



# STM32MP15 backup registers



# STM32MP15 backup registers

Stable: 12.02.2019 - 08:18 / Revision: 06.12.2018 - 11:20

Template:ArticleMainWriter Template:ArticleApprovedVersion

## Contents

1 Article purpose .....	2
2 Overview .....	2
3 Backup registers usage .....	2
<b>3.1 At boot time .....</b>	<b>3</b>
<b>3.2 At runtime .....</b>	<b>3</b>
4 Memory mapping .....	4
5 References .....	8

## 1 Article purpose

The purpose of this article is to explain how the TAMP backup registers are used by STM32MPU Embedded Software.

## 2 Overview

The STM32MP15 embeds 32 backup registers of 32 bits. A programmable border allows to split those backup registers into a secure and a non-secure group.

By default, the ROM code defines the 10 first backup registers as secure, but this secure/non-secure border can be changed later on from the secure world.

## 3 Backup registers usage

This paragraph explains for which purpose some backup registers are used by the ROM code and STM32MPU Embedded Software distribution.

Then, the next chapter shows the backup register mapping used to fulfill those needs.



**It is important to notice that the backup registers are erased when a tamper detection occurs in TAMP internal peripheral**

## 3.1 At boot time

- **Non-secure backup registers** are used:
  - during a cold boot:
    - by U-Boot to initialize the **boot counter**, that should be reset later on by the application.
  - after a reset:
    - by U-Boot to get an eventual **forced boot mode** that was set before reset. This can be useful to set U-Boot in programmer mode after a reboot, for instance. Note that this **forced boot mode** is not interpreted by the ROM code.
    - by U-Boot to increment the **boot counter** and perform given actions if a predefined number of successive boots is reached, due to cyclic resets before the application is alive (and clears the counter).
- **Secure backup registers** are used:
  - to tell to the FSBL (TF-A or U-Boot SPL) how to behave:
    - on cold boot, the ROM code sets the **magic number** to 0x0: this value tells to the FSBL that a complete DDR initialization is needed before jumping to the SSBL (U-Boot).
    - on wakeup from Standby with DDR in self-refresh low power mode, if the **magic number** == 0xCA7FACE0 then the FSBL performs a partial DDR initialization to exit Self-Refresh then it branches the Arm<sup>®</sup> Cortex<sup>®</sup>-A7 core 0 non-secure execution to the given **branch address** (in Linux<sup>®</sup> kernel, that was set during secure context saving before the Standby low power mode entering).
  - by Linux<sup>®</sup> kernel on Arm<sup>®</sup> Cortex<sup>®</sup>-A7 core 0 (via a PSCI secure service) to tell to the ROM code how to start Arm<sup>®</sup> Cortex<sup>®</sup>-A7 core 1 (and enable the SMP mode): when Arm<sup>®</sup> Cortex<sup>®</sup>-A7 core 1 non-secure sees the **magic number** == 0xCA7FACE1 then it jumps to the given **branch address**.
  - by the ROM code during wakeup from Standby low power mode to recover the Cortex<sup>®</sup>-M4 firmware **integrity check value** and compare it to the one computed on RETRAM before starting the Cortex<sup>®</sup>-M4 again.

Notice: the ROM code knows if Cortex<sup>®</sup>-A7 and/or Cortex<sup>®</sup>-M4 have to be restarted after Standby thanks to RCC\_MP\_BOOTCR register, so the backup registers are not used here.

## 3.2 At runtime

- Non secure backup registers
  - own the **boot counter** and should be reset by the application after a successful startup.
  - are used to store Cortex<sup>®</sup>-M4 retention firmware **integrity check value** before going to Standby mode, if the Cortex<sup>®</sup>-M4 needs to be started on wakeup from Standby mode by the ROM code.
- Secure backup registers
  - are used by secure services to store:
    - Arm<sup>®</sup> Cortex<sup>®</sup>-A7 core 0 **branch address** that are used by the ROM code on wakeup from Standby mode.
    - Arm<sup>®</sup> Cortex<sup>®</sup>-M4 **security perimeter** that is restored by the ROM code before starting the Cortex<sup>®</sup>-M4 on wakeup from Standby.



## 4 Memory mapping

The table below shows the backup register mapping used by STM32MPU Embedded Software.  
The TAMP backup register base address is 0x5C00A100, corresponding to TAMP\_BKP0R.

TAMP register	Security	ROM / software register name	Comment
TAMP_BKP31R	No non-secure		
TAMP_BKP30R	No non-secure		
TAMP_BKP29R	No non-secure		
TAMP_BKP28R	No non-secure		
TAMP_BKP27R	No non-secure	BACKUP_M4_WAKEUP_ARE_A_HASH	SHA-256 integrity check value computed on RETRAM by Linux remoteproc during the coprocessor firmware loading and checked by the ROM code on wakeup from Standby before starting the coprocessor
TAMP_BKP26R	No non-secure		



STM32MP15 backup registers

TAM P regist er	Sec urity	ROM / software register name	Comment
TA MP _BK P25 R	No n- se cu re		
TA MP _BK P24 R	No n- se cu re		
TA MP _BK P23 R	No n- se cu re	BACKUP_M4_ WAKEUP_ARE A_LENGTH	Amount of bytes hashed in <b>RETRAM</b> to compute the integrity check value
TA MP _BK P22 R	No n- se cu re	BACKUP_M4_ WAKEUP_ARE A_START	Start address in <b>RETRAM</b> from where the integrity check value has to be computed
TA MP _BK P21 R	No n- se cu re	BACKUP_BOO T_COUNTER	Boot counter
TA MP _BK P20 R	No n- se cu re	BACKUP_BOO T_MODE <sup>[1]</sup>	Boot mode context information
TA MP _BK P19 R	No n- se cu re		
TA MP	No n-		



STM32MP15 backup registers

TAMP register	Security	ROM / software register name	Comment
_BK P18 R	secure		
TAMP _BK P17 R	Non-secure		(Reserved for future use)
TAMP _BK P16 R	Non-secure		(Reserved for future use)
TAMP _BK P15 R	Non-secure		(Reserved for future use)
TAMP _BK P14 R	Non-secure		(Reserved for future use)
TAMP _BK P13 R	Non-secure		(Reserved for future use)
TAMP _BK P12 R	Non-secure		(Reserved for future use)
TAMP _BK P11	Non-secure		



STM32MP15 backup registers

TAMP register	Security	ROM / software register name	Comment
R	re		(Reserved for future use)
TAMP10R	Non-secure		(Reserved for future use)
TAMP9R	Secure		(Reserved for future use)
TAMP8R	Secure		(Reserved for future use)
TAMP7R	Secure		(Reserved for future use)
TAMP6R	Secure		(Reserved for future use)
TAMP5R	Secure	BACKUP_BRANCH_ADDRESSES <sup>[1]</sup>	CPU0 or CPU1 branch address
TAMP4R	Secure	BACKUP_MAGIC_NUMBER <sup>[1]</sup>	CPU0 or CPU1 boot magic number
TAMP3R	Secure	BACKUP_M4_SECURITY_PERIMETER_EXTENSION3	Value of AEIC TZENR3
TAMP	Secure	BACKUP_M4_SECURITY_PE	



TAMP register	Security	ROM / software register name	Comment
_BK_P2R	re	RIMETER_EXT_I2	Value of AEIC TZENR2
TAMP_BK_P1R	Secure	BACKUP_M4_SECURITY_PERIMETER_EXT_I1	Value of AEIC TZENR1
TAMP_BK_P0R	Secure	BACKUP_WAKEUP_SEC	Wakeup parameters

## 5 References

- 1.01.11.2 [arch/arm/mach-stm32mp/include/mach/stm32.h](#)

Das U-Boot -- the Universal Boot Loader (see [U-Boot\\_overview](#))

First Stage Boot Loader

Second Stage Boot Loader

Doubledata rate (memory domain)

Power State Coordination Interface

symetric multiprocessing

Tamper

Secure Hash Algorithm