barebox, the bootloader for Linux kernel developers

Marco Felsch – m.felsch@pengutronix.de

About Me

- Embedded software engineer at **Pengutronix**
  - Kernel, bootloader, graphic development
  - PTXdist/Yocto integration
- Open-Source contributor
- Living in Vechta, Germany
- marco.felsch@pengutronix.de

Agenda

- Brief project introduction
- Add support for a new driver
- Add support for a new board
- Hands on

Welcome to barebox

- Started in 2007 as U-Boot-v2 patchset
- Renamed to barebox in 2009

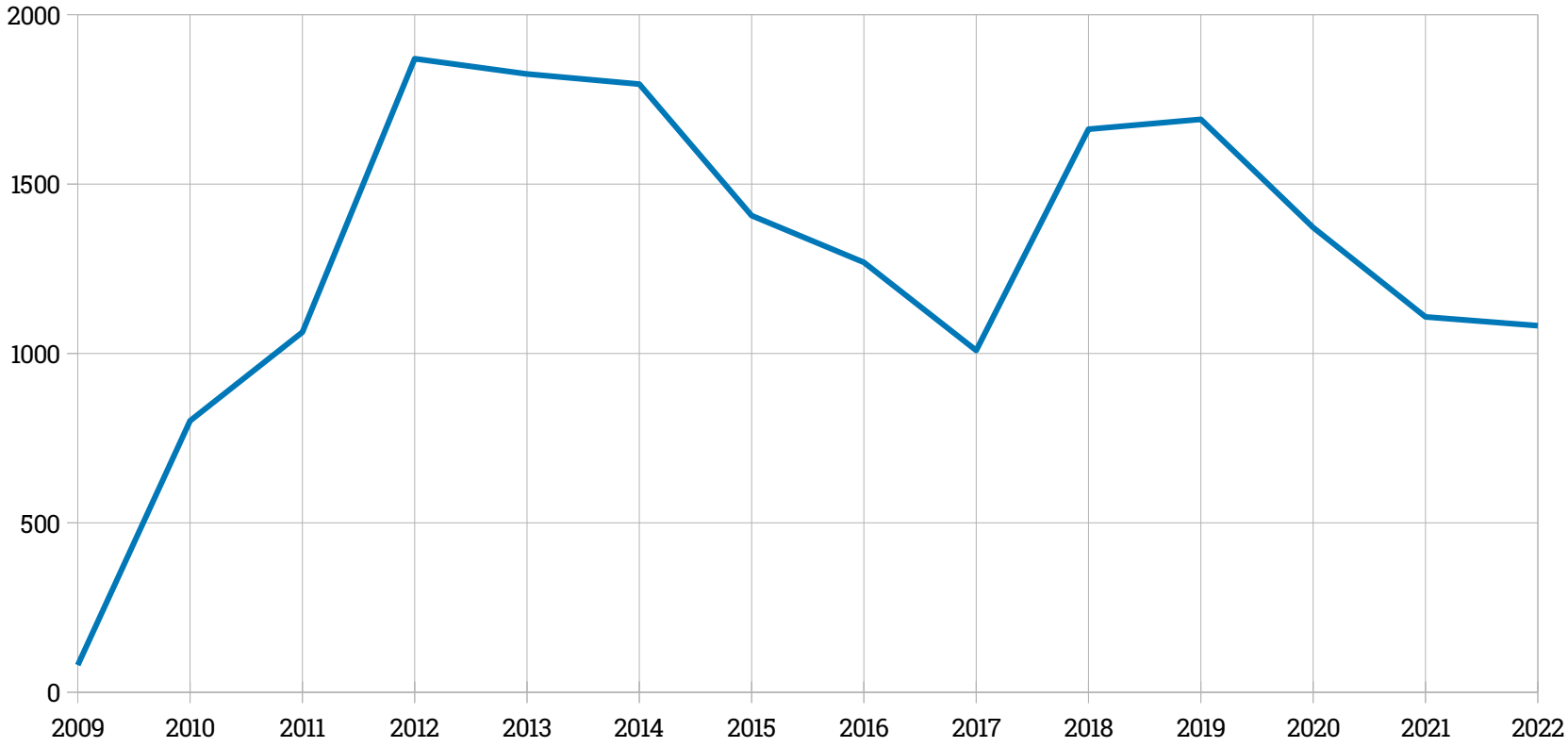
```
commit a3ffa97f40dc81f2d6b07ee964f2340fe0c1ba97
Author: Sascha Hauer <s.hauer@pengutronix.de>
Date:   Tue Dec 15 09:11:09 2009 +0100

    rename U-Boot-v2 project to barebox

    This has been done with the following script:
```

Welcome to barebox

- Started in 2007 as U-Boot-v2 patchset
- Renamed to barebox in 2009
- Monthly releases
- Mainline Buildroot and PTXdist support
- Yocto/OE-Core support through meta-ptx/meta-barebox (mainline coming soon*)
- ~330 Contributors

*https://lore.kernel.org/openembedded-core/20230203135011.2061939-1-m.felsch@pengutronix.de/

Welcome to barebox (insights)



~ 1400 commits per year

Add support for a new driver

Design decisions

- Use a stripped down Linux device/driver model
- Runtime configuration is done via Devicetree and/or Kconfig options
- Reuse stripped down Linux driver frameworks

Add support for a new driver

Design decisions

- Use a stripped down Linux device/driver model
- Runtime configuration is done via Devicetree and/or Kconfig options
- Reuse stripped down Linux driver frameworks

Let’s add a driver!

Add support for a new driver (clk-rk3188)

- Copy the driver from Linux to barebox

Add support for a new driver (clk-rk3188)

- Copy the driver from Linux to barebox
- Adapt the code to barebox

Add support for a new driver (clk-rk3188)

```
--- a/../linux/drivers/clk/rockchip/clk-rk3188.c
+++ b/drivers/clk/rockchip/clk-rk3188.c
@@ -4,13 +4,14 @@
  * Author: Heiko Stuebner <heiko@sntech.de>
  */

+#include <common.h>
 #include <linux/clk.h>
-#include <linux/clk-provider.h>
-#include <linux/io.h>
-#include <linux/of.h>
-#include <linux/of_address.h>
+#include <of.h>
+#include <of_address.h>
 #include <dt-bindings/clock/rk3188-cru-common.h>
 #include "clk.h"
+#include <linux/barebox-wrapper.h>
+#include <init.h>

 #define RK3066_GRF_SOC_STATUS  0x15c
 #define RK3188_GRF_SOC_STATUS  0xac
```

Add support for a new driver (clk-rk3188)

```
--- a/../linux/drivers/clk/rockchip/clk-rk3188.c
+++ b/drivers/clk/rockchip/clk-rk3188.c
@@ -771,18 +772,12 @@ static struct rockchip_clk_provider *__init rk3188_common_clk_init(struct device
        ctx = rockchip_clk_init(np, reg_base, CLK_NR_CLKS);
        if (IS_ERR(ctx)) {
                pr_err("%s: rockchip clk init failed\n", __func__);
-               iounmap(reg_base);
                return ERR_PTR(-ENOMEM);
        }

        rockchip_clk_register_branches(ctx, common_clk_branches,
                                       ARRAY_SIZE(common_clk_branches));

-       rockchip_register_softrst(np, 9, reg_base + RK2928_SOFTRST_CON(0),
-                                 ROCKCHIP_SOFTRST_HIWORD_MASK);
-
-       rockchip_register_restart_notifier(ctx, RK2928_GLB_SRST_FST, NULL);
-
        return ctx;
 }
```

Add support for a new driver (clk-3188)

- Copy the driver from Linux to barebox
- Adapt the code to barebox
  - 15 LOC changes by a driver size of 871 LOC = ~1.72% adapted code
  - Other drivers may need more adaptions, e.g. replace IRQ by polling

Add support for a new driver (clk-3188)

- Copy the driver from Linux to barebox
- Adapt the code to barebox
  - 15 LOC changes by a driver size of 871 LOC = ~1.72% adapted code
  - Other drivers may need more adaptions, e.g. replace IRQ by polling
- Add the Kconfig and Makefile entries

Add support for a new driver (clk-3188)

```
config ARCH_RK3188
        bool
        select ARCH_ROCKCHIP_V7
```

```
# SPDX-License-Identifier: GPL-2.0-only
obj-y += clk-cpu.o clk-pll.o clk.o clk-muxgrf.o clk-mmc-phase.o clk-inverter.o
obj-$(CONFIG_RESET_CONTROLLER) += softrst.o
obj-$(CONFIG_ARCH_RK3188) += clk-rk3188.o
obj-$(CONFIG_ARCH_RK3288) += clk-rk3288.o
obj-$(CONFIG_ARCH_RK3399) += clk-rk3399.o
obj-$(CONFIG_ARCH_RK3568) += clk-rk3568.o
```

Add support for a new driver (clk-3188)

- Copy the driver from Linux to barebox
- Adapt the code to barebox
  - 15 LOC changes by a driver size of 871 LOC = ~1.72% adapted code
  - Other drivers may need more adaptions, e.g. replace IRQ by polling
- Add the Kconfig and Makefile entries
- Compile & Test by using the Linux Devicetrees

Add support for a new driver (clk-3188)

- Copy the driver from Linux to barebox
- Adapt the code to barebox
  - 15 LOC changes by a driver size of 871 LOC = ~1.72% adapted code
  - Other drivers may need more adaptions, e.g. replace IRQ by polling
- Add the Kconfig and Makefile entries
- Compile & Test by using the Linux Devicetrees

Feels like writing a kernel driver right?

Add support for a new driver

Summary

- A barebox driver is just a stripped down Linux driver
- New drivers can be ported with little effort
- Already supported driver frameworks
  - fpga, pci, net-dsa, sound, gpio, i2c, usb, nvmem, ...
- Porting frameworks is more effort (depending on framwork complexity)

Add support for a new board

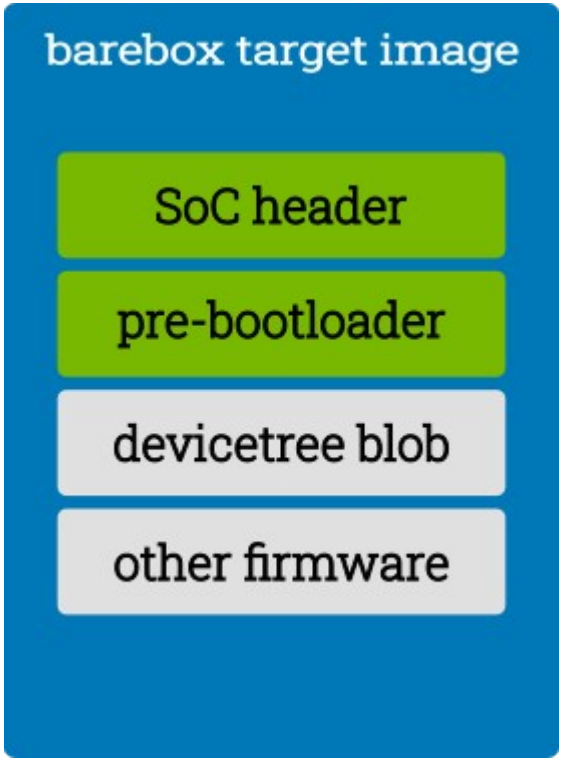
Simplified Image Layout

Add support for a new board

Simplified Image Layout

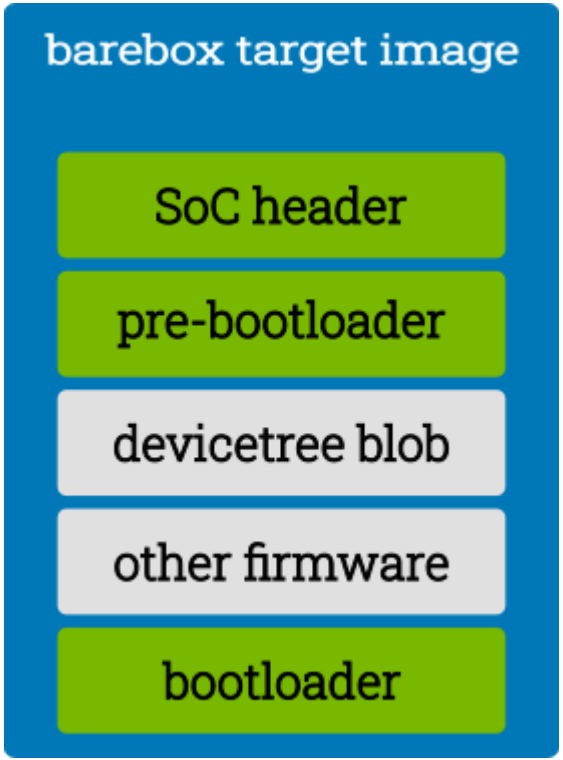
Add support for a new board

Simplified Image Layout



Add support for a new board

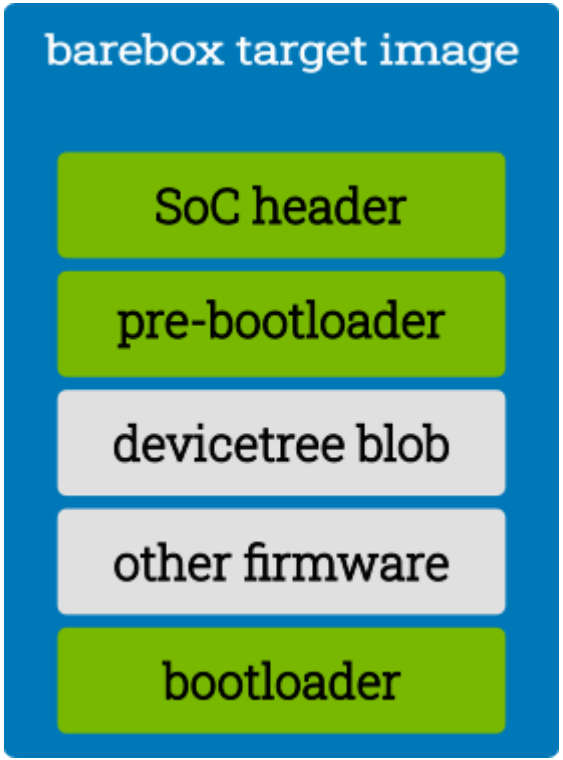
Simplified Image Layout



Add support for a new board

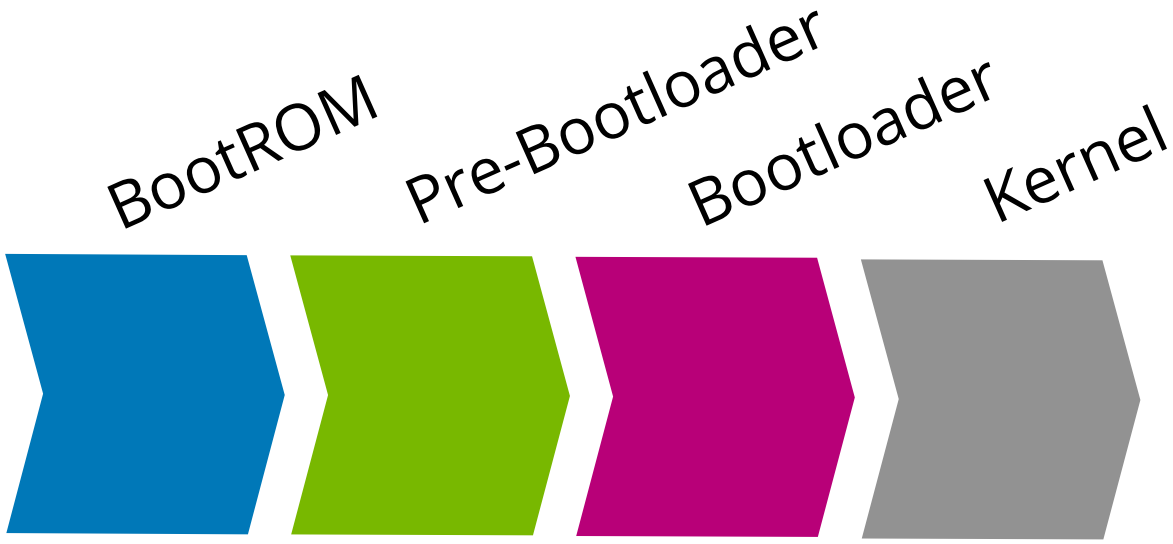
Simplified Image Layout

- Singleton target image but many artifacts
- Artifacts are linked together
  - Depending on build-target: pbl, bootloader
  - Depending on architecture linker scripts
- SoC header created and added by specific image tool



Add support for a new board

Simplified boot flow



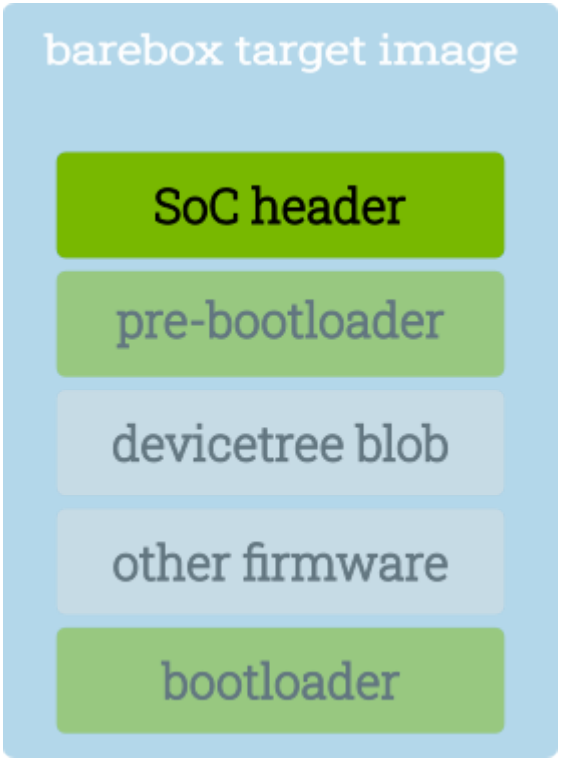
Add support for a new board

Let’s add a new board!

Add support for a new board

BootROM

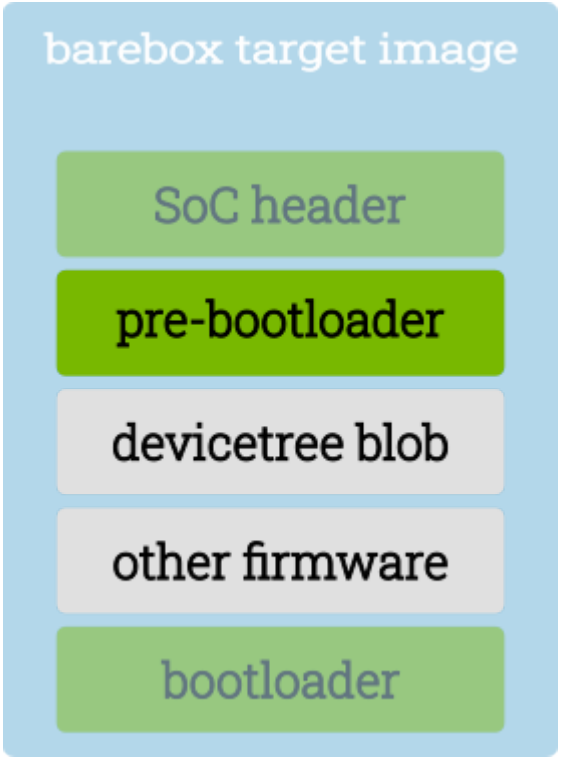
- Intialise the SoC
- Load & execute software
- Very SoC specific



Add support for a new board

Pre-Bootloader (PBL)

- Loaded by the BootROM
- Setup the DRAM
- Load the actual bootloader from the persistent memory into the RAM and execute it
- Strict size limitations: no DeviceTree nor Device/Driver model, like TBL



Add support for a new board (lowlevel)

Pre-Bootloader (PBL)

```
ENTRY_FUNCTION(start_nxp_imx8mn_evk, r0, r1, r2)
{
    imx8mn_cpu_lowlevel_init();

    relocate_to_current_adr();
    setup_c();

    nxp_imx8mn_evk_start();
}
```

Add support for a new board (lowlevel)

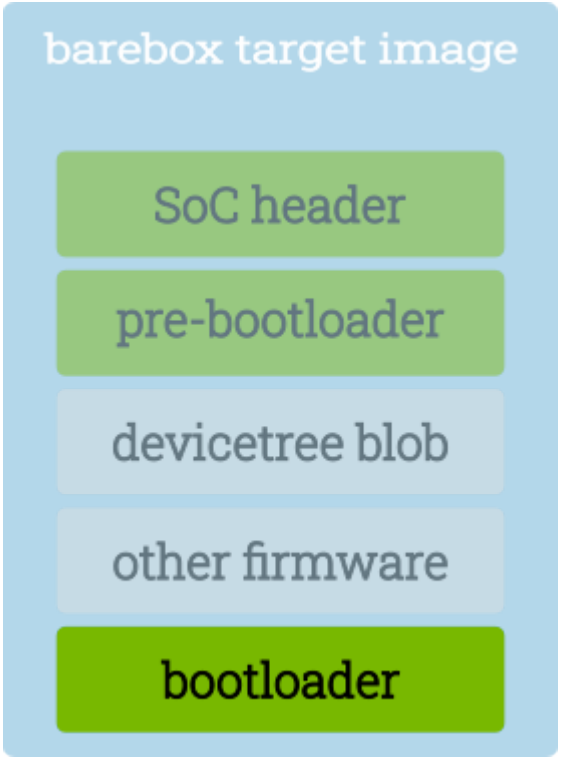
Pre-Bootloader (PBL)

```
static __noreturn noinline void nxp_imx8mn_evk_start(void)
{
    ...
    setup_uart();

    start_atf(); /* Setup DDR, load barebox, start TF-A */
    ...
    imx8mn_barebox_entry(fdt); /* start barebox */
}
```

Add support for a new board

Bootloader

- Board specific boot decisions
- Board specific fixups (e.g. apply Devicetree-Overlays)
- Load & execute the kernel
- Board code is just a driver



Add support for a new board (kernel style)

Bootloader

```
static const struct of_device_id imx8mn_evk_of_match[] = {
    { .compatible = "fsl,imx8mn-evk" },
    { .compatible = "fsl,imx8mn-ddr4-evk" },
    { /* sentinel */ },
};

static struct driver imx8mn_evkboard_driver = {
    .name = "board-imx8mn-evk",
    .probe = imx8mn_evk_probe,
    .of_compatible = DRV_OF_COMPAT(imx8mn_evk_of_match),
};
coredevice_platform_driver(imx8mn_evkboard_driver);
```

Add support for a new board (kernel style)

Bootloader

```
static int imx8mn_evk_probe(struct device *dev)
{
    if (bootsource_get() == BOOTSOURCE_MMC) {
        ...
    }
    imx8m_bbu_internal_mmc_register_handler(...);
    imx8m_bbu_internal_mmcboot_register_handler(...);

    phy_register_fixup_for_uid(...);

    return 0;
}
```

Add support for a new board

Kernel

- Do all the remaining cool stuff

Hands on

- Easy to start with via tinyEMU online RISC-V emulator running barebox:

  https://www.barebox.org/jsbarebox

- Feature rich shell e.g. color, **auto-completion**, history, **scripting**, ...
- Virtual filesystem support (VFS). Forget about magic commands and offsets, just use **cp, ls, rm** or **auto-/mount**
- Memory-mapped IO acces via **md** and **mw**

Hands on

- Updating the bootloader is just one command away using **barebox_update**
- Multi-Image support to **compile all** required images **in one go**, no more wasting compile time

Thank you very much

Question?