart} ${fdtaddr} /boot/${fdtfile}
setenv bootargs console=tty00,115200n8 root=... rw ...
bootz ${loadaddr} - ${fdtaddr}
```

Boot Sequence

`do_bootm_linux()` calls:

- 1:
 - `boot_prep_linux()`
 - `boot_setup_linux()`
 - `image_setup_libfdt()`
 - Populates `/chosen` node with `bootargs`, `initrd-start`, etc.
- 2:
 - `boot_jump_linux()`
 - `kernel_entry(0, machid, r2)`
 - FDT blob address is passed to kernel via `r2`

Boot Methods Table

Boot method	Description
SD card boot	<ul style="list-style-type: none">- Unbrick the board- Doesn't touch eMMC
eMMC boot	<ul style="list-style-type: none">- Regular boot- Fastest and easiest for user
TFTP boot	<ul style="list-style-type: none">- Network boot- Doesn't touch eMMC- Volatile RootFS (RAM disk)
NFS boot	<ul style="list-style-type: none">- Network boot- Transparent RootFS (from host)- Useful for kernel development

IDE Considerations

IDE for Kernel Development: Vim

Pros:

- Very fast
- Available everywhere
- A lot of kernel hackers using it

Cons:

- Learning curve rather steep
- Some functionality is missing
- Requires some configuration

Indexing example (simplified):

```
make ARCH=arm SUBARCH=omap2 cscope tags
```

Details: <https://stackoverflow.com/a/33682137/3866447>

IDE for Kernel Development: Vim (cont'd)

```
gpio-max732x.c (~repos/linux-mainline/drivers/gpio) - VIM
Press ? for help
1475
1476 static irqreturn_t max732x_irq_handler(int irq, void *devid)
1477 {
1478     struct max732x_chip *chip = devid;
1479     uint8_t pending;
1480     uint8_t level;
1481
1482     pending = max732x_irq_pending(chip);
1483
1484     if (!pending)
1485         return IRQ_HANDLED;
1486
1487     do {
1488         level = __ffs(pending);
1489         handle_nested_irq(irq_find_mapping(chip->gpio_chip.irq_domain,
1490                                         level));
1491     } while (pending &= ~(1 << level));
1492
1493     while (pending);
1494
1495     return IRQ_HANDLED;
1496 }
1497
1498 static int max732x_irq_setup(struct max732x_chip *chip,
1499                             const struct i2c_device_id *id)
1500 {
1501     struct i2c_client *client = chip->client;
1502     struct max732x_platform_data *pdata = dev_get_platdata(&client->dev);
1503     int has_irq = max732x_features[id->driver_data] >> 32;
1504     int irq_base = 0;
1505     int ret;
1506
1507     if ((pdata && pdata->irq_base) || client->irq)
1508         && has_irq != INT_NONE) {
1509         if (pdata)
1510             irq_base = pdata->irq_base;
1511         chip->irq_features = has_irq;
1512     }
1513 }
1514
1515 #endif
1516
1517 #endif
1518
1519 #endif
1520
1521 #endif
1522
1523 #endif
1524
1525 #endif
1526
1527 #endif
1528
1529 #endif
1530
1531 #endif
1532
1533 #endif
1534
1535 #endif
1536
1537 #endif
1538
1539 #endif
1540
1541 #endif
1542
1543 #endif
1544
1545 #endif
1546
1547 #endif
1548
1549 #endif
1550
1551 #endif
1552
1553 #endif
1554
1555 #endif
1556
1557 #endif
1558
1559 #endif
1560
1561 #endif
1562
1563 #endif
1564
1565 #endif
1566
1567 #endif
1568
1569 #endif
1570
1571 #endif
1572
1573 #endif
1574
1575 #endif
1576
1577 #endif
1578
1579 #endif
1580
1581 #endif
1582
1583 #endif
1584
1585 #endif
1586
1587 #endif
1588
1589 #endif
1590
1591 #endif
1592
1593 #endif
1594
1595 #endif
1596
1597 #endif
1598
1599 #endif
1600
1601 #endif
1602
1603 #endif
1604
1605 #endif
1606
1607 #endif
1608
1609 #endif
1610
1611 #endif
1612
1613 #endif
1614
1615 #endif
1616
1617 #endif
1618
1619 #endif
1620
1621 #endif
1622
1623 #endif
1624
1625 #endif
1626
1627 #endif
1628
1629 #endif
1630
1631 #endif
1632
1633 #endif
1634
1635 #endif
1636
1637 #endif
1638
1639 #endif
1640
1641 #endif
1642
1643 #endif
1644
1645 #endif
1646
1647 #endif
1648
1649 #endif
1650
1651 #endif
1652
1653 #endif
1654
1655 #endif
1656
1657 #endif
1658
1659 #endif
1660
1661 #endif
1662
1663 #endif
1664
1665 #endif
1666
1667 #endif
1668
1669 #endif
1670
1671 #endif
1672
1673 #endif
1674
1675 #endif
1676
1677 #endif
1678
1679 #endif
1680
1681 #endif
1682
1683 #endif
1684
1685 #endif
1686
1687 #endif
1688
1689 #endif
1690
1691 #endif
1692
1693 #endif
1694
1695 #endif
1696
1697 #endif
1698
1699 #endif
1700
1701 #endif
1702
1703 #endif
1704
1705 #endif
1706
1707 #endif
1708
1709 #endif
1710
1711 #endif
1712
1713 #endif
1714
1715 #endif
1716
1717 #endif
1718
1719 #endif
1720
1721 #endif
1722
1723 #endif
1724
1725 #endif
1726
1727 #endif
1728
1729 #endif
1730
1731 #endif
1732
1733 #endif
1734
1735 #endif
1736
1737 #endif
1738
1739 #endif
1740
1741 #endif
1742
1743 #endif
1744
1745 #endif
1746
1747 #endif
1748
1749 #endif
1750
1751 #endif
1752
1753 #endif
1754
1755 #endif
1756
1757 #endif
1758
1759 #endif
1760
1761 #endif
1762
1763 #endif
1764
1765 #endif
1766
1767 #endif
1768
1769 #endif
1770
1771 #endif
1772
1773 #endif
1774
1775 #endif
1776
1777 #endif
1778
1779 #endif
1780
1781 #endif
1782
1783 #endif
1784
1785 #endif
1786
1787 #endif
1788
1789 #endif
1790
1791 #endif
1792
1793 #endif
1794
1795 #endif
1796
1797 #endif
1798
1799 #endif
1800
1801 #endif
1802
1803 #endif
1804
1805 #endif
1806
1807 #endif
1808
1809 #endif
1810
1811 #endif
1812
1813 #endif
1814
1815 #endif
1816
1817 #endif
1818
1819 #endif
1820
1821 #endif
1822
1823 #endif
1824
1825 #endif
1826
1827 #endif
1828
1829 #endif
1830
1831 #endif
1832
1833 #endif
1834
1835 #endif
1836
1837 #endif
1838
1839 #endif
1840
1841 #endif
1842
1843 #endif
1844
1845 #endif
1846
1847 #endif
1848
1849 #endif
1850
1851 #endif
1852
1853 #endif
1854
1855 #endif
1856
1857 #endif
1858
1859 #endif
1860
1861 #endif
1862
1863 #endif
1864
1865 #endif
1866
1867 #endif
1868
1869 #endif
1870
1871 #endif
1872
1873 #endif
1874
1875 #endif
1876
1877 #endif
1878
1879 #endif
1880
1881 #endif
1882
1883 #endif
1884
1885 #endif
1886
1887 #endif
1888
1889 #endif
1890
1891 #endif
1892
1893 #endif
1894
1895 #endif
1896
1897 #endif
1898
1899 #endif
1900
1901 #endif
1902
1903 #endif
1904
1905 #endif
1906
1907 #endif
1908
1909 #endif
1910
1911 #endif
1912
1913 #endif
1914
1915 #endif
1916
1917 #endif
1918
1919 #endif
1920
1921 #endif
1922
1923 #endif
1924
1925 #endif
1926
1927 #endif
1928
1929 #endif
1930
1931 #endif
1932
1933 #endif
1934
1935 #endif
1936
1937 #endif
1938
1939 #endif
1940
1941 #endif
1942
1943 #endif
1944
1945 #endif
1946
1947 #endif
1948
1949 #endif
1950
1951 #endif
1952
1953 #endif
1954
1955 #endif
1956
1957 #endif
1958
1959 #endif
1960
1961 #endif
1962
1963 #endif
1964
1965 #endif
1966
1967 #endif
1968
1969 #endif
1970
1971 #endif
1972
1973 #endif
1974
1975 #endif
1976
1977 #endif
1978
1979 #endif
1980
1981 #endif
1982
1983 #endif
1984
1985 #endif
1986
1987 #endif
1988
1989 #endif
1990
1991 #endif
1992
1993 #endif
1994
1995 #endif
1996
1997 #endif
1998
1999 #endif
2000
2001 #endif
2002
2003 #endif
2004
2005 #endif
2006
2007 #endif
2008
2009 #endif
2010
2011 #endif
2012
2013 #endif
2014
2015 #endif
2016
2017 #endif
2018
2019 #endif
2020
2021 #endif
2022
2023 #endif
2024
2025 #endif
2026
2027 #endif
2028
2029 #endif
2030
2031 #endif
2032
2033 #endif
2034
2035 #endif
2036
2037 #endif
2038
2039 #endif
2040
2041 #endif
2042
2043 #endif
2044
2045 #endif
2046
2047 #endif
2048
2049 #endif
2050
2051 #endif
2052
2053 #endif
2054
2055 #endif
2056
2057 #endif
2058
2059 #endif
2060
2061 #endif
2062
2063 #endif
2064
2065 #endif
2066
2067 #endif
2068
2069 #endif
2070
2071 #endif
2072
2073 #endif
2074
2075 #endif
2076
2077 #endif
2078
2079 #endif
2080
2081 #endif
2082
2083 #endif
2084
2085 #endif
2086
2087 #endif
2088
2089 #endif
2090
2091 #endif
2092
2093 #endif
2094
2095 #endif
2096
2097 #endif
2098
2099 #endif
2100
2101 #endif
2102
2103 #endif
2
```

IDE for Kernel Development: Eclipse

Pros:

- Easy to use
- Nice tools (indexing, macro unwrapping, etc)

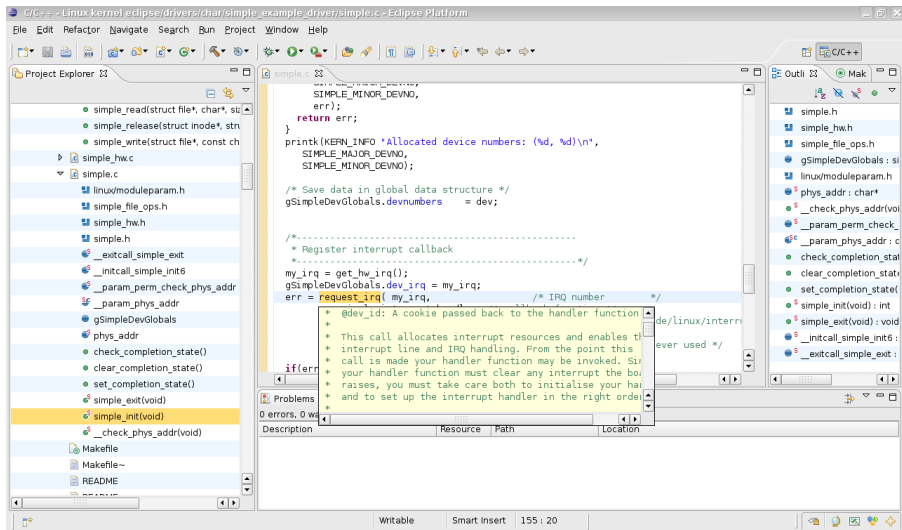
Cons:

- Very slow
- Uses a lot of resources
- Not available on servers, etc.

Be sure to configure CDT for kernel development. Details:

https://wiki.eclipse.org/HowTo_use_the_CDT_to_navigate_Linux_kernel_source

IDE for Kernel Development: Eclipse (cont'd)



Other possible IDEs:

- Emacs
- KDevelop
- QtCreator

Kernel source web-browser (very useful):

- elixir.bootlin.com

Questions so far?

Take Five

Workshop: Bringing up the BBB

Short Quiz

1. Who didn't manage to build all the SW?
2. Who doesn't have a laptop along?

Training Centre Infrastructure

Training Centre PC info:

- Press F9 on boot (show boot menu)
- Select second drive (TS64GSSD370S, 64 GB)
- Login: Lin-Ker
- Password: 123

What does it have?

- Ubuntu 18.04, internet connection, prepared environment
- Toolchains (**/opt/***)
- U-Boot, Linux kernel, BusyBox (see **~/repos/***)
- **Missing:** TRM, datasheet, schematic, BBB instructions guide

Notes:

- Better to use home laptop to keep the single environment
- We only have one card reader :(

Hardware Equipment

- **The board:** BEAGLEBONE BLACK (Rev C)
- **Serial cable:** TTL-232R-3V3
- **OTG USB cable:** mini-USB to USB
- **micro-SD card + adapter:** 8 GB, class 10
- **Ethernet patch cord:** for network boot and networking tasks
- **Development kit:** a set of hardware, will be used later

ESD Safety Note



- Discharge before touching
- Use ESD mat and wrist strap
- Don't wear ESD unsafe clothes
- Remove power before touching



Now we are going to:

- Connect mini-USB cable
 - Powering the board (low consumption use-case)
 - Flashing (**fastboot**, **dfu-util**)
- Connect serial console cable (white dot = black wire = GROUND)
 - Will be used via **minicom** tool
- Insert SD card to the slot (once it's flashed with software)

Connectivity: Serial Console

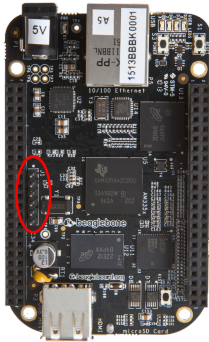


Figure 3: BEAGLEBONE BLACK serial connection



Figure 4: TTL-232R-3V3 FTDI cable

Connectivity: Serial Console (cont'd)

- Board uses UART port (TTL levels, 3.3V) to communicate:
 - Rx line for receiving characters from host
 - Tx line for transceiving characters to host
- Chip in adapter cable converts UART <-> USB
 - FT232R (FTDI): reliable, recommended
 - PL2303: cheap alternative
- Chip driver exposes `/dev/ttyUSB` file (char device)
- **minicom** (or similar tool) allows user to interact with board via `/dev/ttyUSB`

Connectivity: Client USB (OTG)

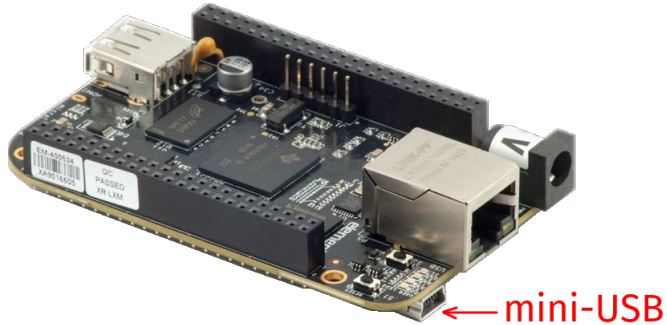


Figure 5: BEAGLEBONE BLACK USB connection

Configure and Run minicom

- Run minicom configuration:

```
$ sudo minicom -s
```

- Select “Serial port setup” menu item and choose next settings:

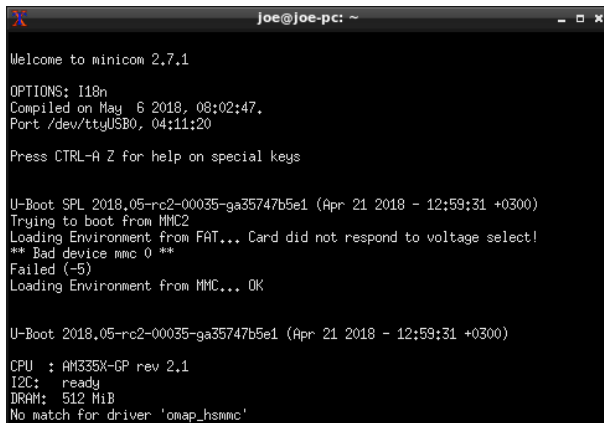
- Serial Device: `/dev/ttyUSB0`
- Bps/Par/Bits: 115200 8N1
- Hardware Flow Control: **No**

- Then select “Save setup as dfl” and “Exit from Minicom”

- Run minicom:

```
$ sudo minicom
```

Configure and Run minicom (cont'd)

A terminal window titled 'joe@joe-pc: ~' with standard window controls. The output shows minicom 2.7.1 welcome message, options (I18n), compilation date (May 6 2018), and port (/dev/ttyUSB0). It then shows U-Boot booting from MMC2, failing to load environment from FAT due to a bad device, and successfully loading from MMC. Finally, it shows U-Boot version and hardware details (CPU: AM335X-GP rev 2.1, I2C: ready, DRAM: 512 MiB) and a warning about a missing driver.

```
joe@joe-pc: ~  
Welcome to minicom 2.7.1  
OPTIONS: I18n  
Compiled on May 6 2018, 08:02:47.  
Port /dev/ttyUSB0, 04:11:20  
  
Press CTRL-A Z for help on special keys  
  
U-Boot SPL 2018.05-rc2-00035-ga35747b5e1 (Apr 21 2018 - 12:59:31 +0300)  
Trying to boot from MMC2  
Loading Environment from FAT... Card did not respond to voltage select!  
** Bad device mmc 0 **  
Failed (-5)  
Loading Environment from MMC... OK  
  
U-Boot 2018.05-rc2-00035-ga35747b5e1 (Apr 21 2018 - 12:59:31 +0300)  
  
CPU : AM335X-GP rev 2.1  
I2C: ready  
DRAM: 512 MiB  
No match for driver 'omap_hsmmc'
```

Figure 6: Terminal with minicom

Formatting SD Card (page 1)

- Insert your SD card in your laptop (use adapter)
- Locate SD card device file (should be `/dev/mmcblk0`):

```
$ sudo dmesg | tail
```

```
$ sudo fdisk -l
```

- Unmount SD card (mounted automatically by Ubuntu):

```
$ sudo umount /dev/mmcblk0p1
```

```
$ sudo umount /dev/mmcblk0p2
```

- Clear SD card MBR:

```
$ sudo dd if=/dev/zero of=/dev/mmcblk0 bs=1M count=1
```

- Create new partition table:

```
$ sudo sfdisk /dev/mmcblk0 << EOF
2048,100M,0x0c,*
,,L,-
EOF
```

- Format both partitions:

```
$ sudo mkfs.vfat -F 32 -n "boot" /dev/mmcblk0p1
$ sudo mkfs.ext4 -F -L "rootfs" /dev/mmcblk0p2
```

- Check new partitions:

```
$ sudo fdisk -l
```

- You should see something like this:

Device	Boot	Start	Size	Id	Type
/dev/mmcblk0p1	*	2048	100M	c	W95 FAT32 (LBA)
/dev/mmcblk0p2		206848	14.9G	83	Linux

Prepare Bootable SD Card (page 1)

- Unmount SD card (mounted automatically by Ubuntu):

```
$ sudo umount /dev/mmcblk0p1
```

```
$ sudo umount /dev/mmcblk0p2
```

- Mount partitions:

```
$ sudo mkdir /mnt/{boot,rootfs}
```

```
$ sudo mount /dev/mmcblk0p1 /mnt/boot
```

```
$ sudo mount /dev/mmcblk0p2 /mnt/rootfs
```

- Check that partitions mounted properly:

```
$ mount | grep mmcblk
```

Prepare Bootable SD Card (page 2)

- Copy U-Boot files to SD card (“boot” partition):

```
$ cd ~/repos/u-boot
```

```
$ sudo cp MLO u-boot.img /mnt/boot
```

- Copy rootfs files to SD card (“rootfs” partition):

```
$ cd ~/repos/busybox/_install
```

```
$ sudo cp -R . /mnt/rootfs
```


- Unmount SD card:
\$ sudo umount /mnt/boot
\$ sudo umount /mnt/rootfs
- Remove your SD card from your laptop

Booting from SD Card

- Unplug mini-USB cable from the board
- Insert SD card in board's slot
- Press and hold **USER/BOOT** button
- Plug mini-USB cable back to board
- Release **USER/BOOT** button
- BusyBox will be loaded from SD card

Assignments

Assignment #1

- Finish up the whole BBB guide:
 - Do all kinds of boot, using 3rd chapter (eMMC boot, TFPT boot, NFS boot)
 - Use previously built software (from lecture 1 assignment)
- Proof:
 - Send me screenshots that prove eMMC boot works for you
 - Send me screenshots that prove NFS boot works for you

Assignment #2

- Install and configure IDE of your choice (use links from slides)
- In your IDE (work dir must be the kernel source dir):
 - Open `init/main.c` file
 - Find `start_kernel()` function implementation
 - In `start_kernel()`, look for `setup_arch()` call
 - Jump to `setup_arch()` (using IDE capabilities)
 - Get back to `start_kernel()` (using IDE capabilities)
 - Find where `start_kernel()` is being called from (using IDE capabilities)
- Proof: send me screenshot that shows your configured IDE

Thank you!

Appendix: Abbreviations

Abbreviations

- BBB - BEAGLEBONE BLACK
- DFU - Device Firmware Upgrade
- DTB - Device Tree Blob
- eMMC - Embedded MMC (MultiMedia Card)
- ESD - Electrostatic Discharge
- FDT - Flattened Device Tree (the same as DTB)
- FTDI - Future Technology Devices International (company name)
- GPIO - General Purpose Input Output
- GPT - GUID Partition Table
- IDE - Integrated Development Environment
- LED - Light-Emitting Diode

Abbreviations (cont'd)

- MBR - Master Boot Record
- MLO - MMC Loader (TI image format for SPL)
- NFS - Network File System
- OTG - USB On-The-Go
- PCB - Printed Circuit Board
- PHY - Physical layer chip
- PMIC - Power Management IC (Integrated Circuit)
- RootFS - Root File System
- SD card - Secure Digital card
- SoC - System on Chip
- SPL - Secondary Program Loader (first part of U-Boot)

Abbreviations (cont'd)

- TFTP - Trivial File Transport Protocol
- TRM - Technical Reference Manual
- TTL - Transistor-to-Transistor Logic
- UART - Universal Asynchronous Receiver-Transmitter