life.augmented

DDRFW-UTIL for DDR sanity check

Agenda

1. DDRFW-UTIL overview
2. Hands-on build DDRFW-UTILS & run DDR tests
3. Adapt DDRFW-UTIL to your board

DDRFW-UTIL Overview

DDRFW-UTIL overview

- **DDRFW-UTIL,** a firmware for DDR initialization and DDR tests

- **When these tests should be run ?**
  - At bring-up phase for **board sanity-check** of DDR bus for confidence prior to production
  - Default test parameters meant to catch low-margin timings of the PCB over the STM32MP1 temperature range
  - DDR errors, to pin-point the root cause

    but

  Does not happen when PCB designers respects “**AN5692** STM32MP13x lines DDR memory routing guidelines”

DDRFW-UTIL overview

Tests

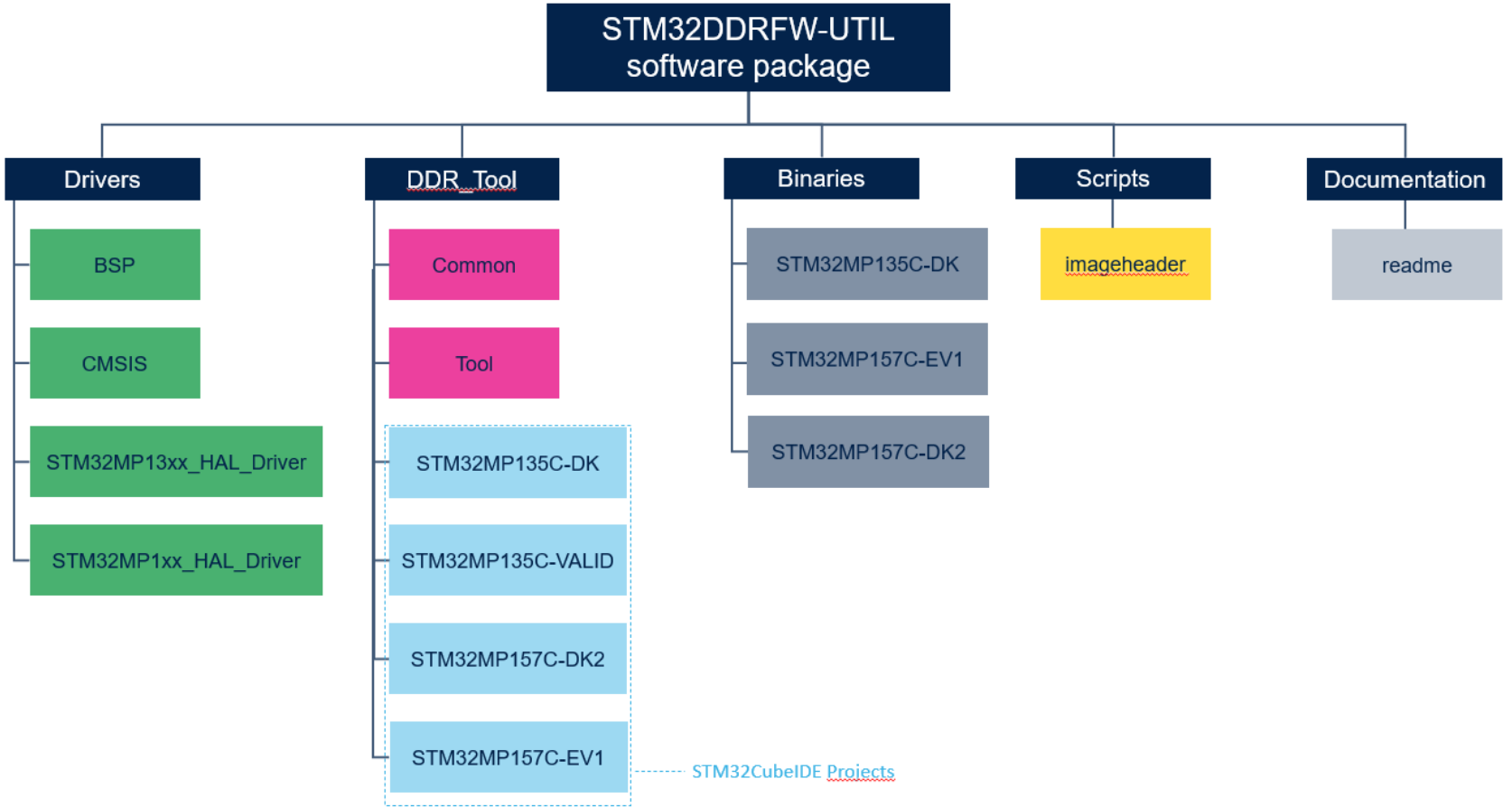
- Basic tests: These simple and running fast tests are intended to capture the major configuration or hardware issues showing off immediately.
- Intensive tests: These tests use extensive coverage of data and address patterns for noise and high SSO conditions, high throughput traffic or interleaved read/write.
- Stress tests: These tests are intensive and executed with stretched conditions

TEST	TEST NAME	TYPE	DESCRIPTION
1	**Simple Databus**	Basic	Verifies each data bus signal can be driven high at the given address.
2	**Databus Walking 0**	Basic	Verifies each data bus signal can be driven low.
3	**Databus Walking 1**	Basic	Verifies each data bus signal can be driven high.
4	**Address Bus**	Basic	Verifies each address bus line in a memory region by performing a walking 1 test on the relevant address bits and checking for aliasing.
5	**Mem Device**	Intensive	Performs read/write over an entire memory region. Each data bit is written and read back with 0 and 1 values.
6	**Simultaneous Switching Output**	Intensive	Stresses the data bus over an address range by doing simultaneous switching output. Writes a pseudo-random value and reads it back.
7	**Noise**	Intensive	Verifies read/write while forcing switching of all data bus lines.
8	**Noise Burst**	Intensive	Verifies read/write while forcing switching of all data bus lines (test based on 8-word bursts).
9	**Random**	Intensive	Verifies read/write with a pseudo-random value on one region.
10	**Frequency Selective Pattern**	Intensive with stress	Stresses data bus by performing successive write 8-word burst operations using mostly zero/one patterns and frequency divider patterns (F/1, F/2, F/4) for 16 and 32 data bus width.
11	**Block sequential**	Intensive	Well known user-space memory tester adapted.
12	**Checkerboard**		
13	**Bit spread**		
14	**Bit flip**		
15	**Walking ones**		
16	**Walking zeroes**		
17	**Infinite read**	Basic	Performs an infinite read for a specific pattern (for debug and lab usage only, not visible).
18	**Infinite write**	Basic	Performs an infinite write access to DDR (for debug and lab usage only, not visible).

DDRFW-UTIL overview

Project architecture

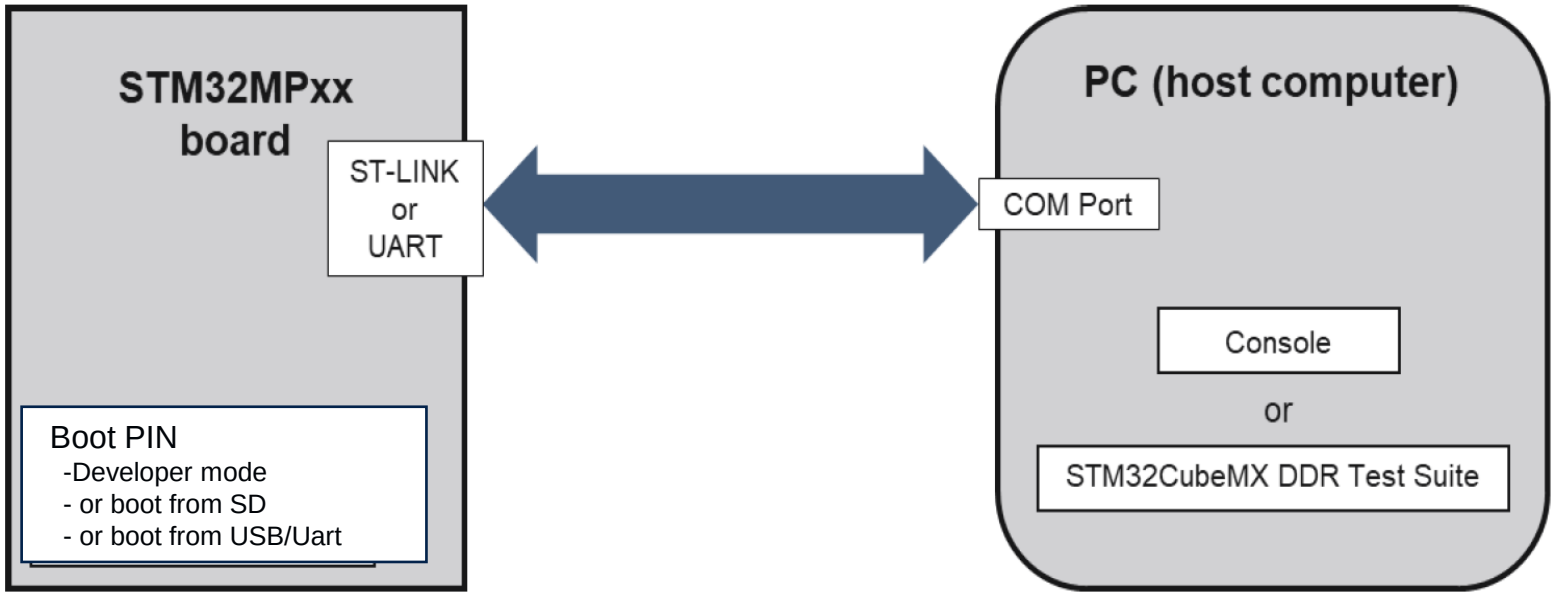
- Firmware delivered as a STM32CubeIDE project, can be generated under Windows OS.
- Source code can be adapted to PCB
- Use same drivers as Azure-RTOS STM32MP13 bare metal



DDRFW-UTIL overview

HW connection for test control in DDRFW-UTIL console

- UART communication is handled either by the ST-LINK VCP, or by a dedicated port.
- By default, UART4 with Bps=115200 Bit=8 Parity=None Stop=1,
  UART can be changed in DDRFW-UTIL sources



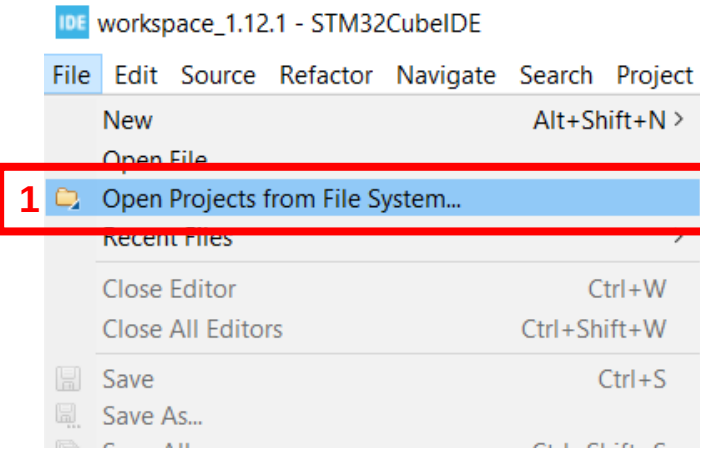
Hands-on Build DDRFW-UTIL and run DDR tests

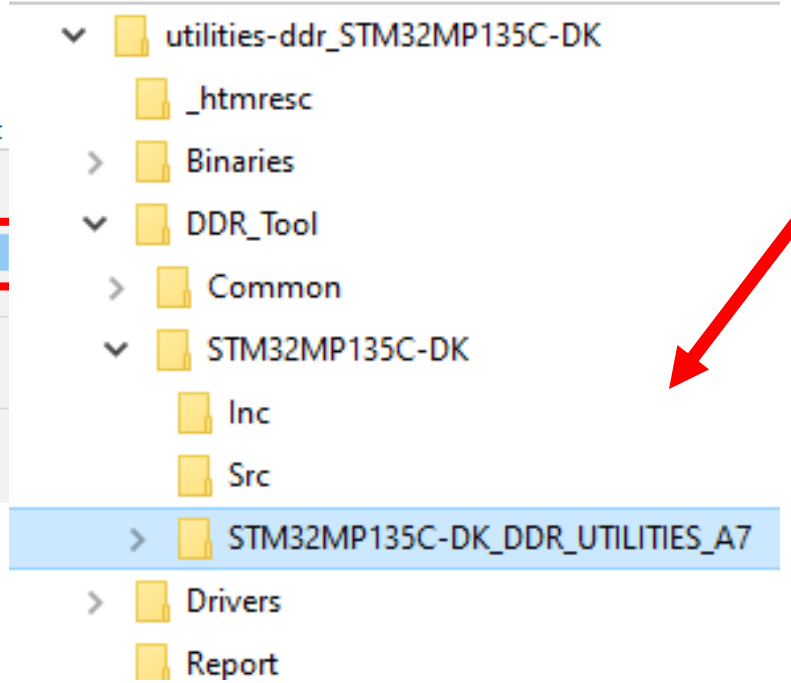
Objectives

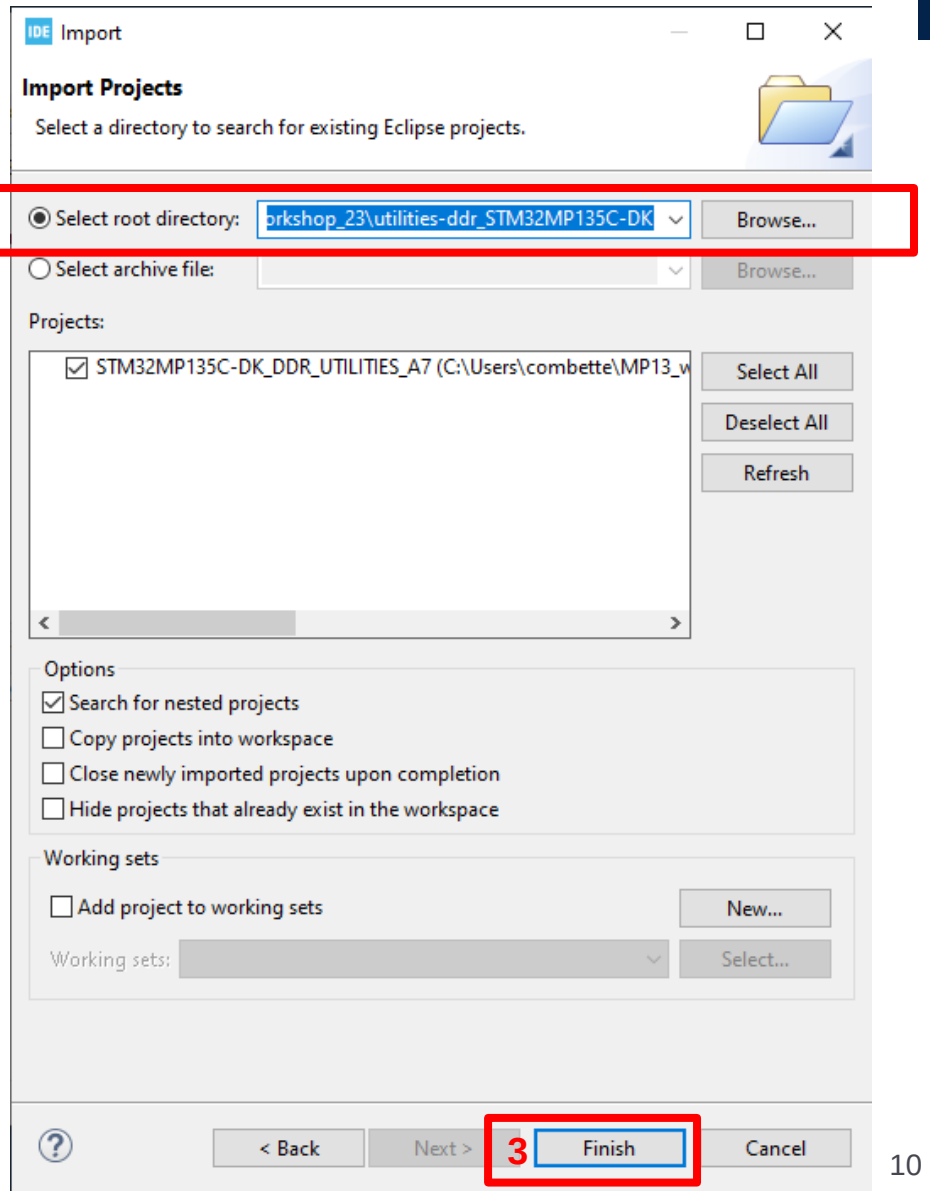
- Run the DDR tests on STM32MP135F-DK board

Build the firmware

- Extract C:\ST_Workshop\STM32DDRFW-UTIL_for_STM32MP135C-DK.zip in C:\ST_Workshop\
- Open STM32MP135C-DK_DDR_UTILITIES STM32CubeIDE project

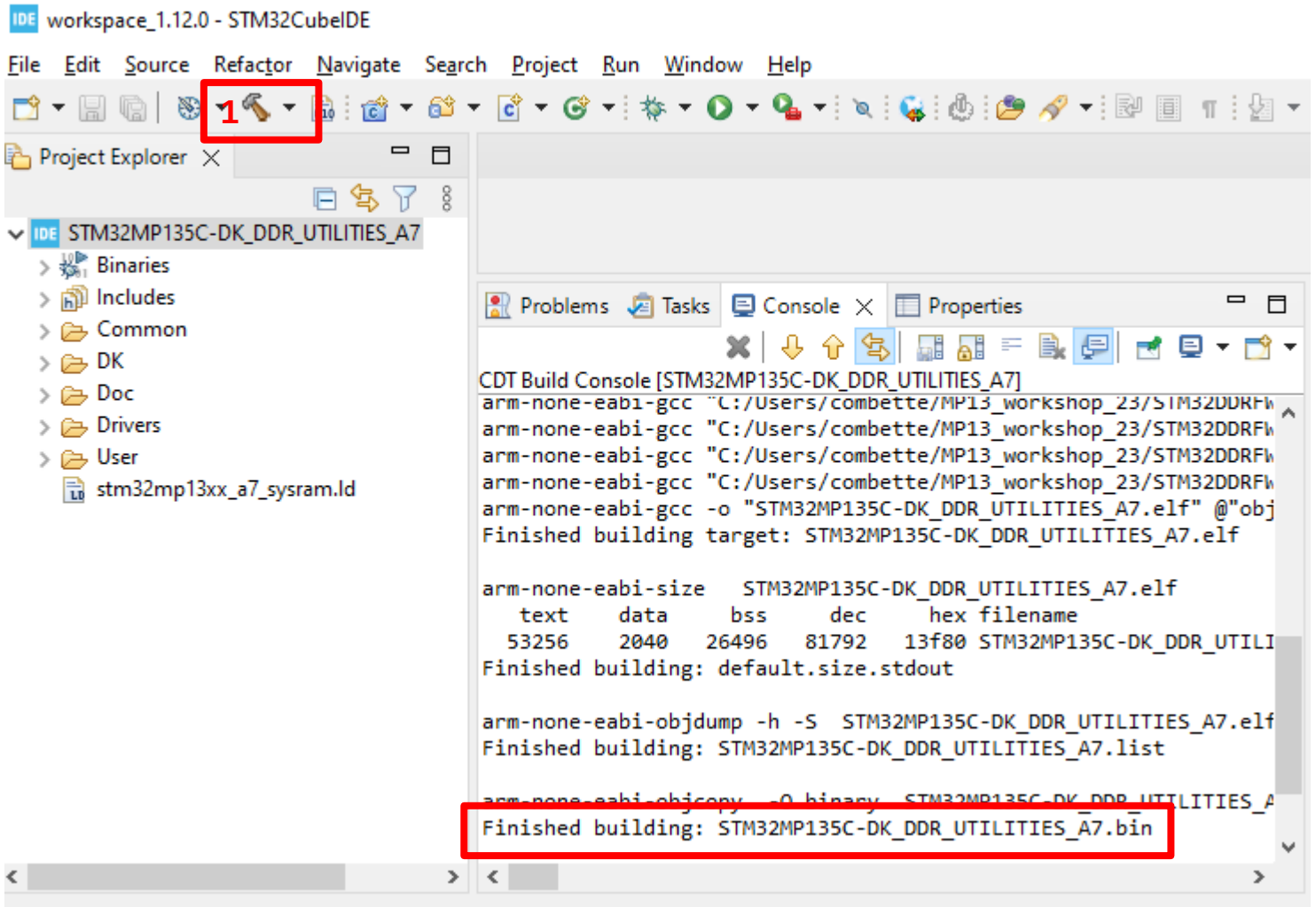






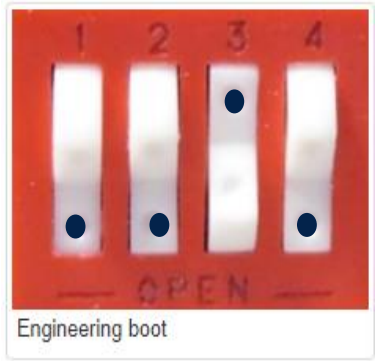
Build the firmware

Note: at that stage, DDRFW-UTIL source code must be adapted to customer PCB for settings for UART, STPMIC I2C, PLL frequency of DDR controller



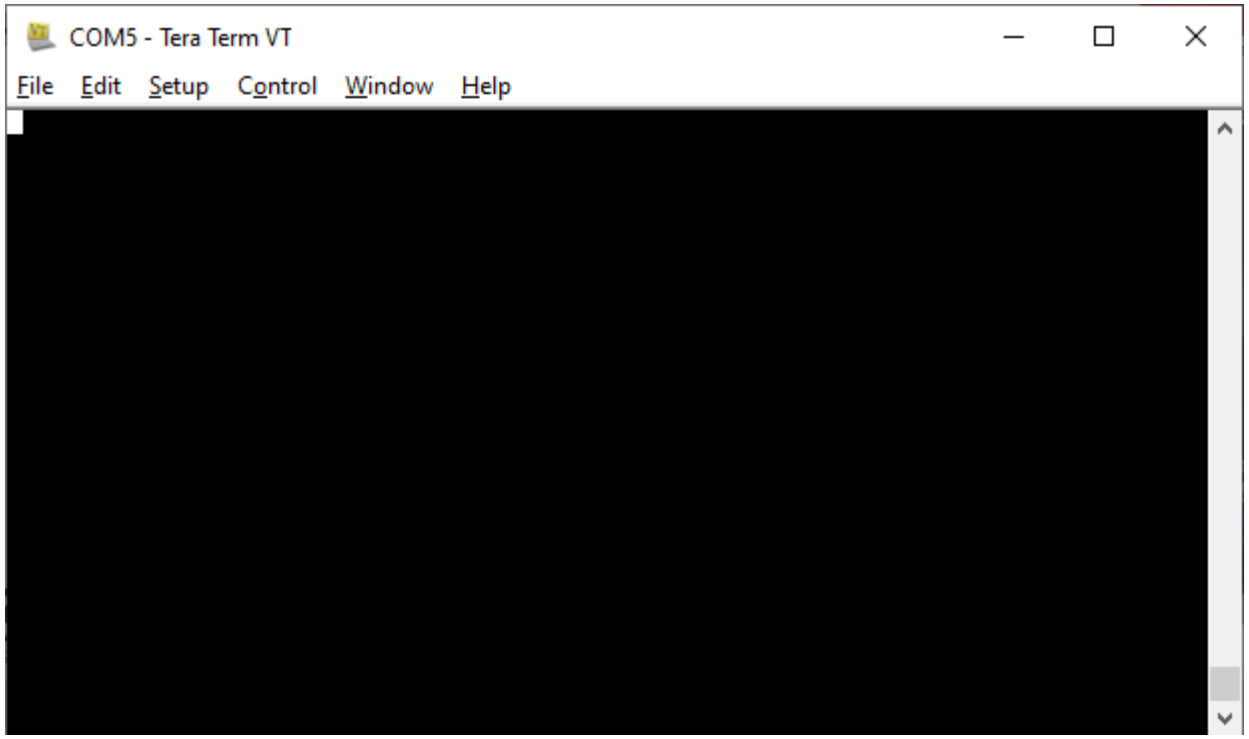
Setup

- Set boot pin in developer mode
- Keep console opened
- Reset the board



Engineering boot

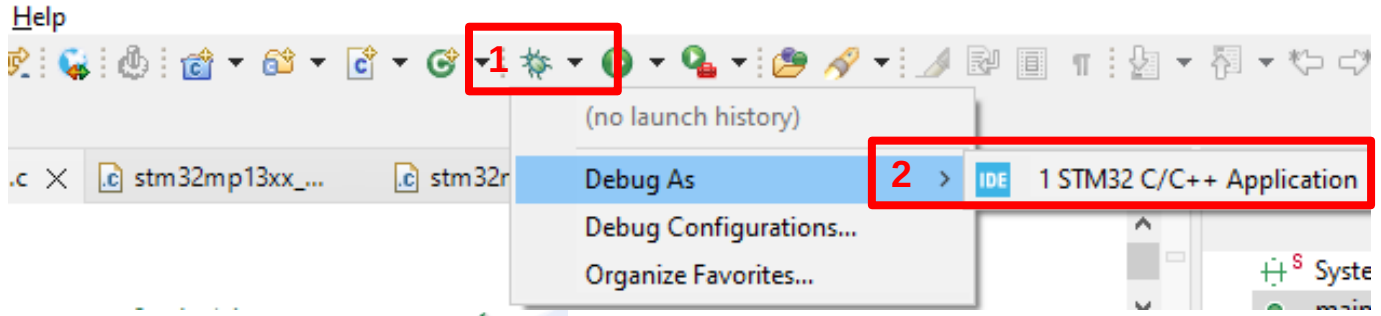
Red LED & screen is black

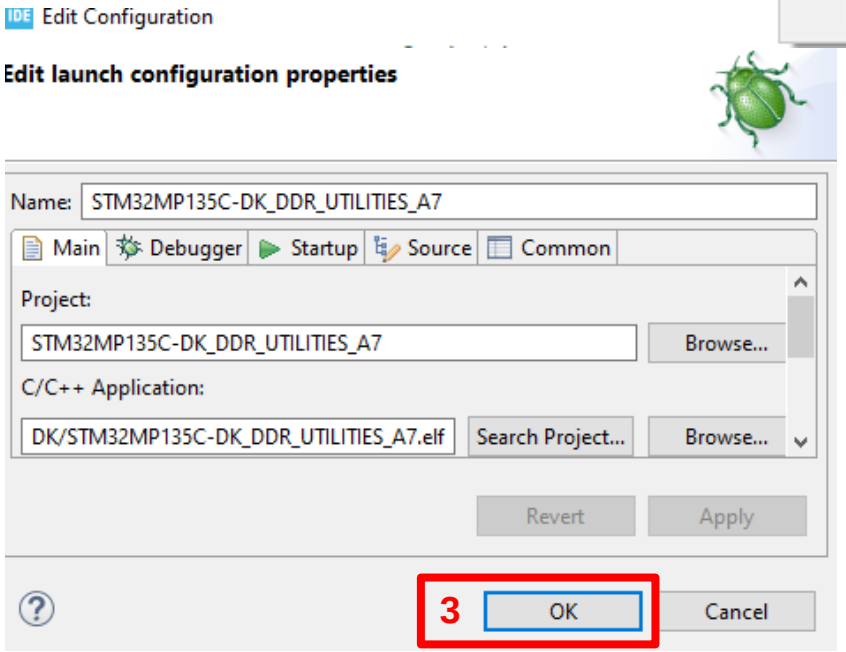


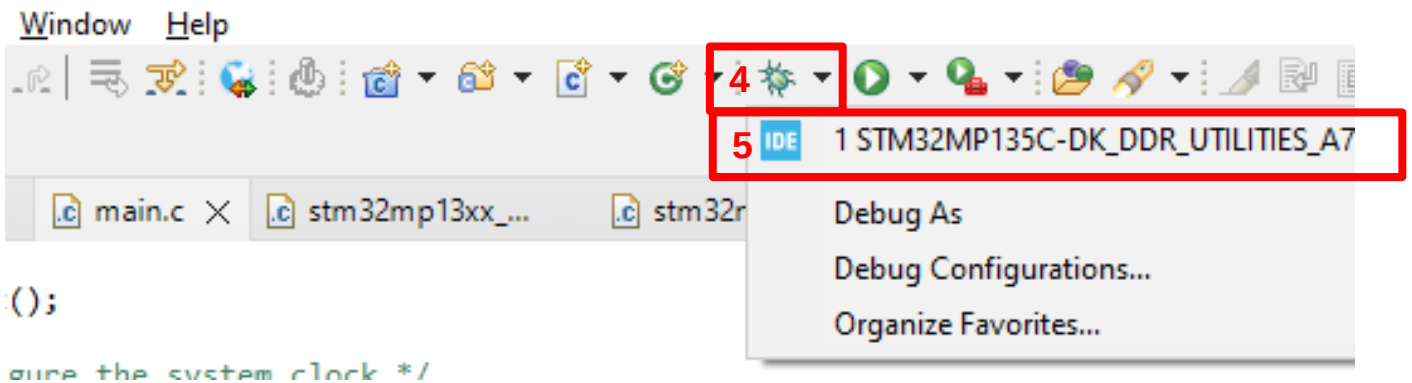
life.augmented

Run DDRFW-UTIL

- Create & start a debug session

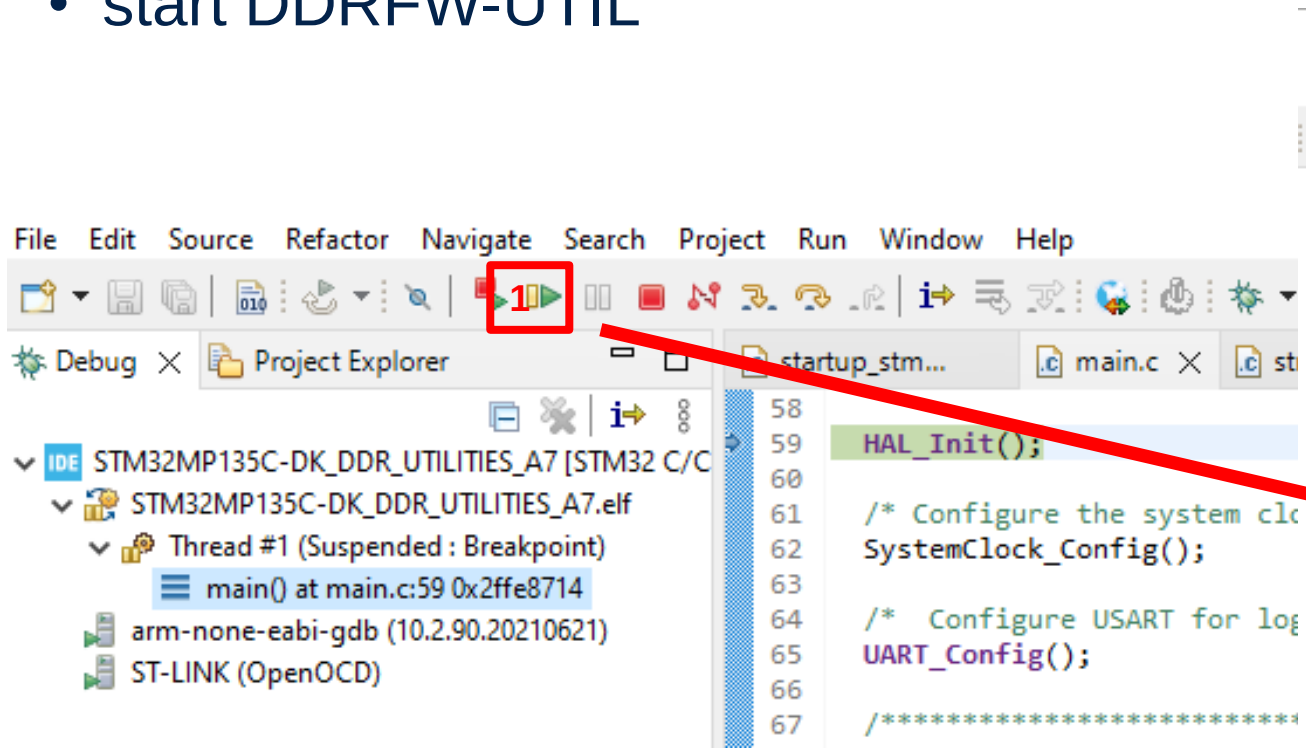


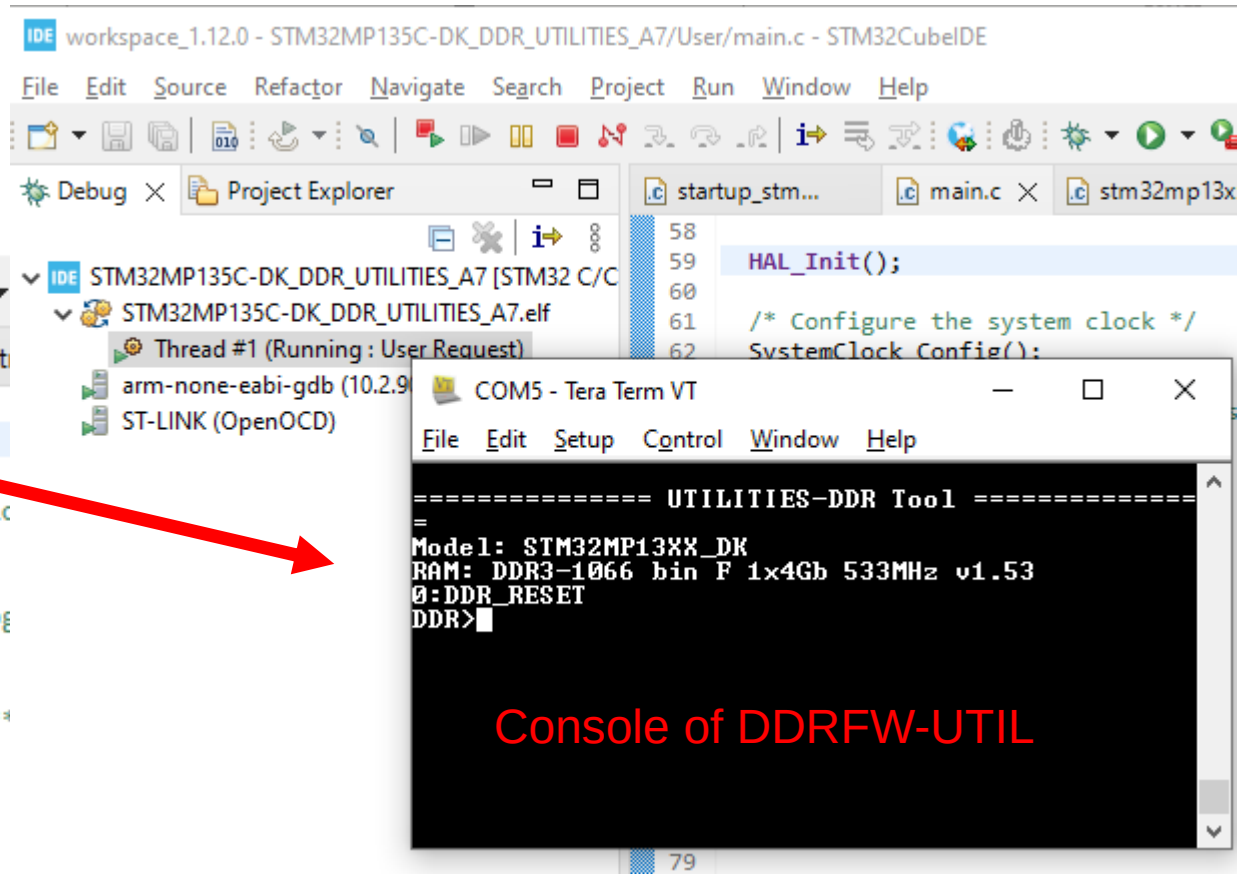




Run DDRFW-UTIL

- start DDRFW-UTIL

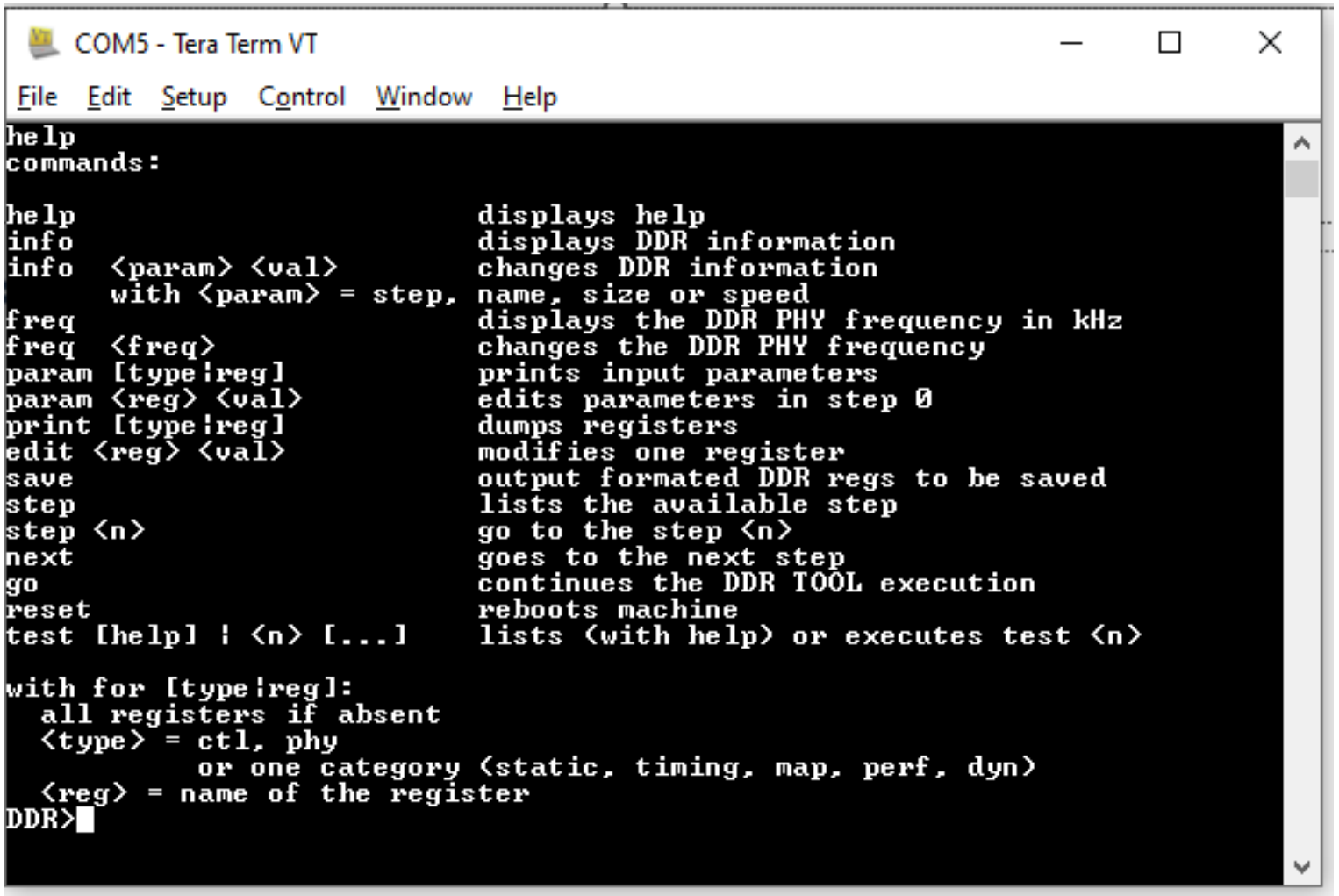




Note : Debug loads FW in SYRAM via STLINK SWD
Fw is breaked at main()

DDRFW-UTIL Commands

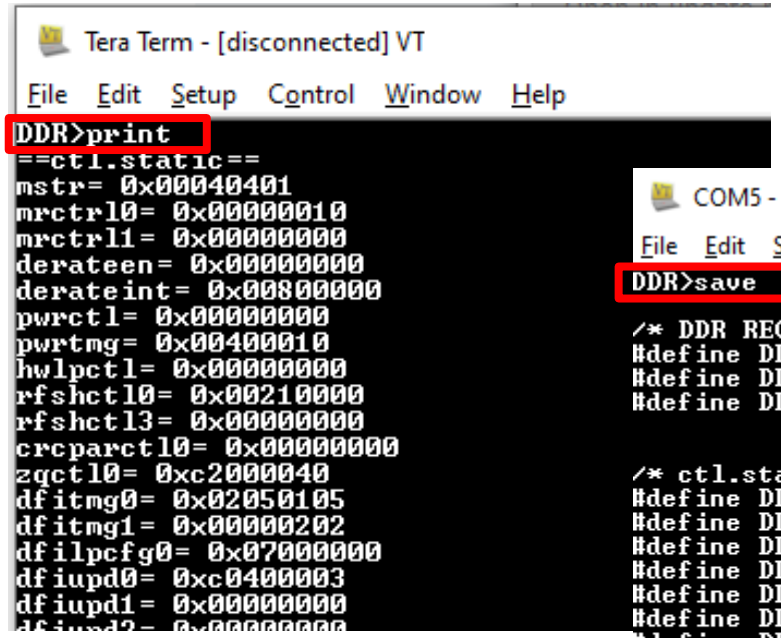
- Commands of DDRFW-UTIL



DDRFW-UTIL commands

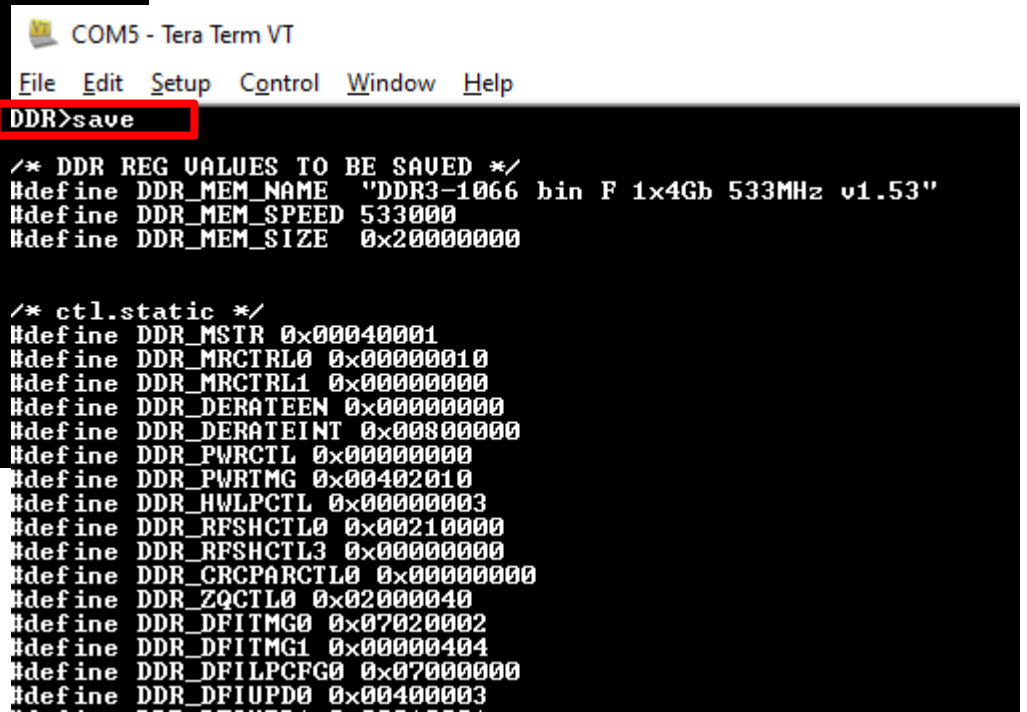
- Other DDR Ctrl commands for debug

DDR>print
see register value

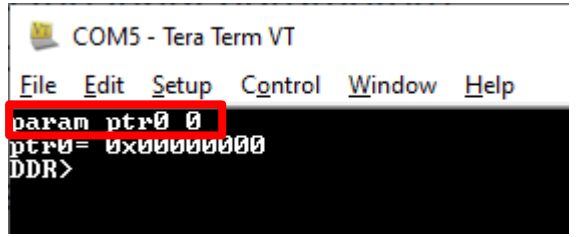


The "print" and "edit" directly access the CTRL and PHY registers

DDR>save
To save in xxx-template.h format



DDR>param
To change value before initialization

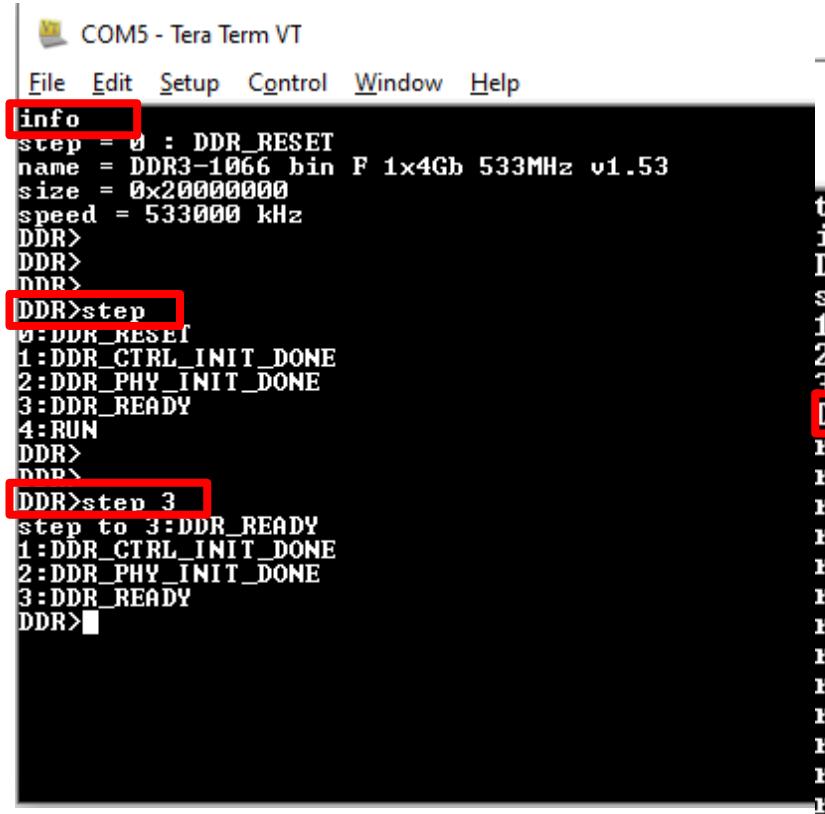


life.augmented

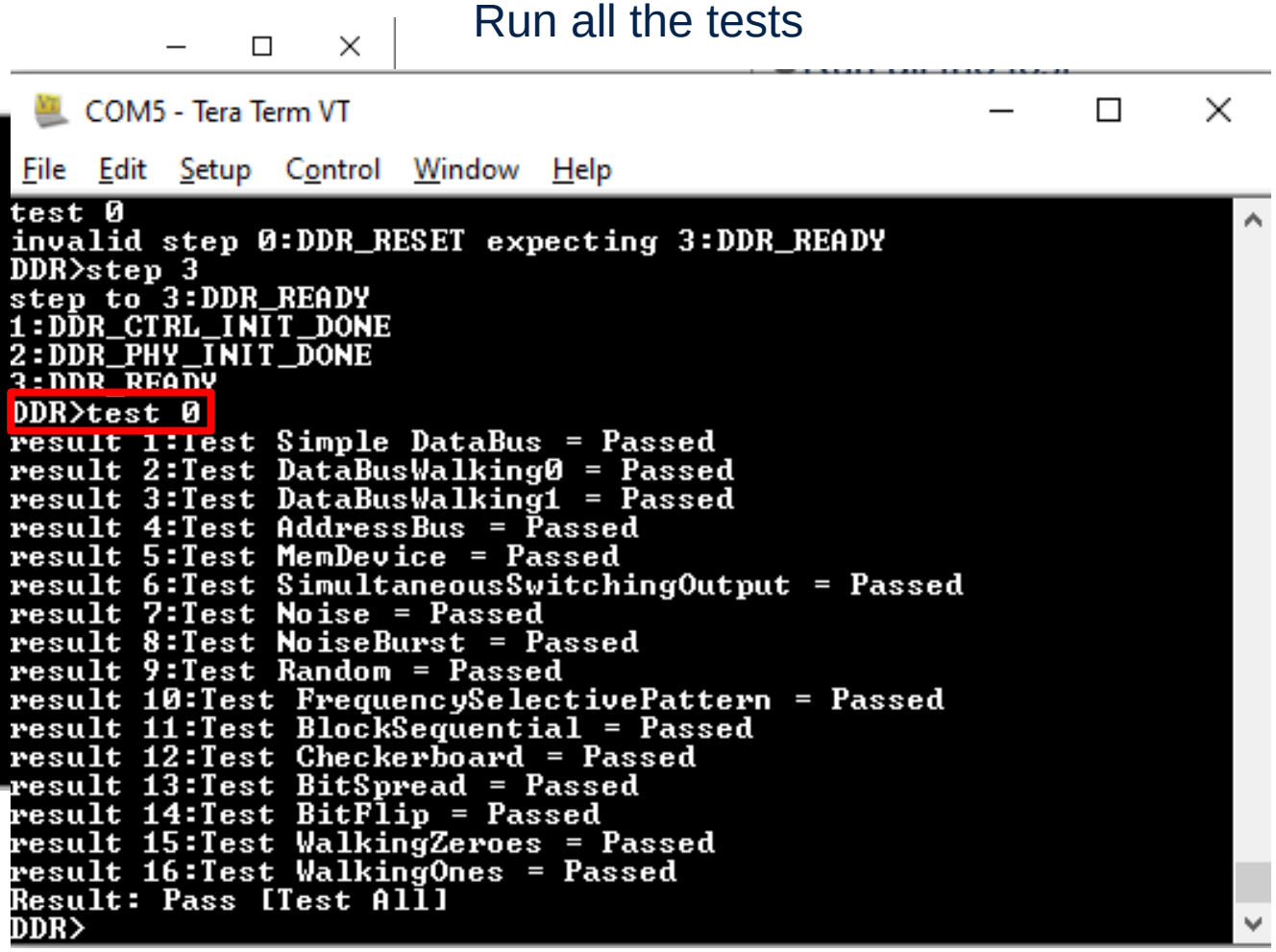
Run DDR test

- sanity check of customer PCB

DDR>step 3
First initialize DDR controller



DDR> test 0
Run all the tests



life.augmented

Run DDR test

- Check the memory integrity, the bus

  - Test the integrity of a physical memory

Check each bit of the memory of first 100Mbytes (set 0 and 1)
DDR> test 5 0x6400000 0xC0000000

  - Test the Simultaneous Switching Output.

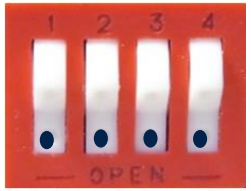
Verifies successive reads and writes to the same memory word, holding one bit constant while toggling all other data bits simultaneously
Over the first 512 Kbytes
DDR> test 6 0x80000 0xC0000000

  - Test the data bus

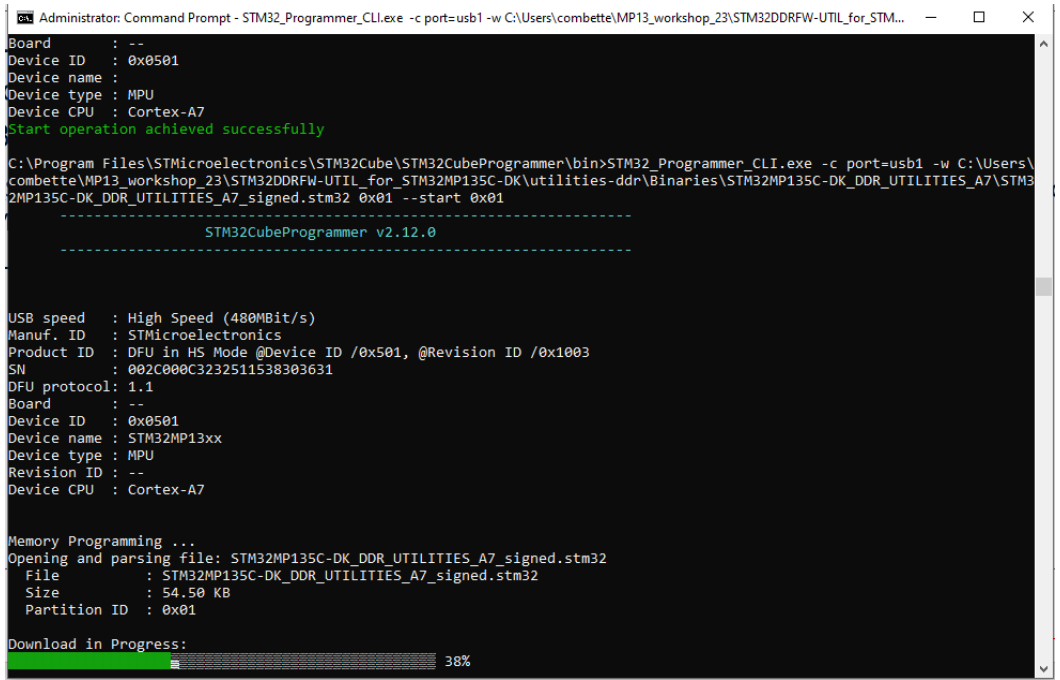
Verifies each data line by walking 1 on fixed address.
Test the data bus wiring in a memory region by performing a walking 1's test at a fixed address within that region.
DDR> test 1 0xC0000000

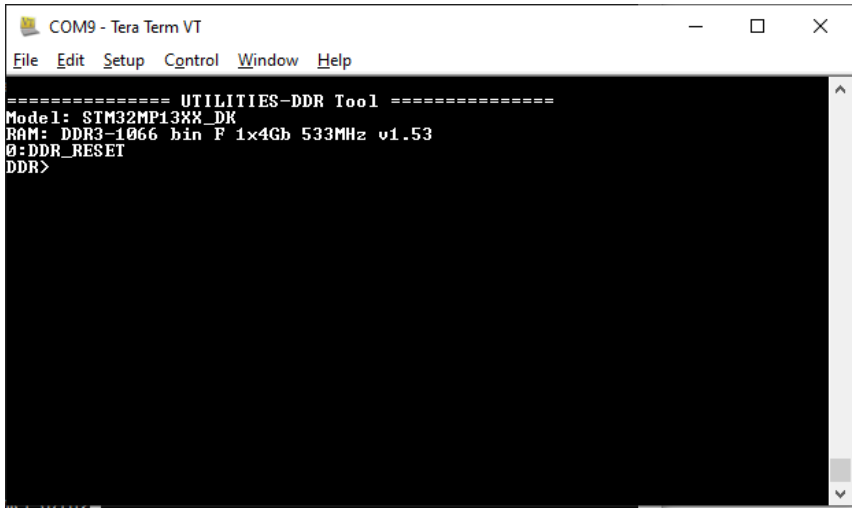
DDRFW-UTIL in standalone

- DDRFW-UTIL started in SYSRAM by USB DFU ROM code
  - Set boot pin to Boot from USB , connect CN7 USB OTG cable to PC
  - Use the .stm32 with STM32CubeProgrammer
- cd C:"\Program Files"\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin
- STM32_Programmer_CLI.exe -c port=usb1 -w C:\ST_Workshop\utilities-ddr_STM32MP135C-DK\DDR_Tool\STM32MP135C-DK\STM32MP135C-DK_DDR_UTILITIES_A7\DK\STM32MP135C-DK_DDR_UTILITIES_A7.stm32 0x01 --start 0x01



boot switches position to program the board via the USB

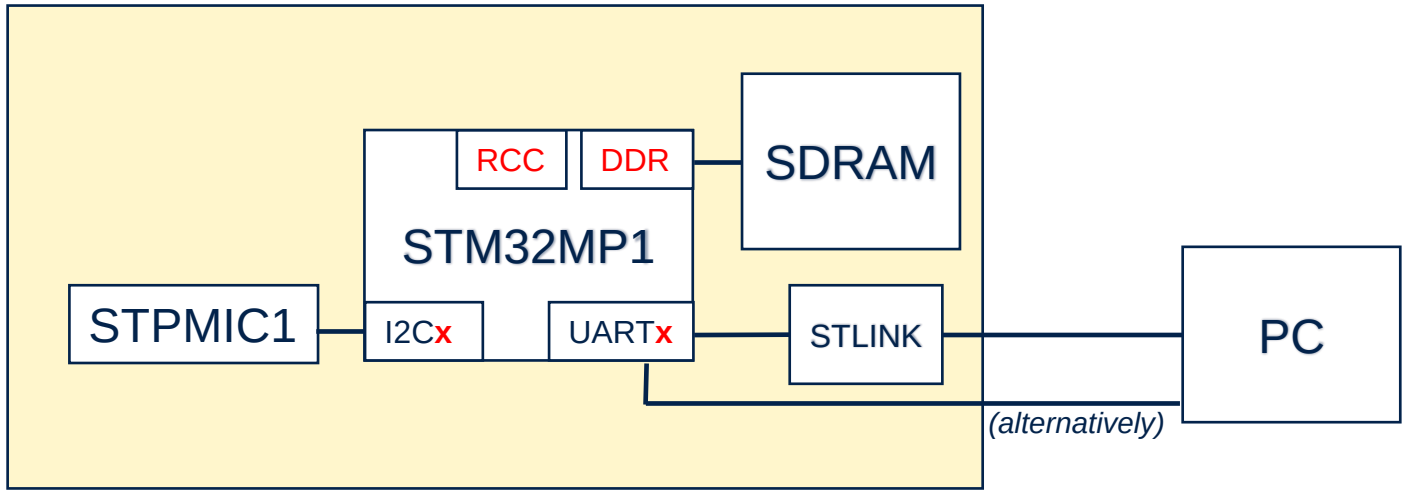




Adapt DDRFW-UTIL to your board

Customize to PCB

Possible DDRFW-UTIL modifications



Customize to PCB

Project

```
<utilities-ddr_STM32MP135C-DK>
    |_ README.md
    |_ <Binaries>
    |    |_ <STM32MP135C-DK_DDR_UTILITIES_A7>
    |    |    |_ STM32MP135C-DK_DDR_UTILITIES_A7_signed.stm32        (to be loaded on SDCARD or via USB DFU)
    |_ <DDR_Tool>
    |    |_ <Common>
    |    |    |_ <Inc>
    |    |    |    |_ stm32mp_util_def.h
    |    |    |    |_ system_time.h
    |    |    |    |_ log.h
    |    |    |_ <Src>
    |    |    |    |_ system_time.c
    |    |    |    |_ syscalls.c
    |    |_ <STM32MP135C-DK>
    |    |    |_ <Inc>
    |    |    |    |_ ddr_tests.h
    |    |    |    |_ ddr_tool.h
    |    |    |    |_ ddr_tool_util.h
    |    |    |    |_ main.h
    |    |    |    |_ RTE_Components.h
    |    |    |    |_ stm32_device_hal.h
    |    |    |    |_ stm32mp13xx-util-ddr3-4Gb-template.h        (DDR registers setting given as example)
    |    |    |    |_ stm32mp13xx_disco_conf.h
    |    |    |    |_ stm32mp13xx_hal_conf.h
    |    |    |    |_ stm32mp13xx_it.h
    |    |    |    |_ stm32mp13xx-util-lpddr2-4Gb-template.h      (given as example)
    |    |    |    |_ stm32mp13xx-util-lpddr3-4Gb-template.h      (given as example)
    |    |    |    |_ stm32mp_util_conf.h                         (to be adapted for customize board)
    |    |    |    |_ stm32mp_util_ddr_conf.h                     (select DDR according to board memory)
    |    |    |_ <Src>
    |    |    |    |_ ddr_tests.c                                 (common file that implements all DDR tests)
    |    |    |    |_ ddr_tool.c (common file that implements HAL_DDR_Interactive function with all available tool commands)
    |    |    |    |_ ddr_tool_util.c                             (board specific file that handles uart communication)
    |    |    |    |_ main.c
    |    |    |    |_ stm32mp13xx_hal_msp.c
    |    |    |    |_ stm32mp13xx_it.c
    |    |    |_ <STM32MP135C-DK_DDR_UTILITIES_A7>
    |    |    |    |_ stm32mp13xx_a7_sysram.ld
    |    |    |    |_ .project
    |    |    |    |_ .cproject
    |_ <Drivers>
    |    |_ <BSP>
    |    |    |_ <STM32MP13xx_DISCO>
    |    |_ <CMSIS>
    |    |    |_ <MP1>
    |    |    |    |_ ...
    |    |    |_ <Device>
    |    |    |    |_ <ST>
    |    |    |    |    |_ <STM32MP13xx> (board specific file that handles stpmic regulators and i2C, leds)
    |    |_ <STM32MP13xx_HAL_Driver>
```

DDR PHY reg values depending on DDR

settings for UART, PMIC I2C
PLL frequency of DDR controller

stm32mp_util... | stm32mp13xx_... | stm32mp13xx_...

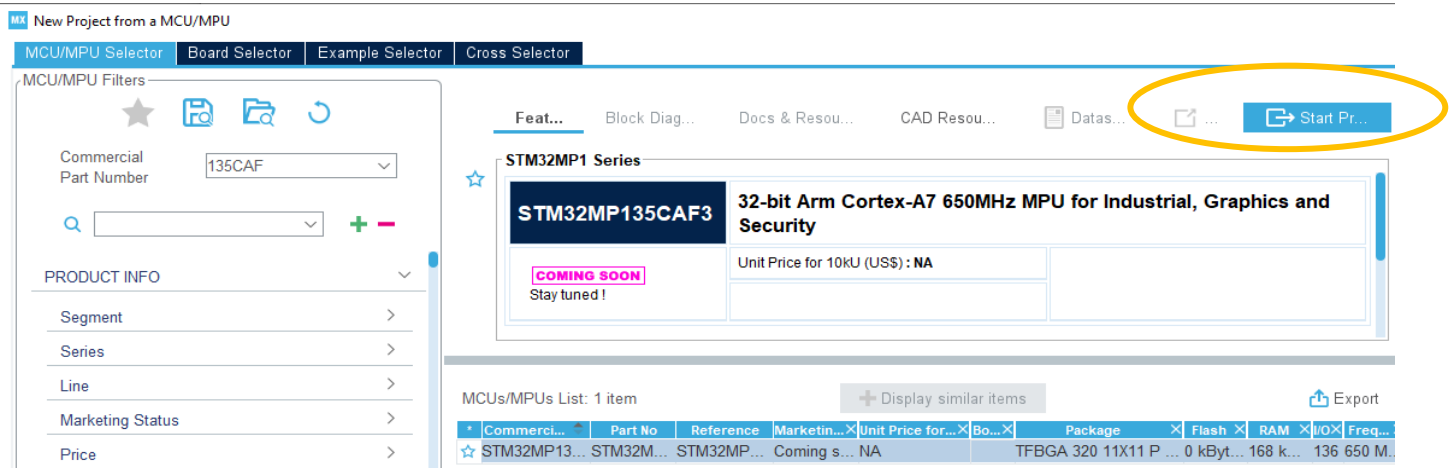
```
31  #define UTIL_PLL2_P          2
32  #define UTIL_PLL2_Q          2
33  #define UTIL_PLL2_R          1
34  #define UTIL_PLL2_FRACV      0x1400
35  #define UTIL_PLL2_MODE       RCC_PLL_FRACTIONAL
36  #define UTIL_PLL2_DIVN_MIN   24
37  #define UTIL_PLL2_DIVN_MAX   199
38
39  /* UART related configuration */
40  #define UTIL_UART_INSTANCE          UTIL_UART4
41  #define UTIL_UART_TX_PIN            GPIO_PIN_6
42  #define UTIL_UART_TX_GPIO_PORT      UTIL_GPIOD
43  #define UTIL_UART_TX_AF             GPIO_AF8_UART4
44  #define UTIL_UART_RX_PIN            GPIO_PIN_8
45  #define UTIL_UART_RX_GPIO_PORT      UTIL_GPIOD
46  #define UTIL_UART_RX_AF             GPIO_AF8_UART4
47  #define UTIL_UART_BAUDRATE          115200
48  #define UTIL_UART_WORDLENGTH        UART_WORDLENGTH_8B
49  #define UTIL_UART_STOPBITS          UART_STOPBITS_1
50  #define UTIL_UART_PARITY            UART_PARITY_NONE
51  #define UTIL_UART_HWFLOWCTL         UART_HWCONTROL_NONE
52  #define UTIL_UART_MODE              UART_MODE_TX_RX
53  #define UTIL_UART_OVERSAMPLING      UART_OVERSAMPLING_16
54
55  /* PMIC related configuration */
56  #define UTIL_USE_PMIC                       1
57  #define UTIL_PMIC_I2C_PORT                  UTIL_I2C4
58  #define UTIL_PMIC_I2C_SCL_PIN               GPIO_PIN_15
59  #define UTIL_PMIC_I2C_SCL_GPIO_PORT         UTIL_GPIOE
```

life.augmented

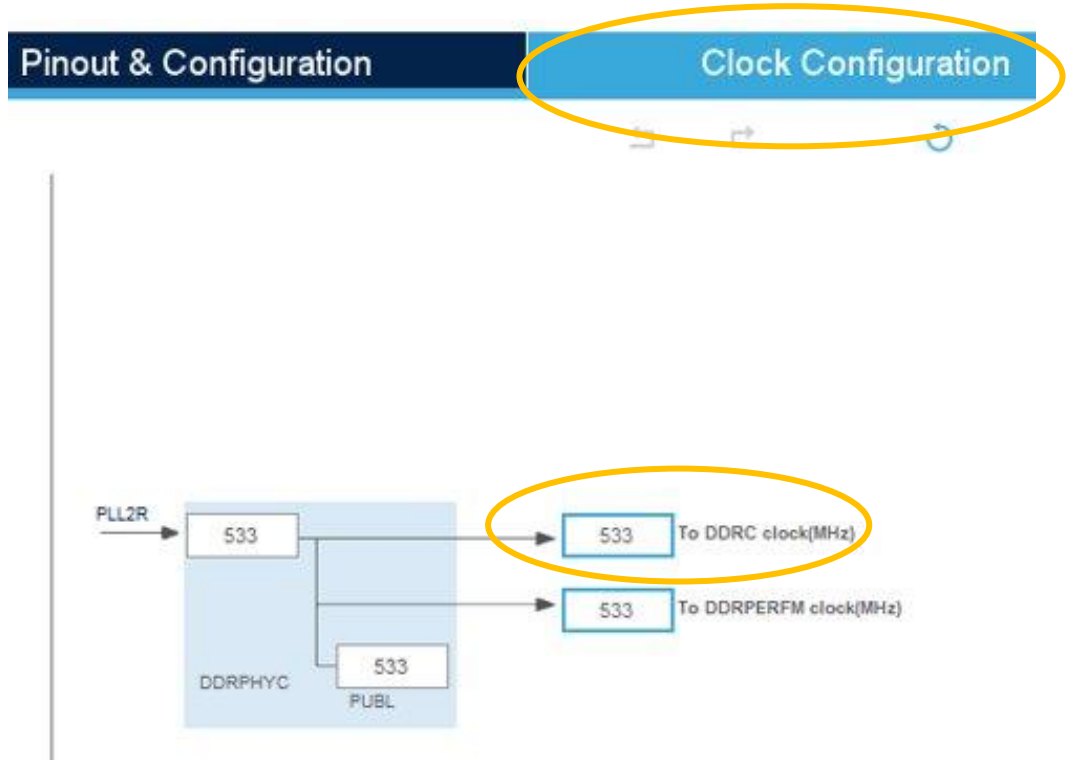
Customize to PCB

For new DDR memory type / frequency : create a new xxx-template.h with the help of STM32CubeMX

1 Open new project with STM32CubeMx



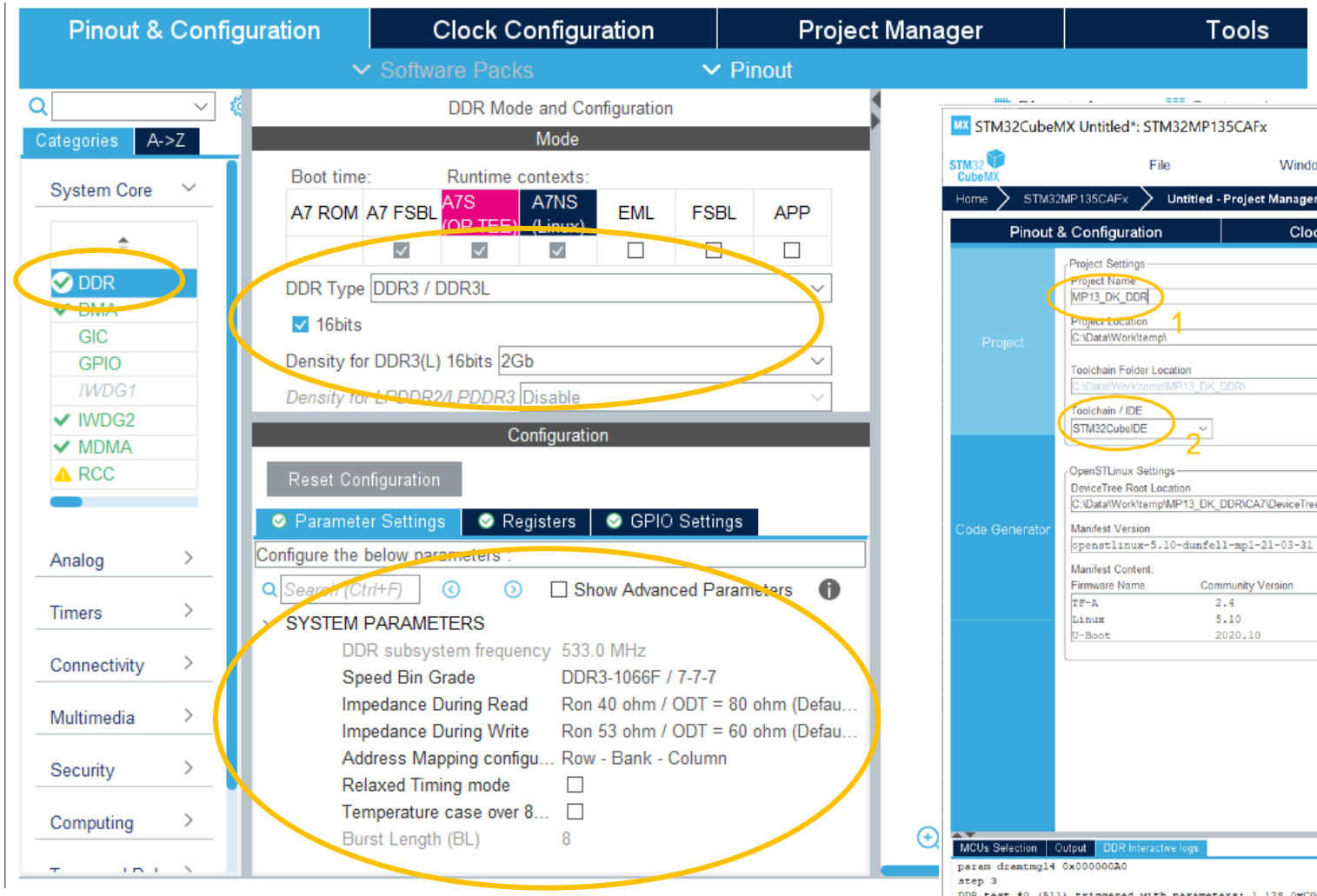
2 Set DDR frequency

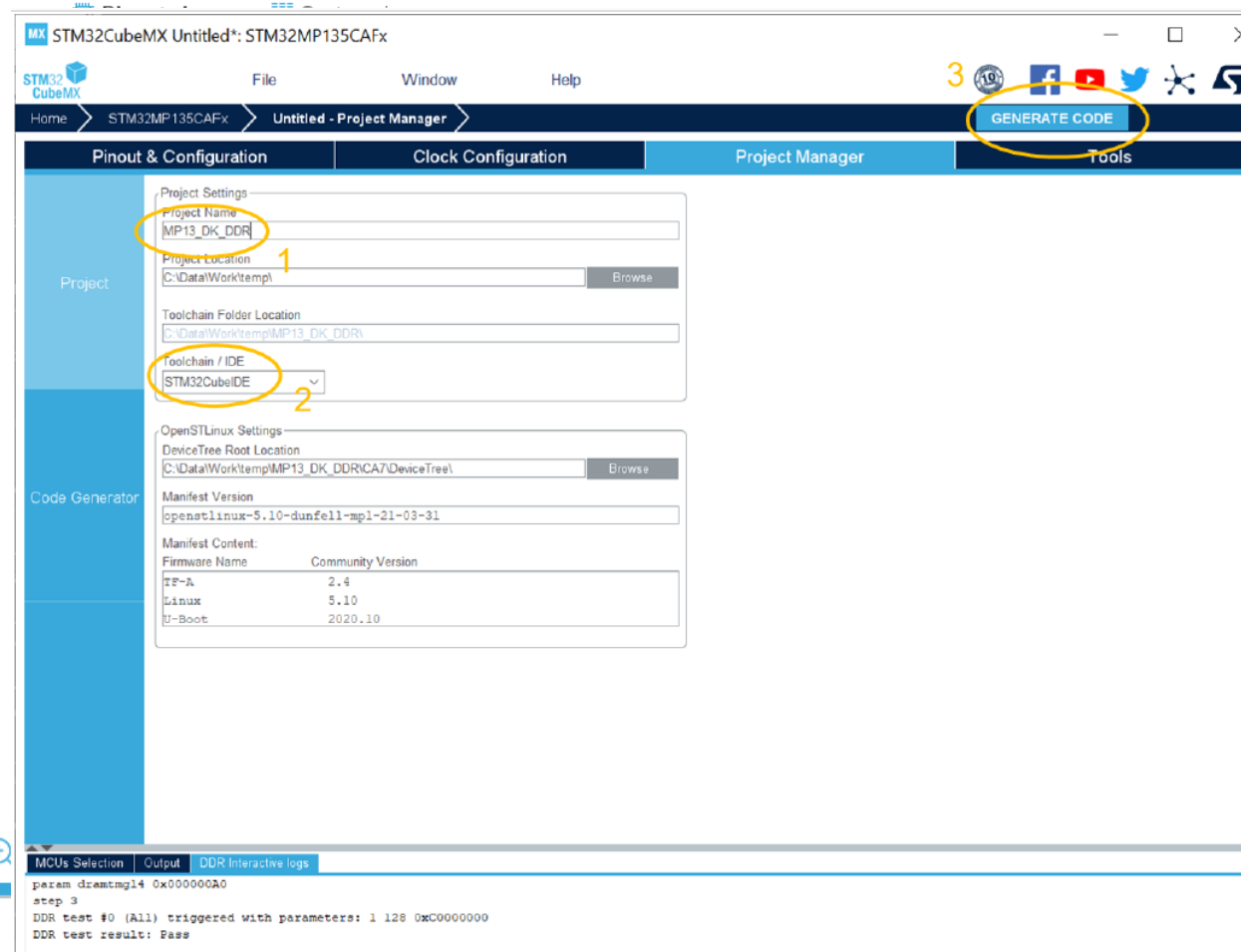


Customize to PCB

3 set DDR type, size, seed/grade, impedances

4 Generate device tree to compute the new DDR timings

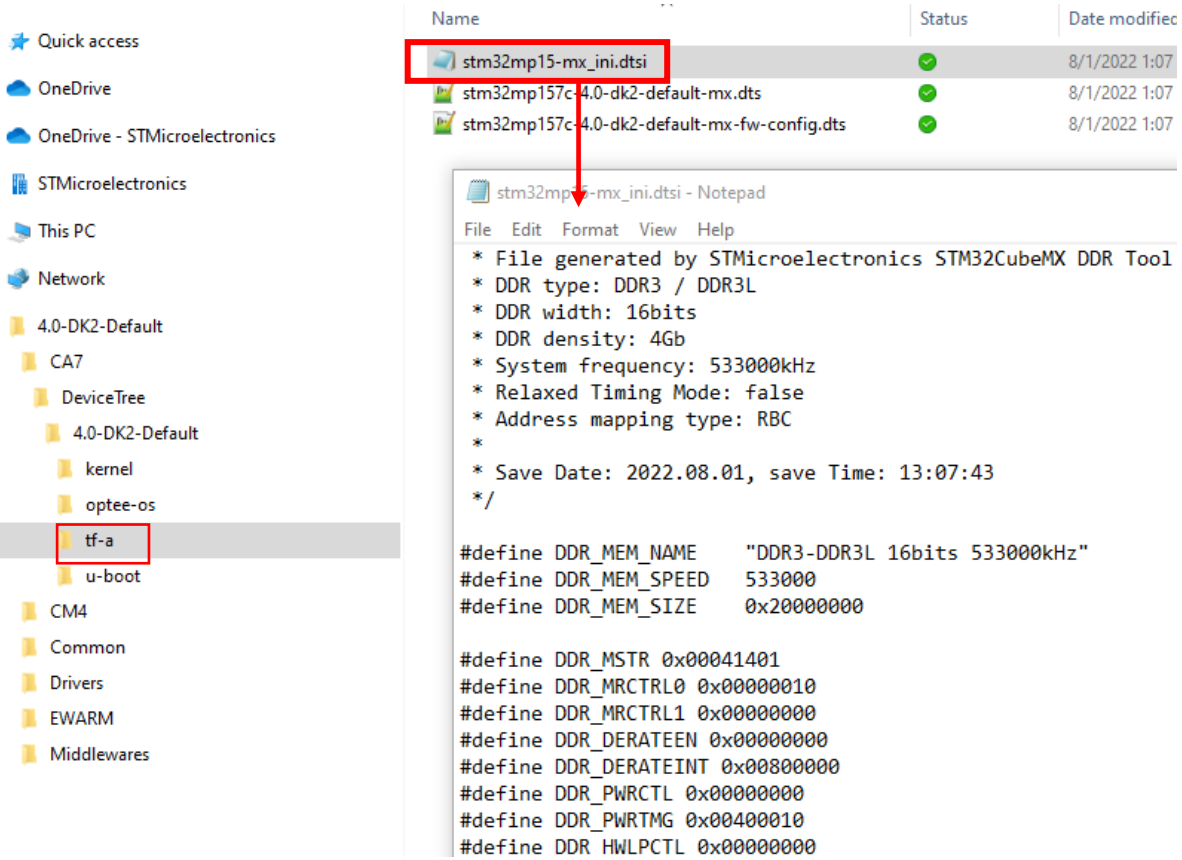




life.augmented

Customize to PCB

5 Copy DDR Setup .dtsi file -> new xxx-template.h file



6 report the PLL2 setting -> in stm32_util_conf.h