Linux ePorting

Topic: ePorting

Time: 2 Hours **Marks:**50

Part A & Part B all questions are mandatory.

Part-A 20*1=20

1.what are the Common Bootmodes:

ans:

1. Serial Bootmodes
2. SDCard Bootmodes
3. USB Bootmode (DFU)
4. Ethernet Bootmode

2.what are the Linux Device Driver Model.

Ans:

1. Platform Device
2. Platform Data
3. Platform Driver

3. what are the U-BOOT Environment Variables.

ipaddr Board IP Address

serverip Server IP Address

bootenv Kernel Command line Arguments

bootcmd Default command executed by u-boot to boot the system

4.what are the boot mode of soc.

Ans:

Boot Mode of SOC:

- Boot Config Pins
- Boot Config Registers

5.write a boot modes supported by RuggedBOARD.

Ans:

RuggedBOARD support boot from

1. NOR Flash
2. SDCARD

3. Serial Boot Mode for flashing image’s using SAM-BA PC Tool.

6.What is k-config?

Ans:

Kconfig is a configuration system used in the Linux kernel, offering an interactive, hierarchical, and dependency-aware interface for customizing kernel options and features before building the kernel.

7.What is the use of primary boot loader?

Ans:

The primary boot loader is responsible for initializing the hardware and loading the secondary boot loader or the operating system kernel, ensuring that the system can start up and execute the desired software.

8.What is Kernel image?

Ans:

A kernel image is a binary file containing the core of an operating system, responsible for managing hardware resources and providing essential services to applications and other system components.

9.Why we using tool chain?

Ans:

We use a toolchain to compile, assemble, and link source code into executable binaries that can run on a target platform, ensuring compatibility and optimization for the specific architecture and operating system.

10.What is defconfig?

Ans:

defconfig" is a commonly used term in the Linux kernel development context, referring to a default configuration file that specifies a set of kernel build options and features to be included when building the kernel.

11.What is the use of secondary bootloader?

Ans:

is responsible for further system initialization, hardware setup, and may also facilitate the loading of the main operating system or a larger and more feature-rich bootloader.

12.What is the size of NAND flash in RuggedBOARD?

Ans:there is no nand flash in RuggedBOARD

13.what are the functions of bootloader?.

Ans:

hardware installation

firmware update

loading os

command line interface.

14.what is porting?.

Ans:

custamizing give s/w in given hardware

15.How can you export a GPIO pin in a rugged board ?

Ans:

echo 17 > /sys/class/gpio/export

16.Write the path to get the NOR Flash images after Compilation of Rugged Board A5d2x.

Ans:

RB-A5D2X/build/tmp/deploy/images/rugged-board-a5d2x/

17.How to enable the environment toolchain for rugged board.

Ans:

. /opt/poky-tiny/2.5.2/environment-setup-cortexa5hf-neon-poky-linux-musleabi.

18.Why we want to use the at91 bootstrap as a primary bootloader.

Ans:

Using the AT91 Bootstrap as a primary bootloader for AT91/SAM microcontrollers offers hardware initialization, flexibility in boot sources, and support for secure boot processes, making it a valuable choice for embedded systems.

19.why do we erase a flash memory before we write any new data.

Ans:

Erasing flash memory before writing new data is necessary to ensure data consistency, maximize memory lifespan, and enable wear-leveling and garbage collection algorithms to operate efficiently

20.What is boot loader version?.

Ans :

2018.07-linux4sam_6.0

Part-B 15*2=30

1.What are the key components of a Device Tree (DTS), and what are the functions provided by the kernel for working with the Device Tree?

ANS:

Key components of a Device Tree (DTS) include:

- **Nodes in DTS:** These represent various hardware components and their properties.
- **Properties of Nodes:** These provide information about the hardware, such as addresses, interrupts, and compatible strings.
- **Kernel APIs:** The kernel provides several functions to work with Device Tree data, including:
    1. `of_platform_populate()`: Used to populate the platform device data from the Device Tree.
    2. `of_driver_match_device()`: Matches a device to a driver based on Device Tree information.
    3. `of_find_node_by_name()`: Locates a device node in the Device Tree by its name.
    4. `of_property_read_u32()`: Reads a 32-bit integer property from a Device Tree node.

These functions are defined in various kernel header files, including `linux/of.h`, `linux/of_platform.h`, and `linux/of_device.h`.

2.What are the steps involved in compiling a Device Tree (DTS) and flashing it onto an embedded Linux system?

ANS:

Compile from Kernel Source:

- Use the `make dtbs` command to compile the Device Tree from the kernel source. This command will generate a `.dtb` (Device Tree Binary) file.

2. **Manual Compilation using dtc:**

- To manually compile a DTS, you can use the Device Tree Compiler (`dtc`). The command format is: `$dtc -I dts -O dtb <dts_filename> -o <dtb_filename>`. This compiles a DTS file into a binary Device Tree file.

3. **Extract DTS from DTB (optional):**

- To extract a DTS file from a DTB file, use the `dtc` command with the appropriate options: `$dtc -I dtb -O dts <dtb_filename> -o <dts_filename>`. This can be useful for editing and modifying the DTS.

3.Why was the Device Tree introduced in embedded Linux.

ANS:

1. Kernel maintainers wanted unified way to add more board without adding board.c files

which are hard to maintain

2. Change in hardware configuration should be possible without re-compiling the kernel

3. Re-use same kernel image for different boards of same SOC

4.Draw the architecture of uboot.

Ans:

5.what are the steps to Adding new command in U-Boot.

Ans:

Step-1 : create dummy.c in u-boot/cmd folder

Step-2: Modify Kconfig file under cmd folder

Step-3: Modify Makefile

Step-4: Compile & Flash

Step-5: Test the command on Target Bootloader prompt.

6. How to port Bootloader on ARM Board.

Ans:

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

2. Check the ARCH & Core support in u-boot location /arch/arm/cpu

3. Check the SOC support location uboot/arch/arm/mach-<soc_family>

4. Create new board folder in u-boot/boards/<board_name>

5. Take ref of existing boards in uboot and develop the code for your board

Add board.c, modify Kconfig & Makefile

6. Create a default configuration file for your board in u-boot/configs

7. Driver level modification if required u-boot/drivers/

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

7.what are the steps to Adding new Driver in U-Boot.

Ans:

#Step-1: Define your device in dts file

Step-2: Define your driver sled.c in uboot/driver folder

Step-3: Add sled configuration in Kconfig file

Step-4: Add sled configuration in Kconfig file

Step-5: Write a test code cmd_sled.c under

command folder and which calls the driver functions

8.explain the sequence of steps involved in flashing U-Boot on the RB-A5D2x using TFTP.

Ans:

Power on board and stop at bootlaoder prompt

#check network connection by pining host PC

u-boot$ ping <serverip>

Download uboot image from PC to Board RAM

u-boot$ tftp 0x21FF0000 u-boot.bin

#erase serial flash(NOR) u-boot partition

u-boot$ sf erase 0x20000 0x80000

copy from uboot image from RAM to NOR Flash

u-boot$ sf write 0x21FF0000 0x20000 0x80000

9.write any 4 functions on U-BOOT Bus Drivers.

Ans:

int gpio_request(unsigned gpio, const char *label);

int gpio_free(unsigned gpio);

int gpio_direction_input(unsigned gpio);

int gpio_direction_output(unsigned gpio, int value);

int gpio_get_value(unsigned gpio);

int gpio_set_value(unsigned gpio, int value);

10.Describe the flow of the driver code in U-Boot.

Ans:

u-boot/driver/gpio/at91_gpio.c Atmel GPIO Driver core bus driver

u-boot/driver/gpio/gpio-uclass.c U-Boot GPIO Subsystem HAL

u-boot/driver/led/sled.c Sled device driver which used gpio bus driver

u-boot/command/sled_cmd.c Test app / command implemented to test sled driver

11.write a ROM Code boot Sequence.

Ans:

12.what are the Components of Toolchain.

Bins

Compiler

Assembler

Linker

Format Convertor

Libs

C Library

pTherad Lib

Other ...

Tools

Debugging tools

13.write the architecture of eLinux System.

Ans:

14.What is the Booting process for Rugged Board?

Ans:

1. Power On SBC

2. SOC BootROM Code will exec

3. BootCFG Pins will define the bootdevice (NAND, NOR, SDCARD .....)

4. From Bootdevice first piece of code (PBL) loaded in SRAM and executed

5. PBL responsible for External RAM Init and loads the BL to External RAM and execute.

6. BL will load the kernel and executes

7. Kernel boots and mounts the RootFS and finally executes the init binary

8. Init will follow init rc scripts to start services and applications

15.What are the images need for Rugged Board and explain the use of each image?

Ans:

BOOT.BIN

- u-boot.bin
- a5d2x-rugged_board.dtb
- zImage
- rb-nor-core-image-minimal-rugged-board-a5d2x.squashfs
- data-image-rootfs.jffs2

16.16.How to change defaualt baud rate in U-Boot for Rugged Board.

Ans:

setenv baudrate 115200

saveenv

boot

17.Modify the bootargs to take the RFS only from SDCard & Test.

Ans:

editenv bootargs

root=/dev/mmcblk1p2 rw rootwait

saveenv

boot

18.Add sled_cmd to test sled driver.

Ans:#

Step-1: Define your device in dts file

$ vim <uboot_path>/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 uboot/driver folder
$ vim <uboot_path>/driver/led/sled.c
# copy the sled.c code

# Step-3: Add sled configuration in Kconfig file
$ vim <uboot_path>/driver/led/Kconfig
config SLED

bool "SLED support for LEDs"
depends on LED
help
Sled driver on RuggedBOARD-A5D2x

# Step-4: Add sled configuration in Kconfig file
$ vim <uboot_path>/driver/led/Makefile
obj-$(CONFIG_SLED) += sled.o

# Step-5: Write a test code cmd_sled.c under
command folder and which calls the driver functions

$ vim <uboot_path>/command/cmd_sled.c
#implement do_sled() & register using U_BOOT_CMD
```