Yocto on BeagleBone Black & Raspberry Pi 3

BeagleBone Black Specification

- Texas Instruments AM335x (ARM Cortex-A8 CPU)
- 512MB DDR3 RAM
- 4GB on-board eMMC storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit micro-controllers
- USB client for power & communications
- USB host, Ethernet, HDMI (micro)
- 2x 46-pin headers with expansion buses (I2C, SPI, UART)
- Various expansion boards (capes)

Building Yocto Image for BeagleBone Black

1. Source the environment:

source poky/oe-init-build-env build_bbb

2. Modify build_bbb/local.conf:
3. MACHINE ?= "beaglebone-yocto"

#MACHINE ??= "qemux86_64"

4. Trigger build:

bitbake core-image-minimal

5. The output directory tmp/deploy/images/beaglebone-yocto/ will contain:
    - First-level bootloader MLO
    - Second-level bootloader U-Boot
    - Kernel image
    - Device tree blobs
    - Root filesystem archive
    - Modules archive

Booting Process in BeagleBone Black

The AM335x is a complex piece of hardware but has limited internal RAM (128 kB). Multiple bootloader stages are needed to systematically unlock the full functionality of the device.

The Four Bootloader Stages:

1. **ROM Code**
    - The first stage bootloader is flashed in ROM on the device by Texas Instruments.
    - Automatically runs at power-on reset (POR).
    - Hardcoded into the device and cannot be changed.
    - Functions:
        - Configures stack, watchdog timer, PLL, and system clocks.
        - Checks boot sources for the next bootloader (SPL).
        - Moves the next bootloader into memory.
    - Boot sources order:
        - MMC1 (onboard eMMC), MMC0 (external uSD), UART0, USB0.
        - Holding down boot switch (S2) changes boot order.
2. **SPL (Secondary Program Loader)**
    - Minimal version of U-Boot.
    - Initializes CPU and loads full U-Boot.
    - Name: MLO
    - Located on the active first partition of MMC (formatted as FAT12/16/32).
3. **U-Boot**
    - Allows control over boot environment via serial terminal.
    - Configures boot arguments, loads kernel and device tree.
    - Environment variables stored in uEnv.txt.
    - Loads kernel and DTS into memory and boots with command-line arguments.
4. **Linux Kernel**
    - Mounts root filesystem (default second partition, ext3 formatted).

Boot Stage	Terminology #1	Terminology #2	Actual Program Name
1	Primary Program Loader	-	ROM code
2	Secondary Program Loader (SPL)	1st stage bootloader	u-boot SPL
3	-	2nd stage bootloader	u-boot
4	-	-	kernel

Reference: StackOverflow - Bootloader Stages

Raspberry Pi 3 Specification

- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2
- Networking: 10/100 Ethernet, 802.11n Wi-Fi
- Bluetooth 4.1
- Storage: microSD
- GPIO: 40-pin header

Booting Process in Raspberry Pi 3

Stage 1 Booting - OnChip ROM

- GPU - On, CPU - Off, SDRAM - Off
- Boot ROM looks for bootcode.bin on the SD card.
- Loads bootcode.bin into L2 cache memory.

Stage 2 Booting - bootcode.bin

- GPU - On, CPU - Off, SDRAM - On
- Executed on VideoCore GPU.
- Enables SDRAM and loads the third stage bootloader start.elf.

Stage 3 Booting - start.elf

- GPU - On, CPU - On, SDRAM - On
- Loads config.txt and cmdline.txt.
- Loads the kernel image and device tree.

Flashing Images to SD Card

1. Unmount partitions:

umount /dev/sdb1

2. Partition the SD card using fdisk.
3. Create boot and root partitions.
4. Format the partitions:
5. sudo mkfs.vfat -n "BOOT" /dev/sdb1

sudo mkfs.ext4 -L "ROOT" /dev/sdb2

6. Copy bootloader and kernel files.
7. Unmount and remove the SD card.

Building Yocto Image for Raspberry Pi 3

1. Clone BSP layers:

git clone git://git.yoctoproject.org/meta-raspberrypi

2. Set up environment:

source poky/oe-init-build-env build_pi

3. Modify local.conf:

MACHINE = "raspberrypi3"

4. Build the image:

bitbake core-image-minimal