A1
A2

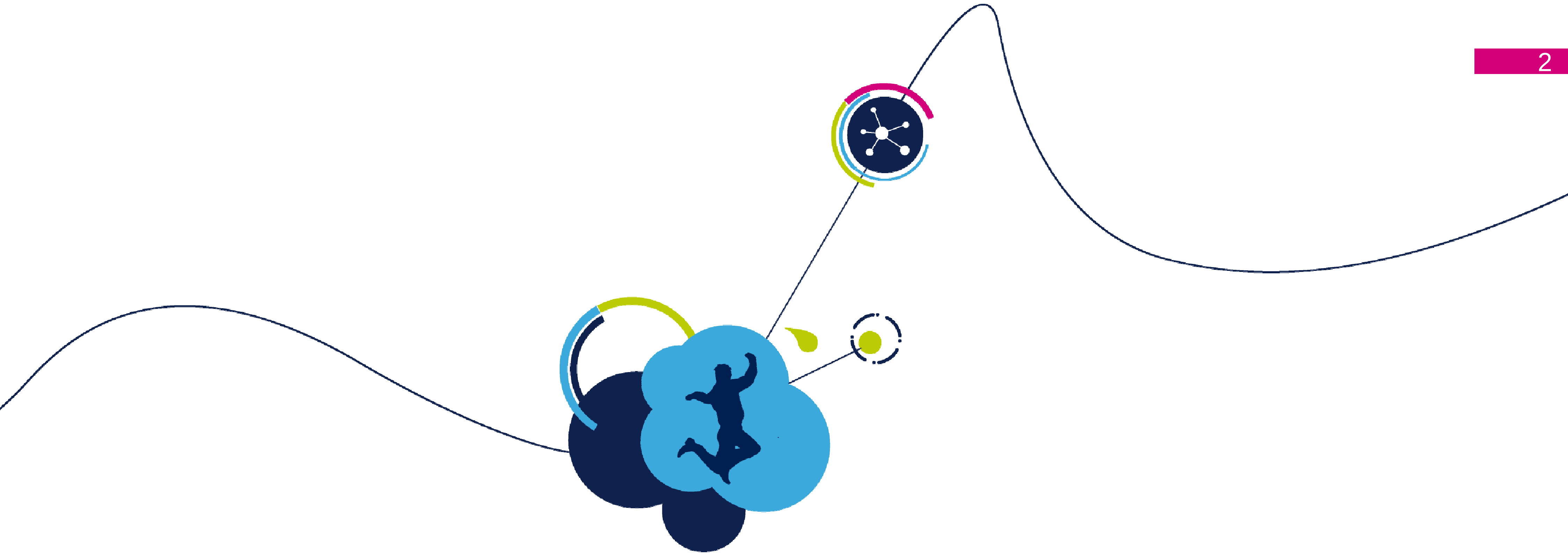
STM32MP1 hands on

Boot

life.augmented

Slide 1

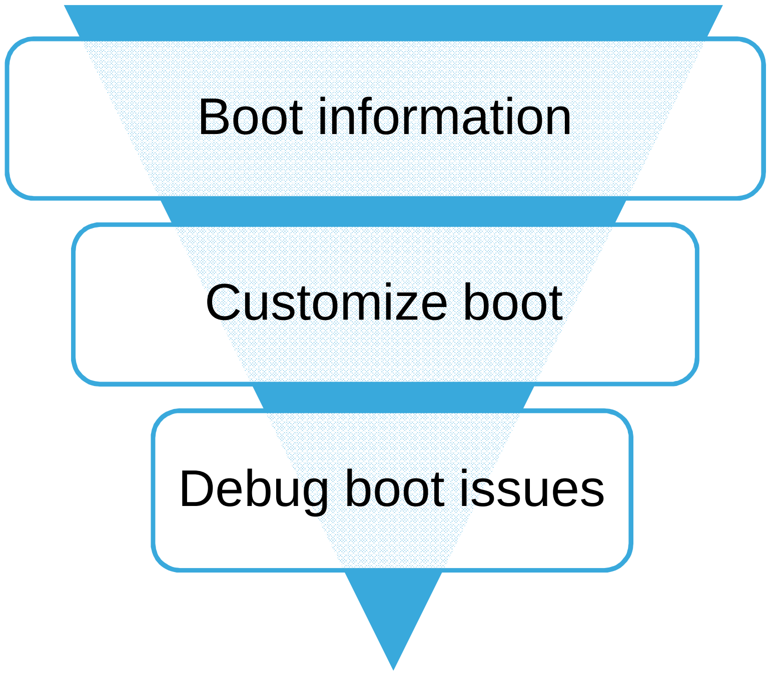
A1	Author; 06/04/2018
A2	general comment: I would have exected more build of both u-boot or ATF with Dev kit. Author; 09/04/2018



Learning program

Objectives

- This learning program provides some guidelines for a “white box” approach of STM32MPU Embedded software (Cortex-A7 and Cortex-M4), to give trainee the tools set for understanding boot, customize and debugging issues.



Slide 3

A3	Better stick to Wiki names = "STM32MPU Embedded Software" Author; 06/04/2018
A4	done Author; 09/04/2018

Benefits

The benefits for the trainee are:

To have a better understanding of boot way of working, how to customize and debug it.

Plan

A5
A6

- Progression is following a common scheme to help trainee from getting simple system information up to make step by step deeper source code debugging.

Update kernel cmdline
Theory + practice: 1h to 2h

Memory mapping
Theory + practice: 1h to 2h

Change splash screen
Theory + practice: 1h to 2h

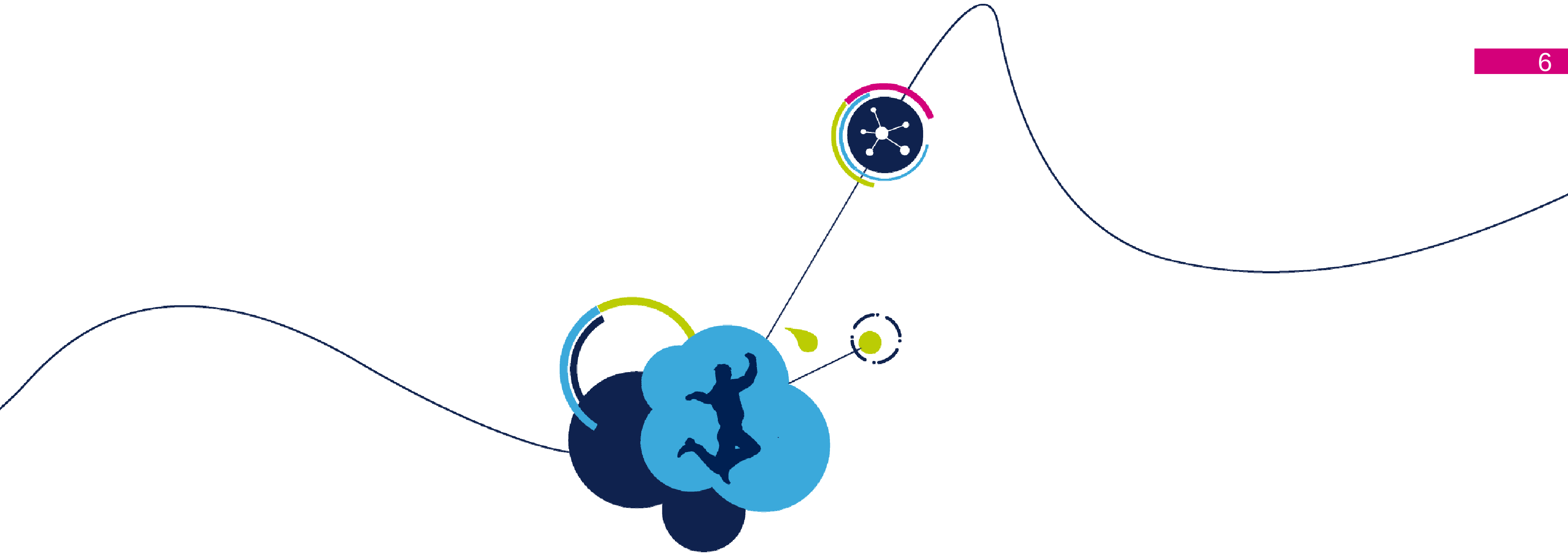
Boot mode : boot from eMMC (Eval board only)
Theory + practice: 1h to 2h

Update security configuration and debug
Theory + practice: 2h to 4h

M4 boot
Theory + practice: 1h to 2h

Slide 5

A5	theme -> scheme ? Author; 06/04/2018
A6	done Author; 09/04/2018



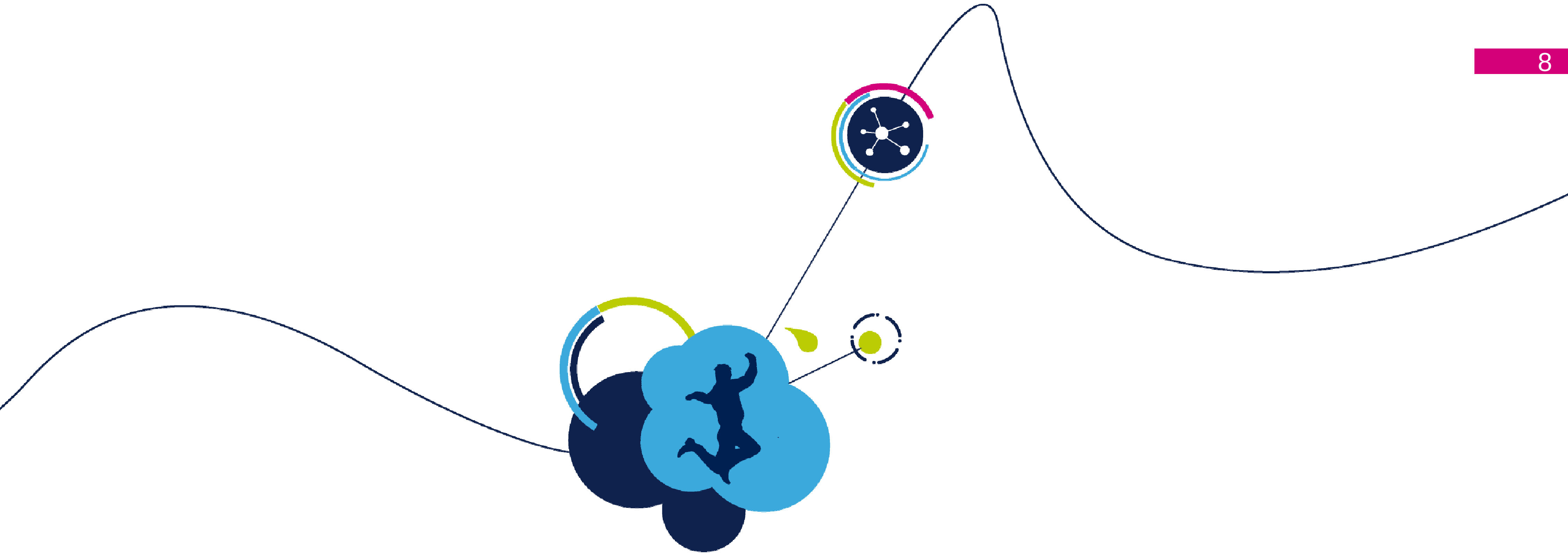
Prerequisites

Prerequisites 1/1

- You must have executed the STM32MP1 OpenSTLinux Distribution Hands on for platform environment setup and play with different kits. All kits are addressed here, but some limitations to be notice with starter kit.

Slide 7

A7	"OpenSTLinux Distribution" is the complete name Author; 06/04/2018
A8	done Author; 09/04/2018



Theoretical school

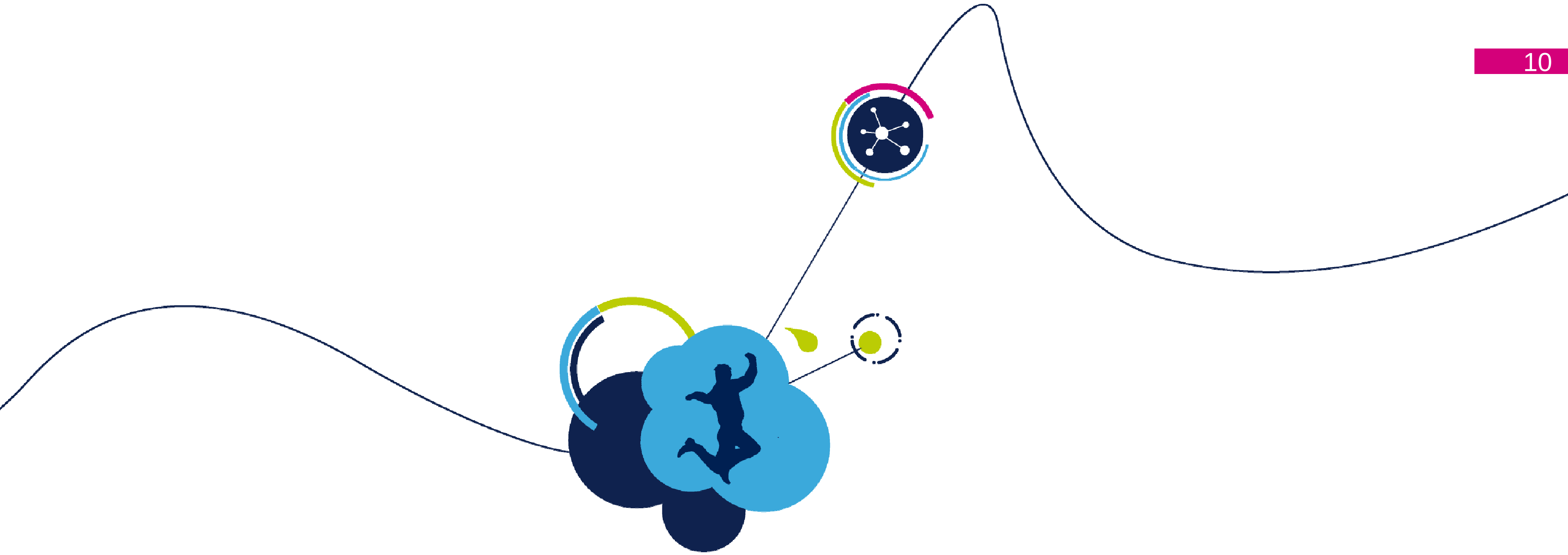
Theoretical school 1/1

- List of reference documentation that may be required for the practical work. Trainee is not supposed to read it now but to make sure he can have access:
  1. {STM32MP15 -Software- Platform boot} = overview of OpenSTLinux boot sequence

- List of internal or external articles to follow before starting practical work (trainee need to understand what he will do later on)
  1. [U-Boot_overview]
  2. [TF-A_overview]
  3. [Coprocessor_management_overview]

Slide 9

A9	Information between <> of the template are for the hands on writer. So maybe to rephase here but anyway to remove the <>. Author; 06/04/2018
A10	done Author; 09/04/2018
A11	Same comment as above Author; 06/04/2018
A12	done Author; 09/04/2018
A13	1, 2 and 3 already seen in prerequisites. Focus on boot only. Author; 06/04/2018
A14	done Author; 09/04/2018
A15	basic -> boot sequence ? Author; 06/04/2018
A16	done Author; 09/04/2018



Practicing school

Lab 1

Practicing school: Update kernel cmdline 1/3

- **Duration**: 1h min

- **Objective**: Understand how u-Boot transmits informations to kernel thanks to extlinux.conf file located in bootfs partition. The purpose of this lab will be to update kernel cmdline by disabling secondary CPU in the kernel.

- **Applicable for starter, developer and distribution kits**

✓ Step1: Boot target and ensure the following log is seen in kernel log:

```
[    0.218963] SMP: Total of 2 processors activated (96.00 BogoMIPS).
```

A18
A19
A20

Practicing school: Update kernel cmdline 2/3

- **Step2 (Starter and Developper Kit):** update exlinux.conf
  - **3 methods to mount bootfs partition :**
    1. **plug SDcard to your PC, and mount bootfs partition (/dev/disk/by-partlabel/bootfs)**
    2. **mount partition though UMS : follow instructions in [STM32MP15_TF-A#Update_via_USB_mass_storage_on_U-boot]**
    3. **On the board, OpenSTLinux mounts bootfs partition in /boot/**
  - **Then edit exlinux.conf file in the folder which corresponds to your board**

```
-APPEND root=/dev/mmcblk0p5 rootwait rw earlyprintk console=ttyS3,115200
+APPEND root=/dev/mmcblk0p5 rootwait rw earlyprintk console=ttyS3,115200 maxcpus=0
```

A17

Slide 13

A17	Bad way of working, you need to use a bbappen and an external layer. But we need to find a trade of for an exercice (maybe some warning ...) ? Author; 06/04/2018
A18	UMS : partition mount in ext4 : linux PC Author; 09/04/2018
A19	Author; 09/04/2018
A20	bootfs mount as /boot in kernel : editable with vi Author; 09/04/2018

Practicing school: Update kernel cmdline 3/3

- **Step3 (Distribution kit):** update exlinux.conf
  - Extlinux.conf content is built in:

    meta-st/meta-st-stm32mp/recipes-bsp/u-boot/u-boot-stm32mp-extlinux.bb
  - Rebuild and reflash bootfs partition.

✓ On next boot, you should see

```
[    0.167010] SMP: Total of 1 processors activated (48.00 BogoMIPS).
```

Slide 14

A21 I don't have it in Alpha Rlelease version. Better update "UBOOT_EXTLINUX_ROOT" variable ...
Author; 06/04/2018

A22 content of patch to be shown before
Author; 09/04/2018

A little Quiz… (click for answers if in Slide show mode)

- **How are named folders containing extlinux.conf in bootfs ?**

  answer : In bootfs, folder names are built from env variables (`printenv` to display them) :

  `${boot_device}${boot_instance}_${board_name}_extlinux`

  these u-boot env variables are built from u-boot device tree.

  More explanations in :

  [U-Boot_overview#Device_tree]

  Developer kit : see <path to u-boot>/board/st/stm32mp1/stm32mp1.c

Lab 2

A23

Practicing school: Flash memory mapping 1/3

- **Duration**: 1h min
- **Objective**: Understand flash memory mapping, and how to get related informations in u-boot and in the kernel.
- Depending on from which device you boot (sdcard, nand, nor, emmc), device's names and partition number could change. This lab will talk about boot from microSD card.
- Flashlayout is described in a *.tsv file given in parameter to flashloader. Example for EV1 board booting from microSD card with Trusted u-boot : FlashLayout_sdcard_stm32mp157c-ev1-trusted.tsv

Slide 17

A23 is there anything missing ? The plan say 1 to 2h ? Could we change the memory mapping for ex ?
Author; 09/04/2018

Practicing school: flash memory mapping 2/3

- **Applicable for starter, developer and distribution kits**

- **Step1 : memory mapping in u-boot console:**
  - **Dump memory mapping:**

```
   Board $> mmc part
```

    (you should see mapping information: partition name, mount point, type, size…)

  - **Read content of bootfs partitions (example for microSD card memory mapping):**

```
   Board $> ext2ls mmc 0:4
```

    (you should see kernel image file, device trees...)

- **Step2 : memory mapping in the kernel console:**
  - **Dump memory mapping:**

```
   Board $> df -h
```

    (you should see mapping information: partition name, mount point, type, size…)

Practicing school: flash memory mapping 3/3

- **Step3 : increase bootfs partition size to 128 Mbytes**

TIPS
- In tsv file, edit Offset of partitions following bootfs, and reflash the new image.

- **Step4 : check that bootfs size has changed**

✓ In the kernel, df command should return 106 Mbytes for bootfs ext4 partition

A little Quiz… (click for answers if in Slide show mode)

- **What is the bootfs partition size in FlashLayout_sdcard_stm32mp157c-ev1-trusted.tsv ?**
  - **Starter/programmer kit : the file is provided in image tarball**
  - **Distribution kit : located in build-*/images/flashlayout*/**

  answer : Extract from the tsv file:

```
1 0x11 bootfs System SDMMC2 0x00284400 st-image-bootfs-openstlinux-weston-extra-stm32mp1.ext4
1 0x12 rootfs FileSystem SDMMC2 0x04284400 st-image-weston-openstlinux-weston-extra-stm32mp1.ext4
```

  So, bootfs partition starts @ 0x00284400 and ends @ 0x04284400, so it takes 64 MBytes

- **u-boot returns mapping addresses in logical blocks. What is the size of a block ?**

  Answer : 0x200 = 512 Bytes

Lab 3

Practicing school: boot from eMMC 1/2

- **ONLY AVAILABLE FOR EVAL BOARD ! Skip this Lab if you have another board.**
- **Duration**: 1h min
- **Objective**: Understand how to change boot mode in order to flash and boot from eMMC.
- **Applicable for starter, developer and distribution kits**
- **Wildcat is able to boot on different devices depending on switch configuration :** [STM32MP157C-EV1_-_hardware_description#Boot_related_switches]

Practicing school: boot from eMMC 2/2

- **Step1 : Download image into the eMMC : [STM32CubeProgrammer#How_to_flash_with_STM32CubeProgrammer]**

- **Step2 : configure boot control to boot from eMMC, and compare logs you get compared with boot from Sdcard.**

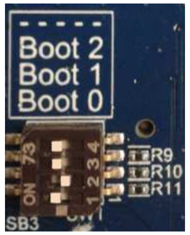
Boot from	U-boot logs	Observed value in kernel cmdline
SDcard	Boot over mmc0!	root=/dev/mmcblk0p5
eMMC	Boot over mmc1!	root=/dev/mmcblk1p3

A little Quiz… (click for answers if in Slide show mode)

- **Why partition number is not similiar between sdcard and eMMC ?**

  hint : have a look into flashlayout files and {STM32MP15-Software- Platform boot}

  answer : for eMMC, FSBL (TF-A) is flashed in dedicated boot partitions (mmcblk1boot0 and mmcblk1boot1 for eval), that explains why there are 2 extra partitions in Sdcard.



- **What is the selected boot mode in the following picture ?**

  hint : have a look [STM32MP157C-EV1_-_hardware_description#Boot_related_switches]

  answer : boot from SDcard

- **At which stage DDR controller is initialized : BootROM, TF-A, u-Boot, Kernel ?**

  answer : TF-A

Lab 4

Practicing school: Update splash screen 1/2

- **Duration**: 1h min (2h if not familiar with picture edition)
- **Objective**: Understand how to update splashscreen.
- **Applicable for starter, developer and distribution kits**
- **By default, u-boot displays a bitmap available in root of one of partition (bootfs for example)**
- **Step 1: create your 8bits bitmap file (with Gimp or Paint for example)**
  - **Keep a reasonable bitmap file size (less than 64 KBytes) to spend less time as possible in file copy.**

Practicing school: Update splash screen 2/2

- **Step 2: download bitmap into bootfs partition:**
  - **3 possibilities:**
    1. **plug SDcard to your PC, and mount bootfs partition (/dev/disk/by-partlabel/bootfs)**
    2. **mount partition though UMS : follow instructions in [STM32MP15_TF-A#Update_via_USB_mass_storage_on_U-boot]**
    3. **boot your kernel with Ethernet network connected:**

```
scp <your bitmap> root@<IP addr of your board>/boot/
```

- **Step 3: rename your bitmap into splash.bmp and reboot**

  - You should see your splashscreen displayed by u-boot.

- **Step 4: (only for developer/distribution kit) by default, splash is centered in both directions. Display it in the upper-left corner.**

  - have a look in splashpos u-boot env value.

A little Quiz… (click for answers if in Slide show mode)

- **What is the splashscreen bitmap load address and why this value ?**

  answer : bitmap must be put into DDR area (from 0xC000 0000 to 0xFFFF FFFF), and mustn't garbage u-boot binary and env which are running at the beginning of the DDR (that explains why 8 MBytes are reserved at the beginning of the DDR.

  So, the bitmap load address is 0xC080 0000.

- **How to customize this address ?**

  Answer:

  - For starter kit, the address is stored in `splashimage` env value, but u-boot env is read only, so it's not possible to update it on the fly, u-boot has to be rebuilt.
  - For developer/distribution kit, The address is given in `<u-boot path>/include/configs/stm32mp1.h` file in `CONFIG_EXTRA_ENV_SETTINGS`

A little Quiz… (click for answers if in Slide show mode)

- **Why the ST bitmap file has a 480x352 px resolution in 8bits ?**

  answer : this resolution is a multiple of 32 px in both directions which is supposed to be aligned/compatible with most of displays, so don’t need to upscale in order to keep ratios.

  Bitmap is converted in 8 bits to decrease the size of the file in order to decrease the duration of loading file into DDR (move bit depth from 24 to 8 will divide bitmap file size by 3)

- **What happends if bitmap file is not found ?**

  answer : as vidconsole output feature is enabled, if u-boot doesn’t find bitmap file, it will display the content of the console. Otherwise, the screen would remain black.

Lab 5

Practicing school: Boot debug 1/3

- **Duration**: 1h min
- **Objective**: Update security configuration in TF-A and analyse the impact with debug tools/methods (trace and GDB).
- Reference : [Category:Trusted_Firmware-A_(TF-A)],[U-Boot_-_How_to_debug], [Gdb], [Gdbgui]
- **Step1** : remove i^{2}c4 from unsecured list <TF-A source path>/fdts/stm32mp157c-security.dtsi : PMIC will not be reachable, and u-boot will freeze.

Practicing school: Boot debug 2/3

- Where does it freeze ? U-boot becomes frozen : enable U-boot debug logs

TIPS

- u-boot seems frozen somewhere between clocks init and NAND init : so enable DEBUG in the appropriated files.
- Logs show that I²C46 clock init never ends. The following log is not displayed :

```
stm32mp1_clk_enable: id clock 160 has been enabled
```

- To investigate a bit more, you add more debug logs, or it can be good to switch GDB debug.

- **Step2 : put in place GDB setup and connect to the target (you can use Gdbgui which offers a better user experience to debug)**

Practicing school: Boot debug 3/3

- In which state is CortexA7 ?

TIPS
- It is running in loop in data_abort vector

- Analyse DFSR register from CP15 : what is the reason of the data abort ?

TIPS
- DFSR=2049=0x801 => Alignment fault caused by write access

TIPS
- More details in Trace & Debug Hands on.

A little Quiz… (click for answers if in Slide show mode)

- **How to easily generate a data abort in u-boot ?**

  answer : by trying to read a forbidden area, a data abort will occur: for example, reading GPIOA bank with a size larger than expected will lead to a data abort:

```
    Board $> md 0x50002000 0x400
```

Lab 6

Practicing school: M4 boot 1/1

- **Duration**: 1h min
- **Objective**: Understand how M4 boots through a quiz
- Reference : [Linux_remoteproc_framework_overview]
- **Applicable for starter, developer and distribution kits**

A little Quiz… (click for answers if in Slide show mode)

- **What are the firmware status after starting or stopping the execution?**

  answer

Resquested state	Firmware state
start	running
stop	offline

A little Quiz… (click for answers if in Slide show mode)

- **At the beginning of dmesg, you can read the following error:**

```
Board $> direct firmware load for rproc-m4-fw failed with error -2
```

Why ?

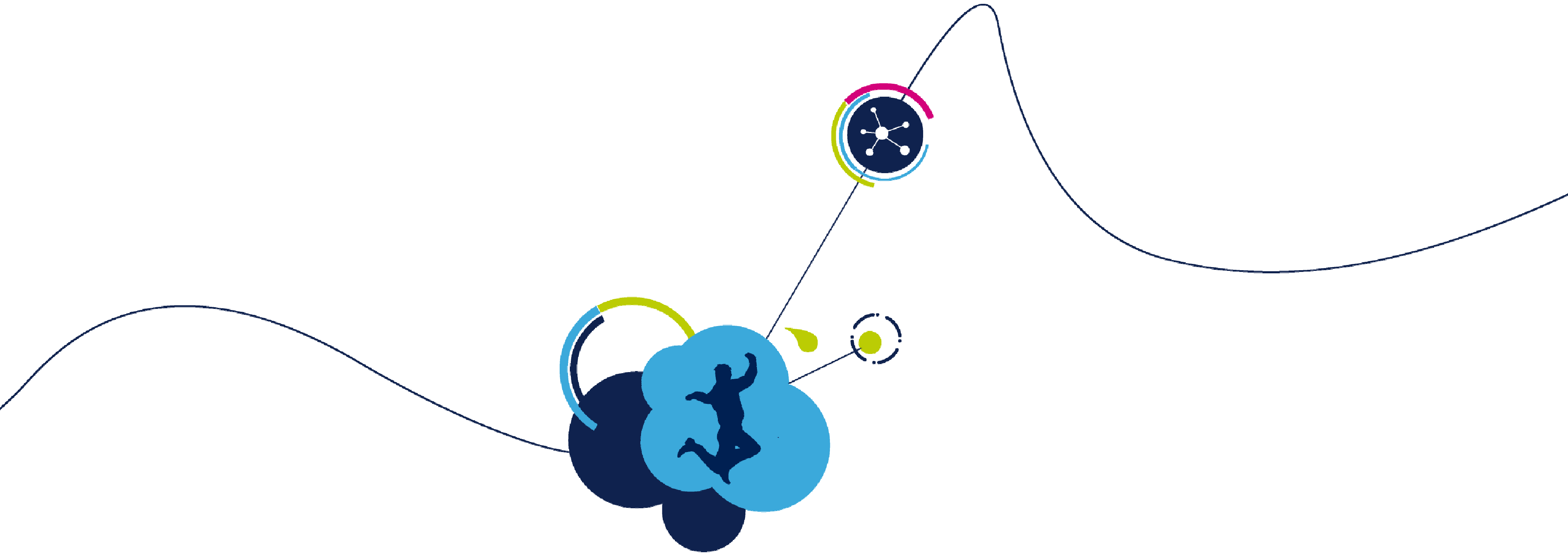
answer : this error is because remoteproc driver wants to load firmware before root partition mount(where firmware is store) : `VFS: mounted root (ext4 filesystem) on device 179:5`

A little Quiz… (click for answers if in Slide show mode)

- **Even if didn’t succeed to load firmware into M4, when checking firmware state, it is running : how is it possible ?**

  answer : At the end of kernel boot, init calls some scripts, and one of them (/sbin/st-m4firmware-load-default.sh) stop firmware, download the firmware (no issue as filesystem is well mounted) and start it.

  For Distribution kit, script is located in meta-st/meta-st-stm32/recipes-extended/m4projects/file/



Congratulations !

life.augmented