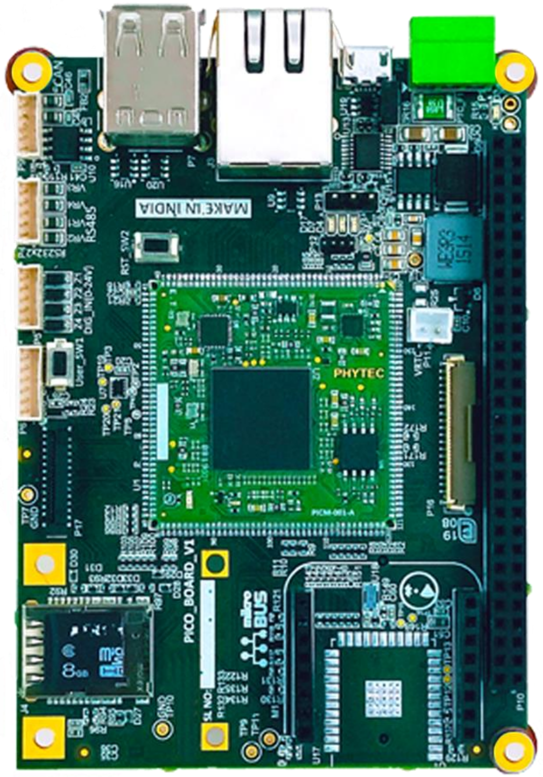
Embedded Linux Porting (C9)

Linux Kernel-2

Agenda

- Linux Kernel Code Flow
- How to modify the Kernel code
- How to integrate new driver / module in kernel image
- Building static and dynamic kernel modules
- How to port Kernel on ARM Hardware

RuggedBOARD



- A5D2x @500MHz
Cortex - A5
64MB Ram
32MB Flash
- 2 x RS232
- 1x RS485
- 1 x CAN
- 1 x Ethernet
- TFT & Cap Touch
- 1 x MicroSD Slot
- 2 x USB
- DC & USB Power
- Expansion Header
- mikroBUS Conn.
- mPCIe conn.
- Micro Sim Slot

Industrial Grade Hardware for IIoT

https://Community.ruggedboard.com

Kernel Source

Browse Source: https://github.com/rugged-board/linux-rba5d2x.git

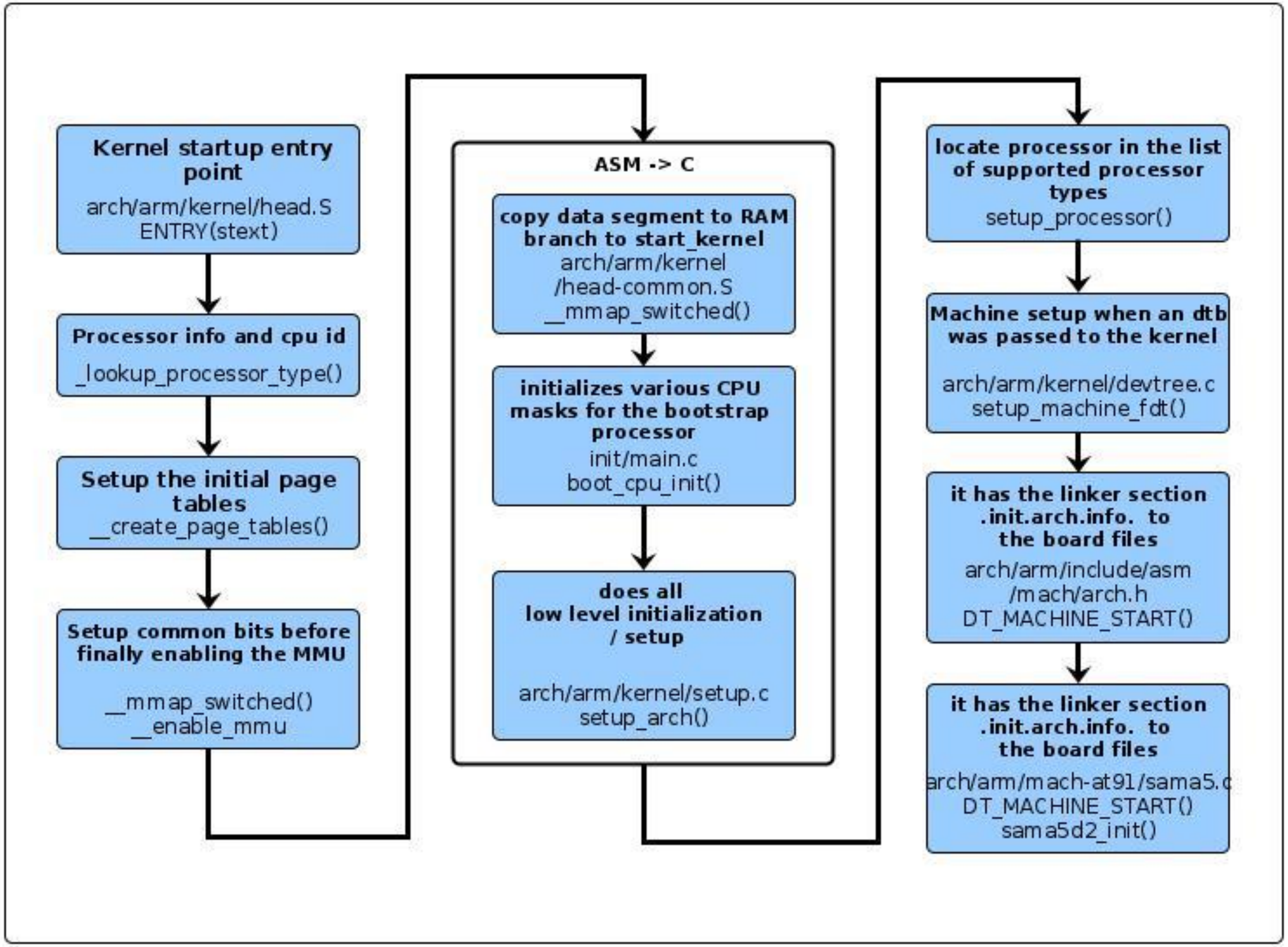
Download U-Boot for RuggedBOARD

```
$ wget https://github.com/rugged-board/linux-rba5d2x/archive/linux-rba5d2x.zip
```

Or

```
$ git clone https://github.com/rugged-board/linux-rba5d2x.git
```

Kernel Code Flow



Kernel startup entry point
arch/arm/kernel/head.S
ENTRY(stext)

Processor info and cpu id
_lookup_processor_type()

Setup the initial page tables
__create_page_tables()

Setup common bits before finally enabling the MMU
__mmap_switched()
__enable_mmu

ASM -> C

copy data segment to RAM branch to start_kernel
arch/arm/kernel/head-common.S
__mmap_switched()

initializes various CPU masks for the bootstrap processor
init/main.c
boot_cpu_init()

does all low level initialization / setup
arch/arm/kernel/setup.c
setup_arch()

locate processor in the list of supported processor types
setup_processor()

Machine setup when an dtb was passed to the kernel
arch/arm/kernel/devtree.c
setup_machine_fdt()

it has the linker section .init.arch.info. to the board files
arch/arm/include/asm/mach/arch.h
DT_MACHINE_START()

it has the linker section .init.arch.info. to the board files
arch/arm/mach-at91/sama5.c
DT_MACHINE_START()
sama5d2_init()

Adding new Driver

#Step-1: Define your device in dts file

```
$ vim linux/arch/arm/dts/rugged_board_a5d2x.dts

leds {
        compatible = "sled-testing";
        status = "okay";

        UserLed {
            label = "UserLed";
                          sled-default-state = "blink";
        };
    };
```

# Step-2: Define your driver sled.c in linux/driver/misc folder

```
$ vim linux/driver/misc/sled.c
# copy the sled.c code
```

# Step-3: Add sled configuration in Kconfig file

```
$ vim linux/driver/misc/Kconfig
```

```
config SLED
        tristate "SLED support for LEDs"
        depends on LED
        default y
        help
          Sled driver on RuggedBOARD-A5D2x
```

# Step-4: Add sled entry to Makefile

```
$ vim linux/driver/misc/Makefile
```

```
obj-$(CONFIG_SLED) += sled.o
```

# Step-5: Enable the SLED from Menuconfig or add it in board_dfconfig file under linux/arch/arm/configs

Kernel Modules Compilation

Kernel Modules can be compiled as

- Satic modules part of Linux Kernel Image
- Dynamic modules .ko files which should be copied to RFS

```
#In Makefile
obj-$(CONFIG_<MODULE>)  += module.o

#In menuconfig we can select as * or M which will result
obj-y                              // '*' option Static Module
obj-m for M                        // 'M' Dynamic Module
```

Compiling Linxu Kernel

Developer Wiki Page link here …

#1 Download the source

#2 Set the toolchain

#3 Clean the source only for the first time

```
$make distclean
```

#4 Configure the kernel source for ruggedboard-a5d2x

```
$make rb_a5d2x_defconfig
```

#5 Do additional configuration if required using menuconfig

```
$make menuconfig
```

#6 Compile the Kernel code

```
$make
```

Porting Linux Kernel to New Board

Adding ARM New Board:

1. Identify the ARCH, CORE & SOC used in your board

2. Check the ARCH & Core support in kernel location linux/arch/arm/kernel

3. Check the SOC support in kernel location linux/arch/arm/mach-or-plat-<soc_family>

4. Register your board using DT_MACHINE_START() macro if compatible board is not present in linux/arch/arm/mach-<soc_family>/<board_name>.c

5. Add the board dts file in linux/arch/arm/boot/dtb folder

6. Create a default configuration file for your board in linux/arch/arm/configs

7. Make sure you did modified Makfiles corresponding to your code/file changes.

8. Driver level modification if required. linux/drivers/

Kernel Experiments

1. Delete RootFS Image from u-boot and boot the board (make note of the error log)
2. Delete Kernel Image from u-boot and boot the board (make note of the error log)
3. Do Kernel banner modifications, compile & test the new kernel Image
4. Disable the Sound subsystem in Kernel
   a) Compare the Kernel image size with & without Sound Subsystem
   b) Compare the Kernel boot-log with & without Sound Subsystem.
5. Add sled driver under driver/misc folder and test the kernel

Open Discussions

Connect-us

in

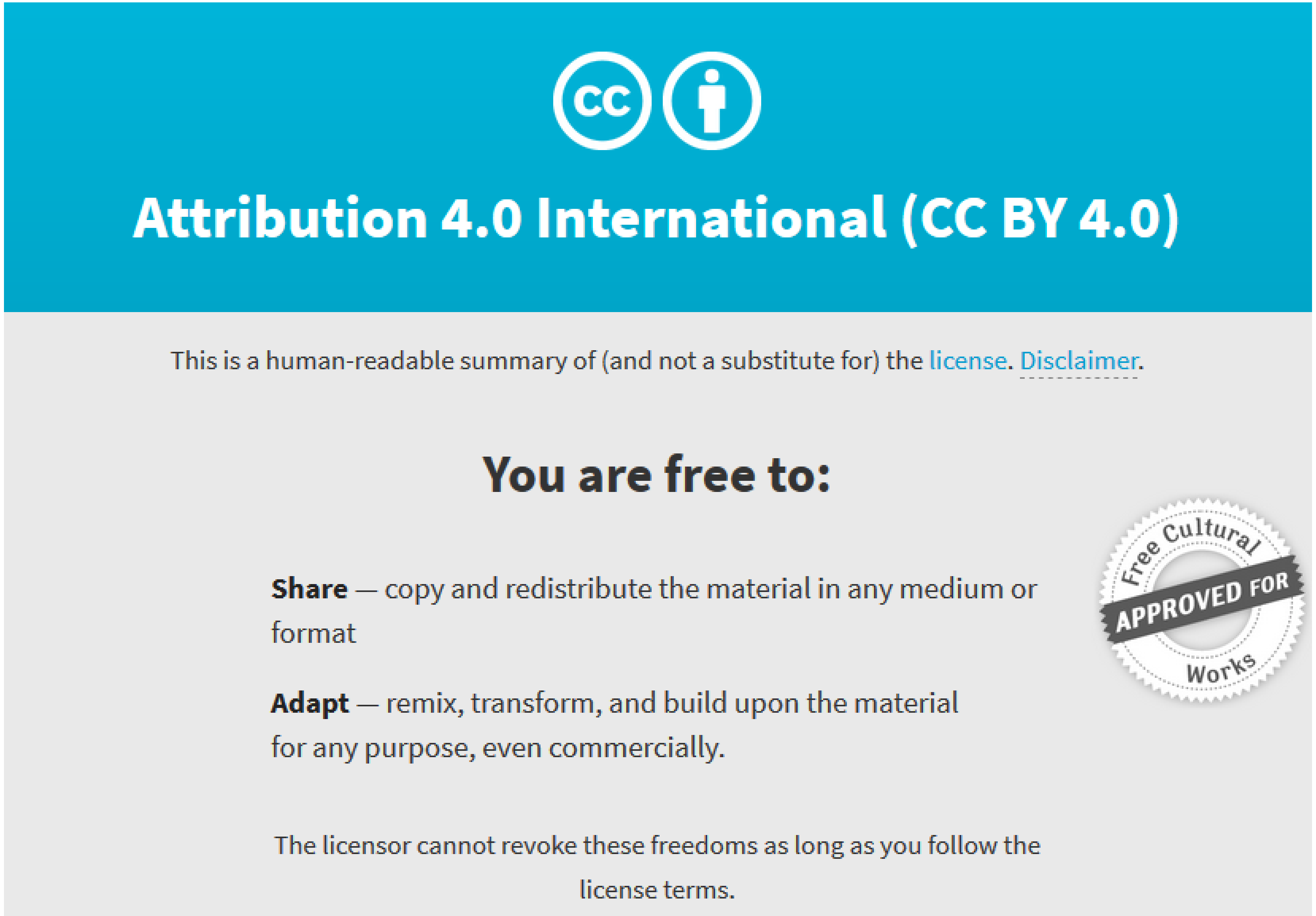
BLOG

You Tube

FORUM

Developer Wiki

Follow us on twitter



Attribution 4.0 International (CC BY 4.0)

This is a human-readable summary of (and not a substitute for) the license. Disclaimer.

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.