

STM32MP15 backup registers

Stable: 12.02.2019 - 09:18 / Revision: 06.12.2018 - 12:20

Contents

| | |
|--------------------------------|---|
| 1 Article purpose | 1 |
| 2 Overview | 1 |
| 3 Backup registers usage | 1 |
| 3.1 At boot time | 1 |
| 3.2 At runtime | 2 |
| 4 Memory mapping | 2 |
| 5 References | 4 |

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[®] Cortex[®]-A7 core 0 non-secure execution to the given **branch address** (in Linux[®] kernel, that was set during secure context saving before the Standby [low power mode](#) entering).
 - by Linux[®] kernel on Arm[®] Cortex[®]-A7 core 0 (via a PSCI secure service) to tell to the [ROM](#) code how to start Arm[®] Cortex[®]-A7 core 1 (and enable the SMP mode): when Arm[®] Cortex[®]-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[®]-M4 firmware **integrity check value** and compare it to the one computed on [RETRAM](#) before starting the Cortex[®]-M4 again.

Notice: the [ROM](#) code knows if Cortex[®]-A7 and/or Cortex[®]-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[®]-M4 retention firmware **integrity check value** before going to Standby mode, if the Cortex[®]-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[®] Cortex[®]-A7 core 0 **branch address** that are used by the [ROM](#) code on wakeup from Standby mode.
 - Arm[®] Cortex[®]-M4 **security perimeter** that is restored by the [ROM](#) code before starting the Cortex[®]-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.

ROM / software

| TAMP register | Security | register name | Comment |
|----------------------|-----------------|--|--|
| TAMP_BKP31R | Non-secure | | SHA-256 integrity check |
| TAMP_BKP30R | Non-secure | | value computed on RETRAM |
| TAMP_BKP29R | Non-secure | | by Linux remoteproc |
| TAMP_BKP28R | Non-secure | BACKUP_M4_WAKEUP_AREAS_HASH | during the coprocessor |
| TAMP_BKP27R | Non-secure | | firmware loading and |
| TAMP_BKP26R | Non-secure | | checked by the ROM |
| TAMP_BKP25R | Non-secure | | code on wakeup from |
| TAMP_BKP24R | Non-secure | | Standby before starting |
| | | | the coprocessor |
| TAMP_BKP23R | Non-secure | BACKUP_M4_WAKEUP_AREAS_LENGTH | Amount of bytes hashed |
| | | | in RETRAM to compute |
| | | | the integrity check value |
| | | | Start address in RETRAM |
| TAMP_BKP22R | Non-secure | BACKUP_M4_WAKEUP_AREAS_START | from where the integrity |
| | | | check value has to be |
| | | | computed |
| TAMP_BKP21R | Non-secure | BACKUP_BOOT_COUNTER | Boot counter |
| TAMP_BKP20R | Non-secure | BACKUP_BOOT_MODE ^[1] | Boot mode context |
| | | | information |
| TAMP_BKP19R | Non-secure | | (Reserved for future use) |
| TAMP_BKP18R | Non-secure | | (Reserved for future use) |
| TAMP_BKP17R | Non-secure | | (Reserved for future use) |
| TAMP_BKP16R | Non-secure | | (Reserved for future use) |
| TAMP_BKP15R | Non-secure | | (Reserved for future use) |
| TAMP_BKP14R | Non-secure | | (Reserved for future use) |
| TAMP_BKP13R | Non-secure | | (Reserved for future use) |
| TAMP_BKP12R | Non-secure | | (Reserved for future use) |
| TAMP_BKP11R | Non-secure | | (Reserved for future use) |
| TAMP_BKP10R | Non-secure | | (Reserved for future use) |
| TAMP_BKP9R | Secure | | (Reserved for future use) |
| TAMP_BKP8R | Secure | | (Reserved for future use) |
| TAMP_BKP7R | Secure | | (Reserved for future use) |
| TAMP_BKP6R | Secure | | (Reserved for future use) |
| TAMP_BKP5R | Secure | BACKUP_BRANCH_ADDRESSES ^[1] | CPU0 or CPU1 branch |
| | | | address |
| TAMP_BKP4R | Secure | BACKUP_MAGIC_NUMBER ^[1] | CPU0 or CPU1 boot magic |
| | | | number |
| TAMP_BKP3R | Secure | BACKUP_M4_SECURITY_PERIMETER_EXTI3 | Value of AEIC TZENR3 |
| TAMP_BKP2R | Secure | BACKUP_M4_SECURITY_PERIMETER_EXTI2 | Value of AEIC TZENR2 |
| TAMP_BKP1R | Secure | BACKUP_M4_SECURITY_PERIMETER_EXTI1 | Value of AEIC TZENR1 |

TAMP_BKP0R

Secure

BACKUP_WAKEUP_SEC

Wakeup parameters

5 References

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

First Stage Boot Loader

Second Stage Boot Loader

Doubledata rate (memory domain)

Power State Coordination Interface

symetric multiprocessing

Tamper

Secure Hash Algorithm