Porting Linux to an Embedded Platform without pain

(or at least with less pain :-))

Fabio Estevam
NXP Semiconductors

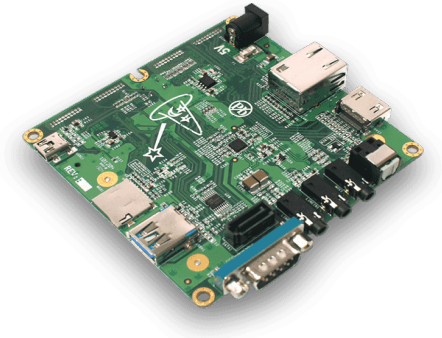
Agenda

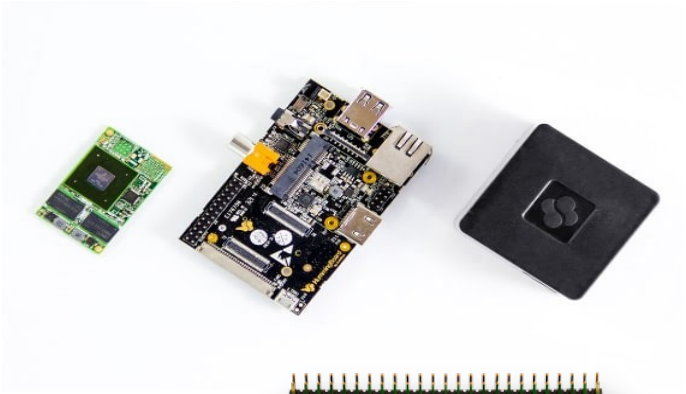
- Main components of an Embedded Linux system
- Porting U-Boot to a custom board
- Porting Linux to a custom board
- Using Buildroot to provide a full bootable image
- The importance of using mainline components in the final product

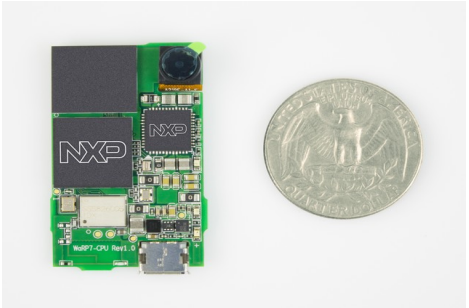
What is Embedded Linux?

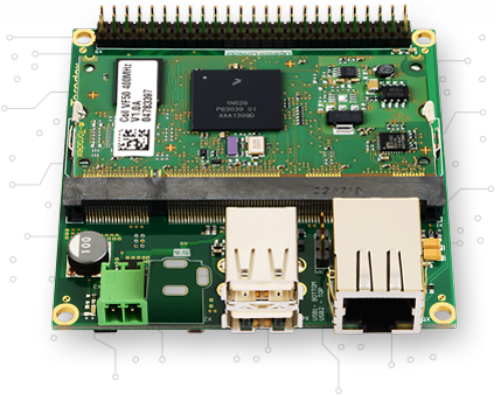
- Think of Linux running outside of a traditional PC

- Examples: network routers, smart TVs, e-books, tablets, smartphone, car infotainment, smart watches

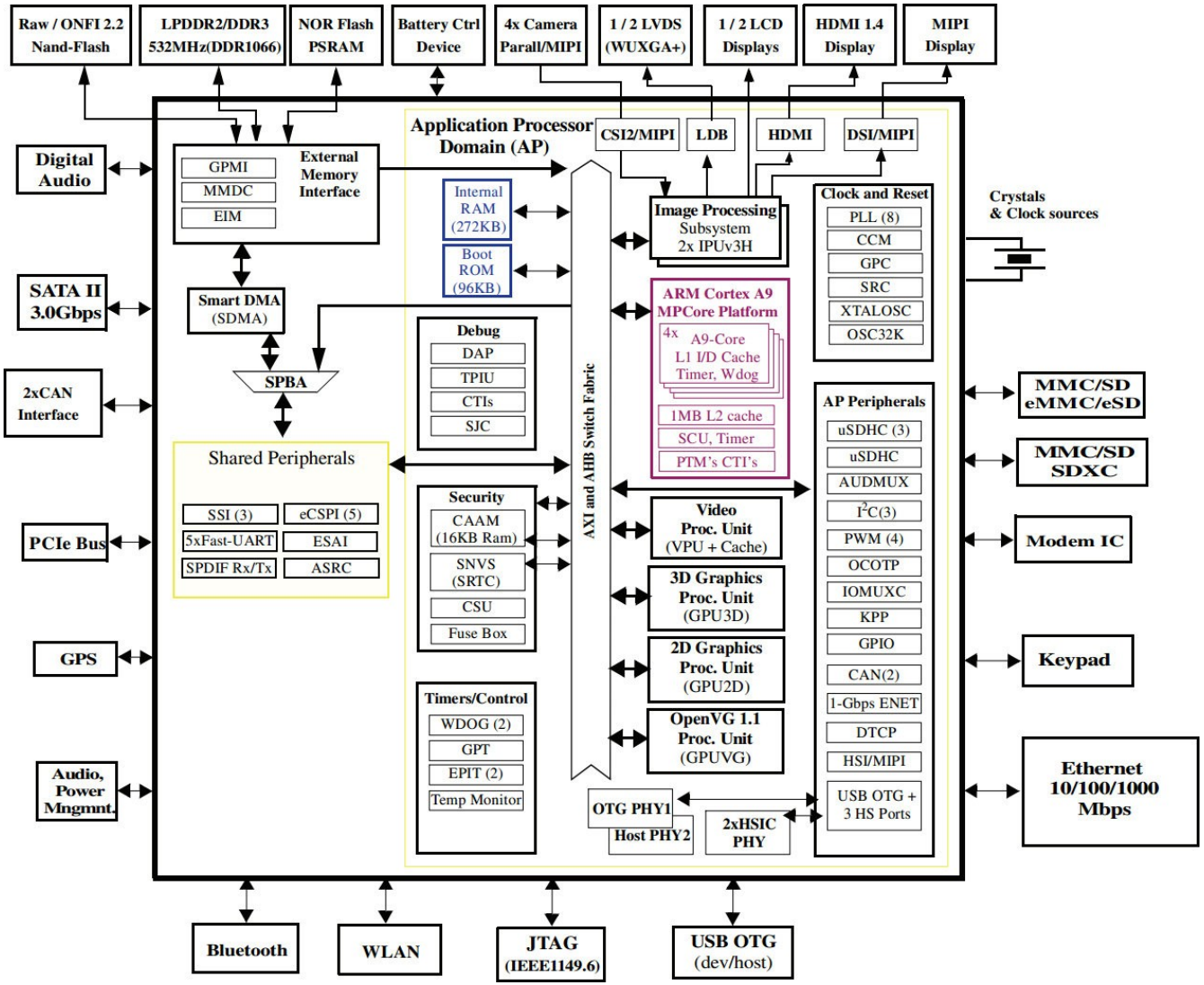








SoC Example (i.MX6)



Main Software Components of an Embedded Linux System

- **Bootloader -** initialization of RAM, clocks, basic peripherals

- **Kernel** - manages the hardware resources, such as memory space, interrupts, hardware peripheral access

- **Root File System** - provides the userspace applications and libraries. Example: busybox, Gstreamer, QT, Webkit

What is the role of the Bootloader?

- Bootloader’s main task is to **load the kernel into RAM** and pass the control to it

- First “external” software that runs on the system

- Perform the initial configuration of the system , initialize the memory controller, system clocks (CPU, peripherals), basic peripherals like serial ports, MMC controller and copy the kernel into RAM

- Provide some diagnostic utilities (memory dump, PMIC registers, direct register access, network stack)

Most common bootloaders

- U-boot : http://www.denx.de/wiki/U-Boot
- Barebox : http://barebox.org/

Getting U-boot source code

- U-boot source code is available at: git://git.denx.de/u-boot.git

- ***git clone git://git.denx.de/u-boot.git***
- ***git checkout -b 18.07 v2018.07***

At the time of writing this presentation the most recent version of U-boot is 2018.07

Porting U-Boot to a custom board

- An example of a U-Boot commit adding a new board support:

```
commit 55a42b33f2e9b9f6330396fc6d89878a5deacc75
Author: Peng Fan <van.freenix@gmail.com>
Date:   Thu Aug 11 14:02:57 2016 +0800

    arm: imx: add i.MX6ULL 14x14 EVK board support

    Add i.MX6ULL EVK board support:
    Add device tree file, which is copied from NXP Linux.
    Enabled DM_MMC, DM_GPIO, DM_I2C, DM_SPI, PINCTRL, DM_REGULATOR.
    The uart iomux settings are still keeped in board file.

    Boot Log:
    U-Boot 2016.09-rc1-00366-gbb419ef-dirty (Aug 11 2016 - 13:08:58 +0800)

    CPU:   Freescale i.MX6ULL rev1.0 at 396MHz
    CPU:   Commercial temperature grade (0C to 95C) at 15C
    Reset cause: POR
    Model: Freescale i.MX6 ULL 14x14 EVK Board
    Board: MX6ULL 14x14 EVK
    DRAM:  512 MiB
    MMC:   initialized IMX pinctrl driver
    FSL_SDHC: 0, FSL_SDHC: 1
    In:    serial
    Out:   serial
    Err:   serial
    Net:   CPU Net Initialization Failed
    No ethernet found.
    Hit any key to stop autoboot:  0
    => mmc dev 1
    switch to partitions #0, OK
    mmc1 is current device

    Signed-off-by: Peng Fan <peng.fan@nxp.com>
    Cc: Stefano Babic <sbabic@denx.de>
```

Adding Kconfig entry

```
diff --git a/arch/arm/cpu/armv7/mx6/Kconfig b/arch/arm/cpu/armv7/mx6/Kconfig
index 32405c6..d851b26 100644
--- a/arch/arm/cpu/armv7/mx6/Kconfig
+++ b/arch/arm/cpu/armv7/mx6/Kconfig
@@ -137,6 +137,12 @@ config TARGET_MX6UL_14X14_EVK
        select DM_THERMAL
        select SUPPORT_SPL

+config TARGET_MX6ULL_14X14_EVK
+       bool "Support mx6ull_14x14_evk"
+       select MX6ULL
+       select DM
+       select DM_THERMAL
+
 config TARGET_NITROGEN6X
        bool "nitrogen6x"

@@ -226,6 +232,7 @@ source "board/freescale/mx6slevk/Kconfig"
 source "board/freescale/mx6sxsabresd/Kconfig"
 source "board/freescale/mx6sxsabreauto/Kconfig"
 source "board/freescale/mx6ul_14x14_evk/Kconfig"
+source "board/freescale/mx6ullevk/Kconfig"
 source "board/phytec/pcm058/Kconfig"
 source "board/gateworks/gw_ventana/Kconfig"
 source "board/kosagi/novena/Kconfig"
diff --git a/arch/arm/dts/Makefile b/arch/arm/dts/Makefile
index 032c5ae..19140b4 100644
--- a/arch/arm/dts/Makefile
+++ b/arch/arm/dts/Makefile
@@ -280,6 +280,8 @@ dtb-$(CONFIG_VF610) += vf500-colibri.dtb \
        vf610-twr.dtb \
        pcm052.dtb

+dtb-$(CONFIG_MX6) += imx6ull-14x14-evk.dtb
+
 dtb-$(CONFIG_SOC_KEYSTONE) += k2hk-evm.dtb \
        k2l-evm.dtb \
```

Hardware description via device tree

```
--- /dev/null
+++ b/arch/arm/dts/imx6ull-14x14-evk.dts
@@ -0,0 +1,527 @@
+/*
+ * Copyright (C) 2016 Freescale Semiconductor, Inc.
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+/dts-v1/;
+
+#include <dt-bindings/input/input.h>
+#include "imx6ull.dtsi"
+
+/ {
+       model = "Freescale i.MX6 ULL 14x14 EVK Board";
+       compatible = "fsl,imx6ull-14x14-evk", "fsl,imx6ull";
+
+       chosen {
+               stdout-path = &uart1;
+       };
+
+       memory {
+               reg = <0x80000000 0x20000000>;
+       };
+
+       backlight {
+               compatible = "pwm-backlight";
+               pwms = <&pwm1 0 5000000>;
+               brightness-levels = <0 4 8 16 32 64 128 255>;
+               default-brightness-level = <6>;
+               status = "okay";
+       };
```

Hardware description via device tree

```
+&fec1 {
+       pinctrl-names = "default";
+       pinctrl-0 = <&pinctrl_enet1>;
+       phy-mode = "rmii";
+       phy-handle = <&ethphy0>;
+       status = "okay";
+};
+
+&fec2 {
+       pinctrl-names = "default";
+       pinctrl-0 = <&pinctrl_enet2>;
+       phy-mode = "rmii";
+       phy-handle = <&ethphy1>;
+       status = "okay";
+
+       mdio {
+               #address-cells = <1>;
+               #size-cells = <0>;
+
+               ethphy0: ethernet-phy@2 {
+                       compatible = "ethernet-phy-ieee802.3-c22";
+                       reg = <2>;
+               };
+
+               ethphy1: ethernet-phy@1 {
+                       compatible = "ethernet-phy-ieee802.3-c22";
+                       reg = <1>;
+               };
+       };
+};
```

Board level Kconfig and Maintainers

```
diff --git a/board/freescale/mx6ullevk/Kconfig b/board/freescale/mx6ullevk/Kconfig
new file mode 100644
index 0000000..7eec497
--- /dev/null
+++ b/board/freescale/mx6ullevk/Kconfig
@@ -0,0 +1,12 @@
+if TARGET_MX6ULL_14X14_EVK
+
+config SYS_BOARD
+       default "mx6ullevk"
+
+config SYS_VENDOR
+       default "freescale"
+
+config SYS_CONFIG_NAME
+       default "mx6ullevk"
+
+endif
diff --git a/board/freescale/mx6ullevk/MAINTAINERS b/board/freescale/mx6ullevk/MAINTAINERS
new file mode 100644
index 0000000..4137674
--- /dev/null
+++ b/board/freescale/mx6ullevk/MAINTAINERS
@@ -0,0 +1,6 @@
+MX6ULLEVK BOARD
+M:     Peng Fan <peng.fan@nxp.com>
+S:     Maintained
+F:     board/freescale/mx6ullevk/
+F:     include/configs/mx6ullevk.h
+F:     configs/mx6ull_14x14_evk_defconfig
```

RAM initialization

- board/freescale/mx6ullevk/imximage.cfg
- Key element for system **stability**
- Incorrect RAM initialization may lead to random kernel crashes
- Recommended to run memory stress test tools such as ‘memtester’

Defconfig file

```
--- /dev/null
+++ b/configs/mx6ull_14x14_evk_defconfig
@@ -0,0 +1,30 @@
+CONFIG_ARM=y
+CONFIG_ARCH_MX6=y
+CONFIG_TARGET_MX6ULL_14X14_EVK=y
+CONFIG_DM_GPIO=y
+CONFIG_DEFAULT_DEVICE_TREE="imx6ull-14x14-evk"
+CONFIG_SYS_EXTRA_OPTIONS="IMX_CONFIG=board/freescale/mx6ullevk/imximage.cfg"
+CONFIG_BOOTDELAY=3
+CONFIG_HUSH_PARSER=y
+CONFIG_CMD_BOOTZ=y
+# CONFIG_CMD_IMLS is not set
+CONFIG_CMD_MEMTEST=y
+CONFIG_CMD_MMC=y
+CONFIG_CMD_I2C=y
+CONFIG_CMD_GPIO=y
+CONFIG_CMD_DHCP=y
+CONFIG_CMD_PING=y
+CONFIG_CMD_CACHE=y
+CONFIG_CMD_EXT2=y
+CONFIG_CMD_EXT4=y
+CONFIG_CMD_EXT4_WRITE=y
+CONFIG_CMD_FAT=y
+CONFIG_CMD_FS_GENERIC=y
+CONFIG_OF_CONTROL=y
+CONFIG_DM_74X164=y
+CONFIG_DM_I2C=y
+CONFIG_DM_MMC=y
+CONFIG_PINCTRL=y
+CONFIG_PINCTRL_IMX6=y
+CONFIG_DM_REGULATOR=y
+CONFIG_DM_SPI=y
```

Board .h file

- include/configs/mx6ullevk.h
- Passes the kernel command line, for example:

console=ttymxc0,115200 root=/dev/nfs ip=dhcp nfsroot=192.168.1.100:/tftpboot/nfs

- dtb script logic selection
- Distro config option (doc/README.distro)

```
“ This document defines a common set of U-Boot features that
are required for a distro to support the board in a generic
fashion.”
```

Building and flashing U-boot

- Export the cross toolchain

export ARCH=arm
export CROSS_COMPILE=/usr/bin/arm-linux-gnueabi

- ***make mx6ull_14x14_evk_defconfig***

- ***make -j4***

- Copy resultant U-boot binary into the SD card

sudo dd if=u-boot-dtb.imx of=/dev/mmcblk0 bs=1k seek=1; sync

Hopefully a serial console will show up :-)

- U-Boot messages should appear in the serial console

- If not, time to further debug it and maybe also consider the usage of hardware debug tools (JTAG)

- OpenOCD project is a nice open source JTAG software (***http://openocd.org/***)

SPL and Falcon Mode

- On a embedded system it is often desired to have quick “turn-on” response in the LCD

- Providing **splash screen** support allows for a quick visual feedback to the user

- U-boot **Falcon** mode: mechanism that allows to boot a Linux kernel (or whatever image) without the need of a full blown U-Boot (doc/README.falcon)

Errata handling

- Many ARM errata can only be applied in the bootloader. Example: for the Spectre v2 workaround on ARM Cortex-A8

```
From 363173d7f241ac09c7c0f3b14b25de78a8a0ff26 Mon Sep 17 00:00:00 2001
From: Fabio Estevam <festevam@gmail.com>
Date: Thu, 12 Jul 2018 14:45:03 -0300
Subject: mx5: Implement Spectre v2 workaround for Cortex-A8

Since 4.18-rc1 kernel the following warning is seen on i.MX51 and
i.MX53:

CPU0: Spectre v2: firmware did not set auxiliary control register IBE bit, system vulnerable

Implement the suggested workaround by setting the IBE bit in the
auxiliary control register, which allows the kernel to flush the
BTB properly.

Based on commit 7b37a9c732bf ("ARM: Introduce ability to enable ACR::IBE
on Cortex-A8 for CVE-2017-5715") from U-Boot.

With this patch applied the kernel now reports:

CPU0: Spectre v2: using BPIALL workaround

Tested on a imx51 babbage.

Signed-off-by: Fabio Estevam <festevam@gmail.com>
Signed-off-by: Sascha Hauer <s.hauer@pengutronix.de>
```

Getting the kernel source code

- From silicon vendors: Board Support Packages (BSP)
- From mainline kernel - **recommended**

- ***git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git***
- ***git checkout -b 4.18 origin/linux-4.18.y***

Building a kernel

- Kernel offers pre-configured configurations inside **arch/<architecture>/configs/**
- For example: building for i.MX ARMv6/ARMv7 chips
- ***make imx_v6_v7_defconfig***
- ***make -j4***
- Kernel binary is generated at arch/arm/boot/zImage

Porting Linux to a custom board

- With the adoption of device tree the process of adding support of a new board to the kernel becomes much easier

- Need to create a device tree file that represents the hardware

- If the SoC and the board hardware have already support in the kernel then that’s all :-)

Device Tree Example

```
// SPDX-License-Identifier: GPL-2.0
//
//Copyright (C) 2013 Freescale Semiconductor, Inc.

/dts-v1/;

#include <dt-bindings/gpio/gpio.h>
#include <dt-bindings/input/input.h>
#include "imx6sl.dtsi"

/ {
        model = "Freescale i.MX6 SoloLite EVK Board";
        compatible = "fsl,imx6sl-evk", "fsl,imx6sl";

        chosen {
                stdout-path = &uart1;
        };

        memory@80000000 {
                reg = <0x80000000 0x40000000>;
        };

        backlight_display: backlight_display {
                compatible = "pwm-backlight";
                pwms = <&pwm1 0 5000000>;
                brightness-levels = <0 4 8 16 32 64 128 255>;
                default-brightness-level = <6>;
        };

        leds {
                compatible = "gpio-leds";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_led>;

                user {
                        label = "debug";
                        gpios = <&gpio3 20 GPIO_ACTIVE_HIGH>;
                        linux,default-trigger = "heartbeat";
                };
        };

        reg_usb_otg1_vbus: regulator-usb-otg1-vbus {
                compatible = "regulator-fixed";
                regulator-name = "usb_otg1_vbus";
                regulator-min-microvolt = <5000000>;
                regulator-max-microvolt = <5000000>;
                gpio = <&gpio4 0 GPIO_ACTIVE_HIGH>;
                enable-active-high;
                vin-supply = <&swbst_reg>;
        };
```

Device Tree Example

```
&i2c2 {
        clock-frequency = <100000>;
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_i2c2>;
        status = "okay";

        codec: wm8962@1a {
                compatible = "wlf,wm8962";
                reg = <0x1a>;
                clocks = <&clks IMX6SL_CLK_EXTERN_AUDIO>;
                DCVDD-supply = <&vgen3_reg>;
                DBVDD-supply = <&reg_aud3v>;
                AVDD-supply = <&vgen3_reg>;
                CPVDD-supply = <&vgen3_reg>;
                MICVDD-supply = <&reg_aud3v>;
                PLLVDD-supply = <&vgen3_reg>;
                SPKVDD1-supply = <&reg_aud4v>;
                SPKVDD2-supply = <&reg_aud4v>;
        };
};

&ssi2 {
        status = "okay";
};

&uart1 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_uart1>;
        status = "okay";
};

&usbotg1 {
        vbus-supply = <&reg_usb_otg1_vbus>;
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_usbotg1>;
        disable-over-current;
        status = "okay";
};

&usbotg2 {
        vbus-supply = <&reg_usb_otg2_vbus>;
        dr_mode = "host";
        disable-over-current;
        status = "okay";
};

&usdhc1 {
        pinctrl-names = "default", "state_100mhz", "state_200mhz";
        pinctrl-0 = <&pinctrl_usdhc1>;
        pinctrl-1 = <&pinctrl_usdhc1_100mhz>;
        pinctrl-2 = <&pinctrl_usdhc1_200mhz>;
        bus-width = <8>;
        cd-gpios = <&gpio4 7 GPIO_ACTIVE_LOW>;
        wp-gpios = <&gpio4 6 GPIO_ACTIVE_HIGH>;
        status = "okay";
};
```

Booting a device tree machine

- Let’s use i.MX6DL Wandboard as an example
- Build the dtb file: ***make imx6dl-wandboard.dtb***
- Build the kernel: ***make imx_v6_v7_defconfig; make***
- U-boot scripts:

  => tftp 0x12000000 zImage

  => tftp 0x11000000 imx6dl-wandboard.dtb

  => bootm 0x12000000 - 0x11000000

Concept of root file system

- Root file system provides all the userspace applications and libraries (GUI like QT, media player, busybox, browser, etc)

- Typical root file system layout:

root@freescale /$ ls
bin dev home mnt proc sbin tmp var
boot etc lib opt root sys usr

Build systems

- Yocto: http://www.yoctoproject.org
- Buildroot: https://buildroot.org/

Using Buildroot to build the entire image

- https://buildroot.org/

- Very simple tool to generate a complete bootable image

git clone git://git.buildroot.net/buildroot

- Adding support for a new target is very simple. Needs to add a defconfig file

Buildroot defconfig example

```
BR2_arm=y
BR2_cortex_a9=y
BR2_ARM_ENABLE_NEON=y
BR2_ARM_ENABLE_VFP=y
BR2_ARM_FPU_VFPV3=y

# Linux headers same as kernel, a 4.17 series
BR2_PACKAGE_HOST_LINUX_HEADERS_CUSTOM_4_17=y

# System
BR2_TARGET_GENERIC_GETTY_PORT="ttymxc0"

# required tools to create the SD card image
BR2_PACKAGE_HOST_DOSFSTOOLS=y
BR2_PACKAGE_HOST_GENIMAGE=y
BR2_PACKAGE_HOST_MTOOLS=y

# Filesystem
BR2_ROOTFS_POST_IMAGE_SCRIPT="board/freescale/common/imx/post-image.sh"
BR2_TARGET_ROOTFS_EXT2=y
BR2_TARGET_ROOTFS_EXT2_4=y

# Bootloader
BR2_TARGET_UBOOT=y
BR2_TARGET_UBOOT_BOARDNAME="mx6sabresd"
BR2_TARGET_UBOOT_CUSTOM_VERSION=y
BR2_TARGET_UBOOT_CUSTOM_VERSION_VALUE="2018.05"
BR2_TARGET_UBOOT_FORMAT_IMG=y
BR2_TARGET_UBOOT_SPL=y
BR2_TARGET_UBOOT_SPL_NAME="SPL"

# Kernel
BR2_LINUX_KERNEL=y
BR2_LINUX_KERNEL_CUSTOM_VERSION=y
BR2_LINUX_KERNEL_CUSTOM_VERSION_VALUE="4.17.4"
BR2_LINUX_KERNEL_DEFCONFIG="imx_v6_v7"
BR2_LINUX_KERNEL_DTS_SUPPORT=y
BR2_LINUX_KERNEL_INTREE_DTS_NAME="imx6q-sabresd imx6dl-sabresd imx6qp-sabresd"
BR2_LINUX_KERNEL_NEEDS_HOST_OPENSSL=y
```

Building a Buildroot image

- Select a defconfig
  make imx6-sabresd_defconfig

- Build it
  make

- Image is created at output/images/sdcard.img

- Flash it into the SD card

sudo dd if=output/images/sdcard.img of=/dev/mmcblk0; sync

The importance of upstreaming

- Electronic components get obsoleted. Replacement chips may not be supported in older kernels
- Bug fixes and security fixes
- Long time support maintenance
- You can improve your Linux skills and have some fun ;-)