mind
embedded development
an ESSENSIUM division

# Safe upgrade of embedded systems

*Arnout Vandecappelle*

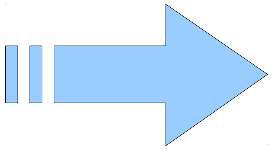
ESSENSIUM

http://mind.be/content/Presentation_Safe-Upgrade.pdf or .odp

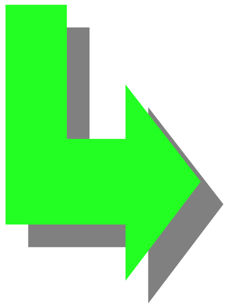
# You never know where your product will be used

High-precision GNSS receiver

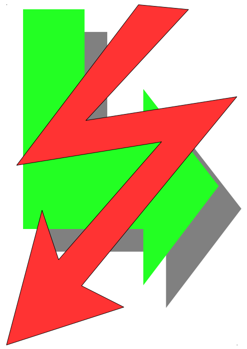
## You never know where your product will be used



# What if you install new firmware on remote systems?



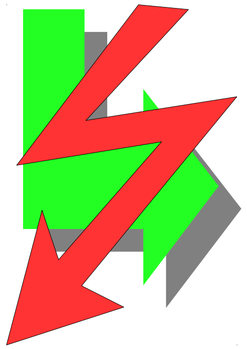
# What if you install new firmware on remote systems?



Murphy's Law

# What if you install new firmware on remote systems?

Murphy's Law



# Safe upgrade of embedded systems

*Arnout Vandecappelle*

http://mind.be/content/Presentation_Safe-Upgrade.pdf or .odp

## Overview

1 Failure mechanisms

- Power failure
- Bad firmware
- Flash corruption
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

## Power failure

Power fails *during* upgrade

⇒ new firmware only partially written

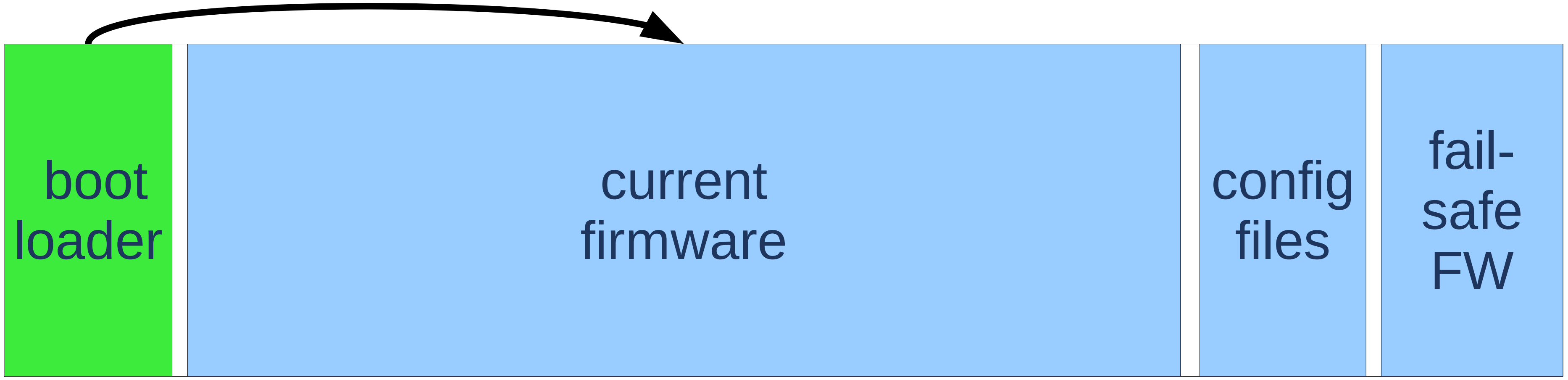
Solutions:

- Add *fail-safe* firmware
- *Detect* failed power
- *Atomic* update of firmware images

- Use journalling filesystem for writable data

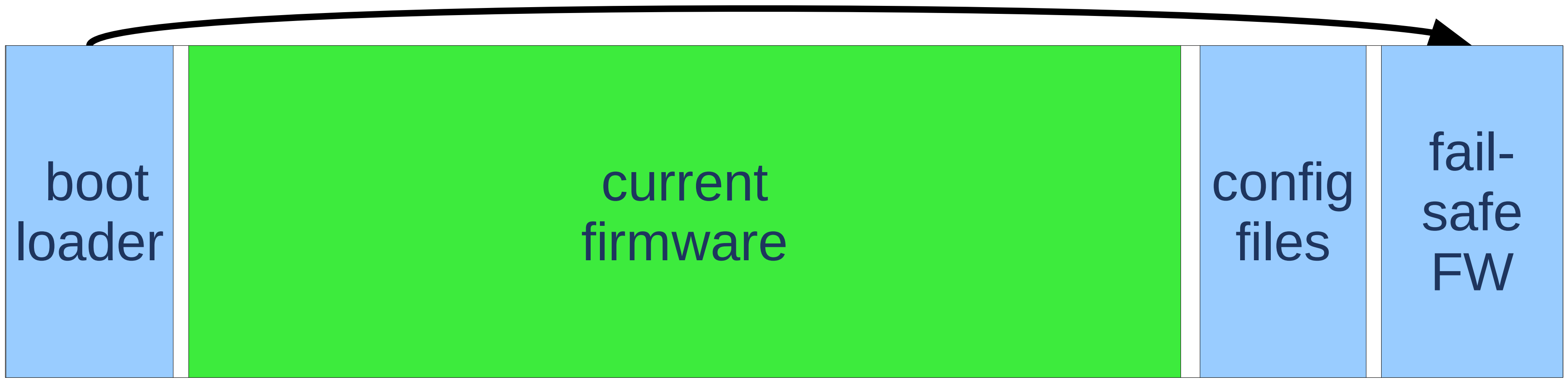
## Detecting power failure: Switch to fail-safe firmware

1. Boot current firmware

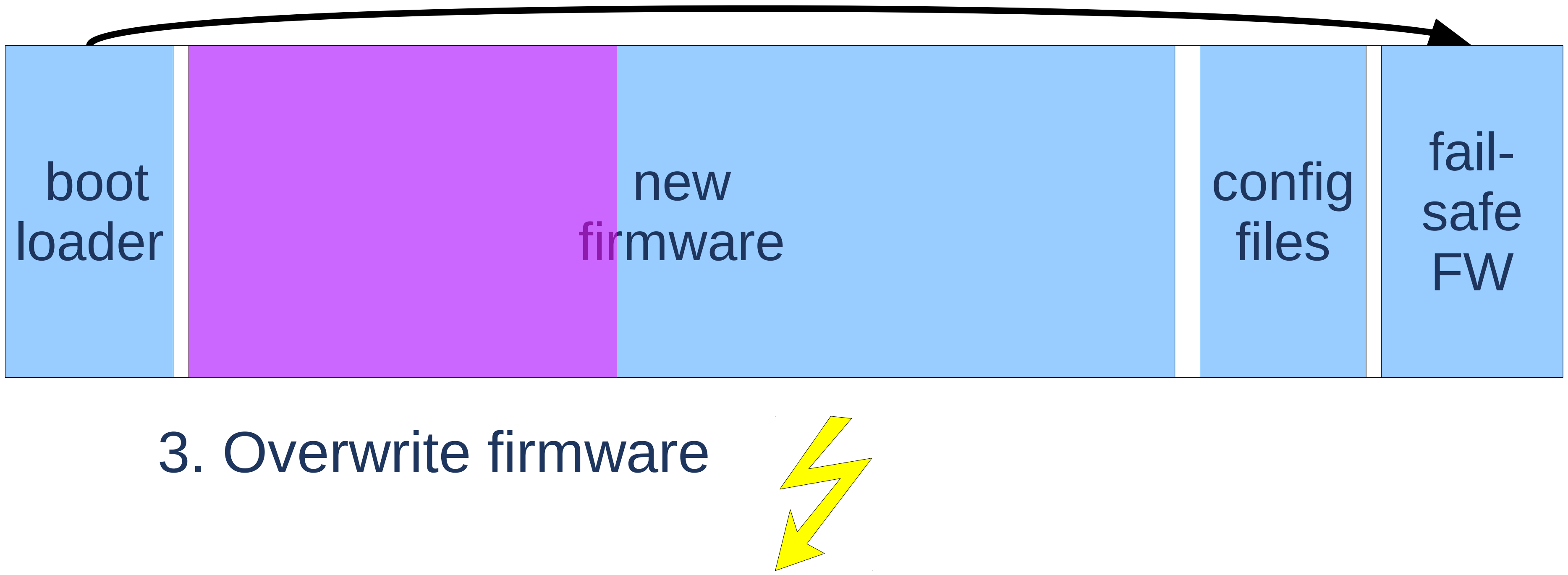


## Detecting power failure: Switch to fail-safe firmware

2. Switch to fail-safe



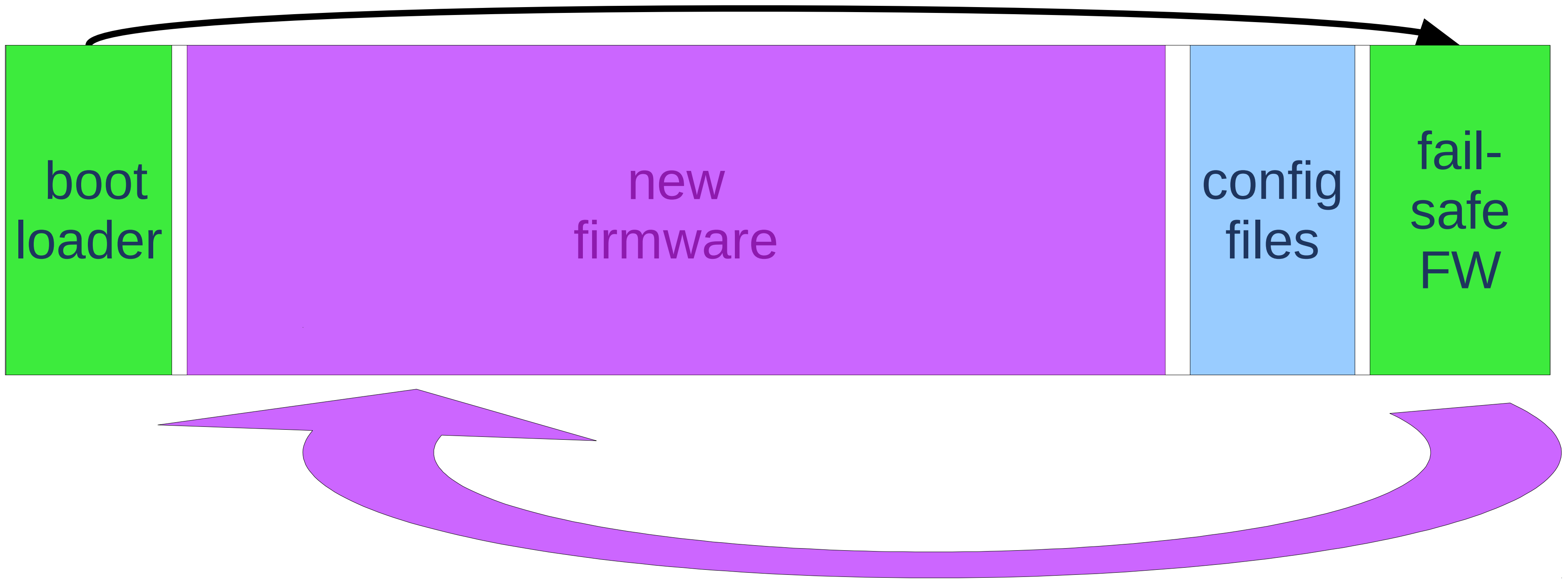
## Detecting power failure: Switch to fail-safe firmware



3. Overwrite firmware

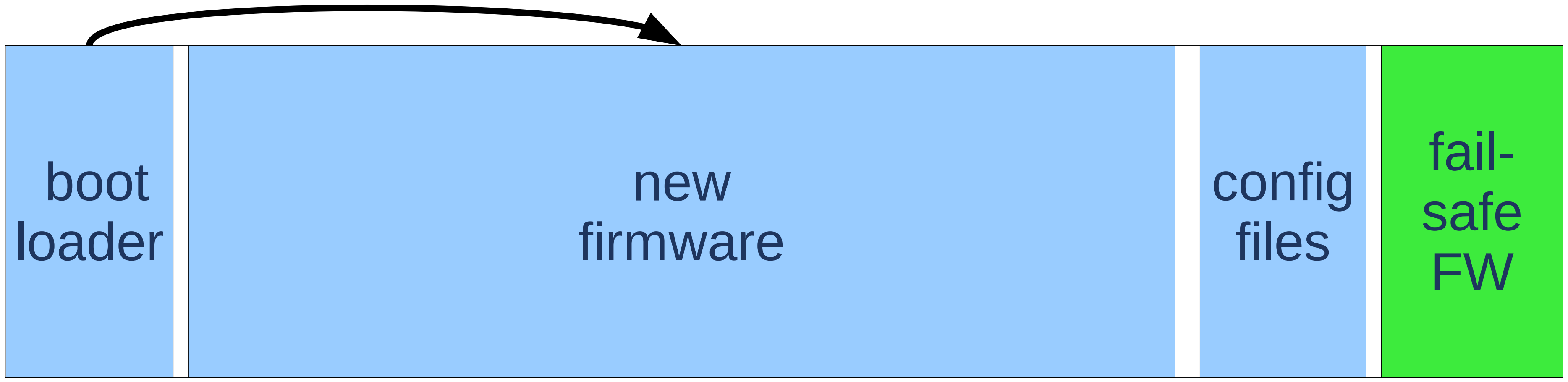
## Detecting power failure: Switch to fail-safe firmware

4. Fail-safe restarts upgrade



## Detecting power failure: Switch to fail-safe firmware

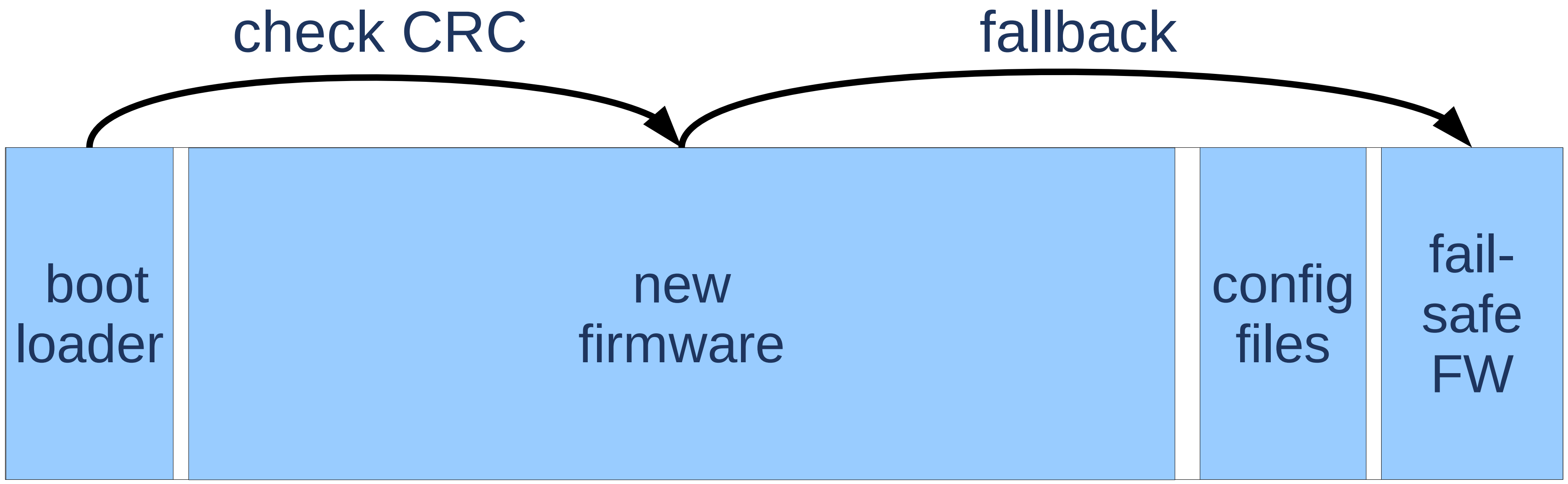
5. back to new firmware



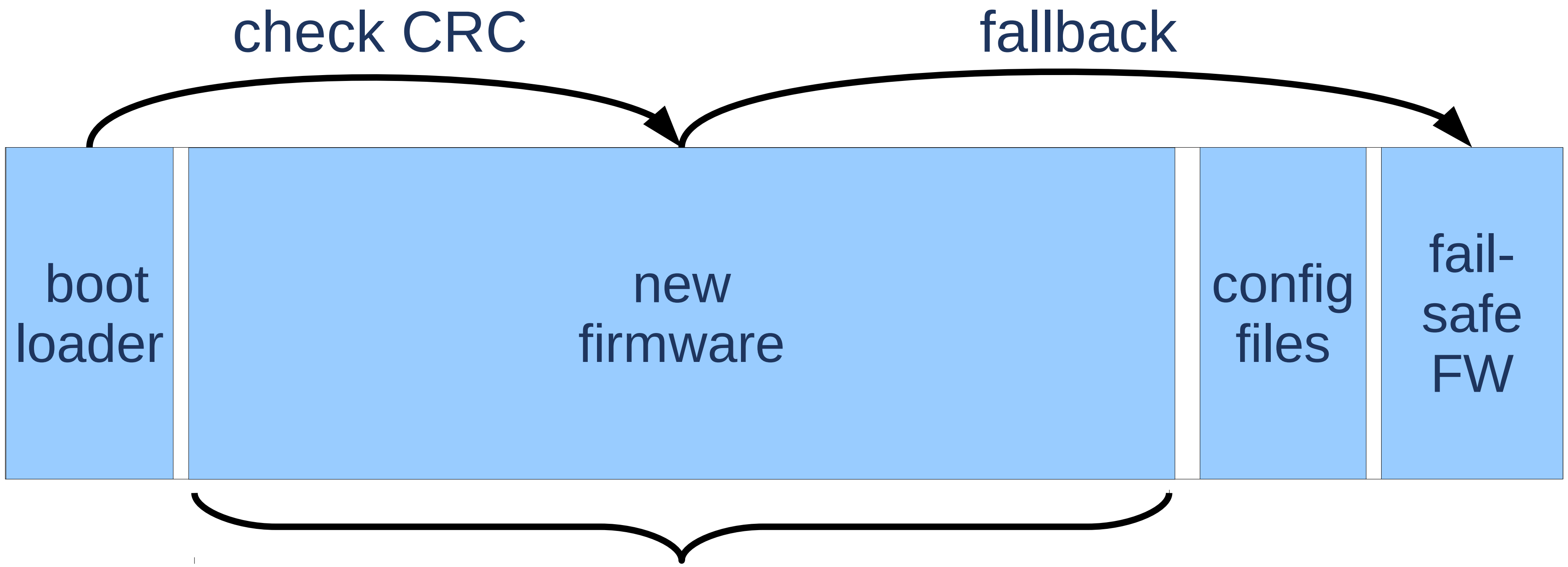
## Can bootloader switch to fail-safe *atomically*?

- Grub, extlinux
  - Overwrite a file
  - ⇒ Make sure overwrite is atomic, using rename(2)
  - ⇒ Relies on atomicity of underlying filesystem implementation
    e.g. ext4: mount with `barrier=1`
- U-Boot
  - Overwrite environment
  - ⇒ Catastrophic if power fails during environment write
- Use CRC to validate new image

# Detecting power failure: CRC check

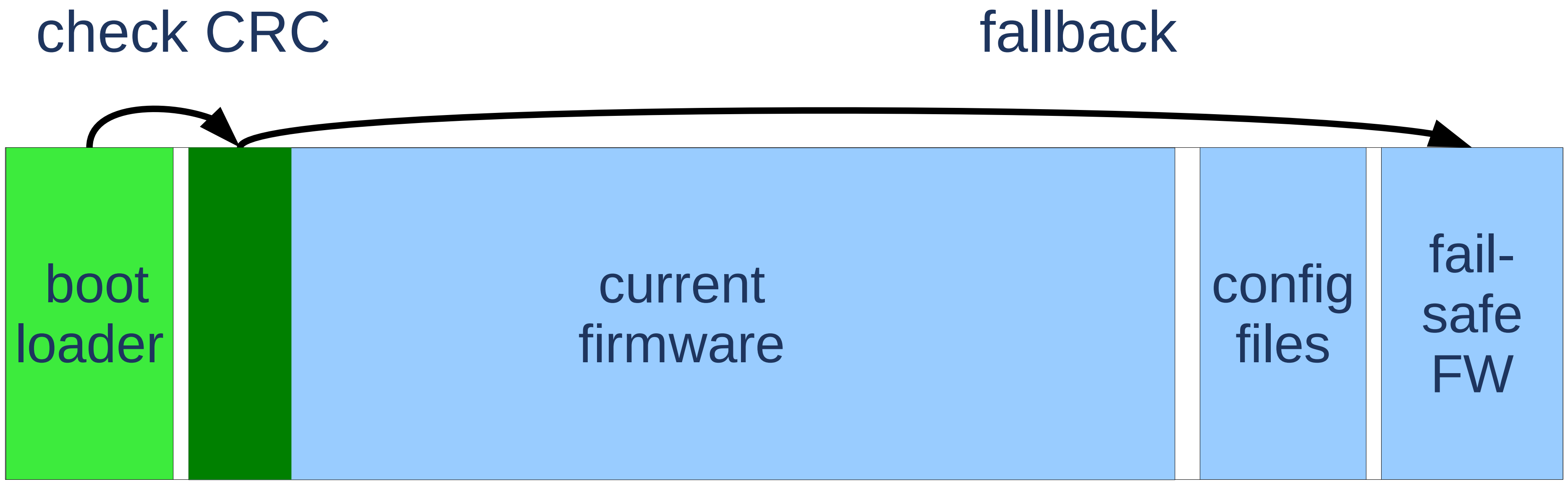


## Detecting power failure: CRC check doesn't (always) work

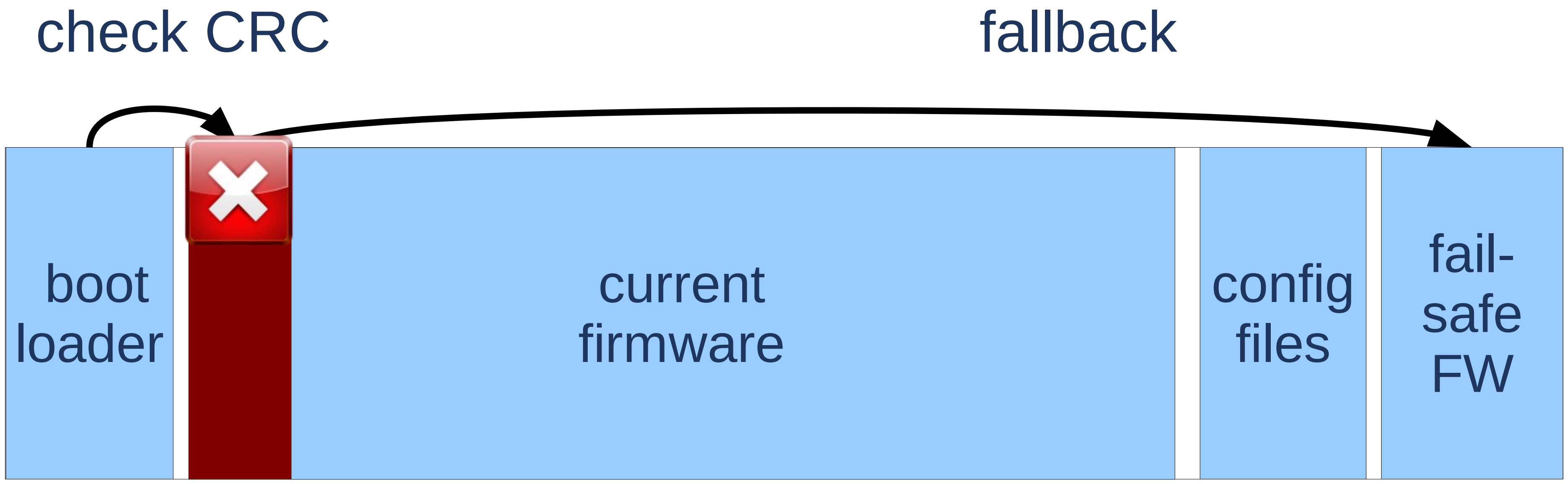


60MB ⇒ several minutes to check

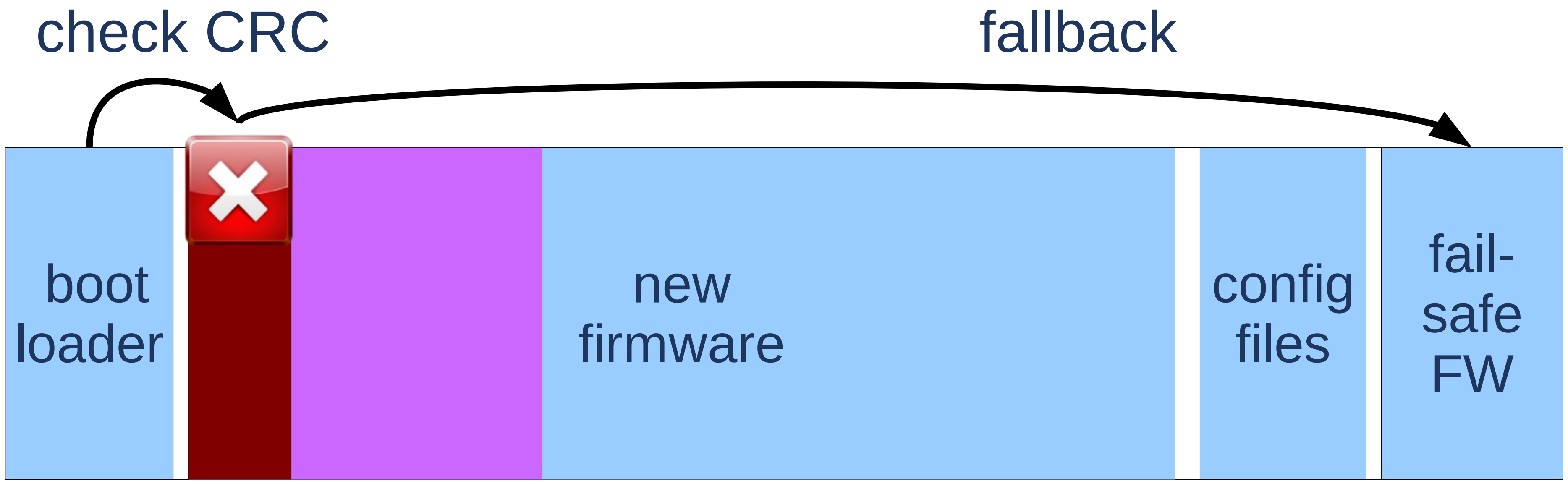
## Detecting power failure: CRC check of header only



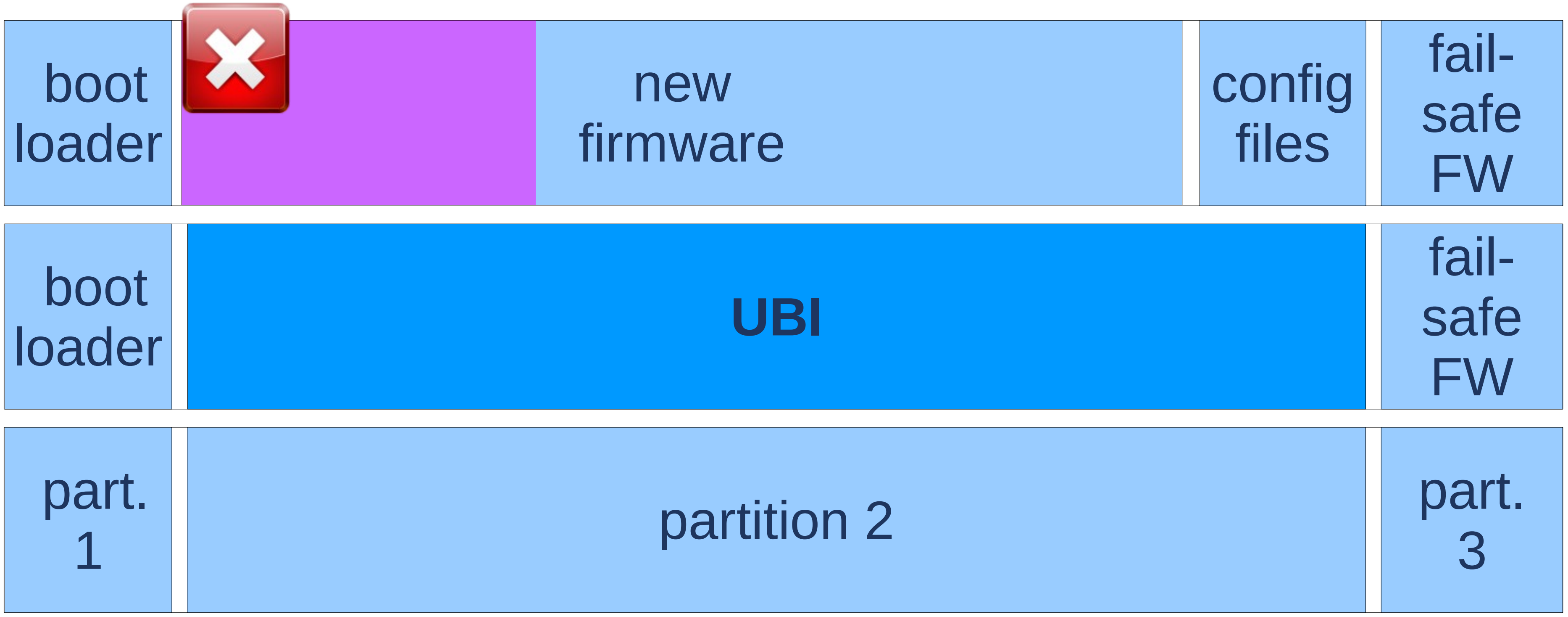
## Detecting power failure with CRC: Write images non-linearly



# Detecting power failure with CRC: Write images non-linearly



## UBI provides NAND-aware atomic updates



MTD device (NAND Flash)

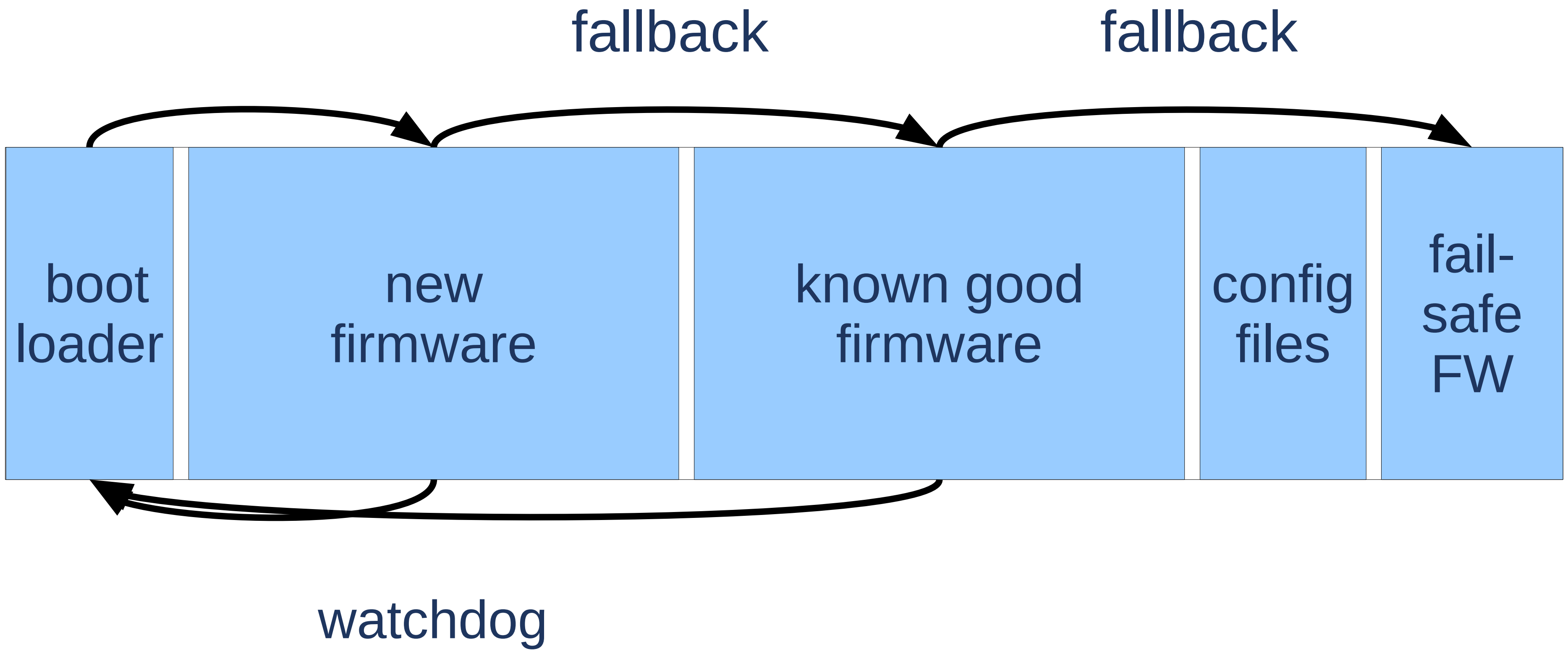
## Bad firmware

New firmware fails on some devices

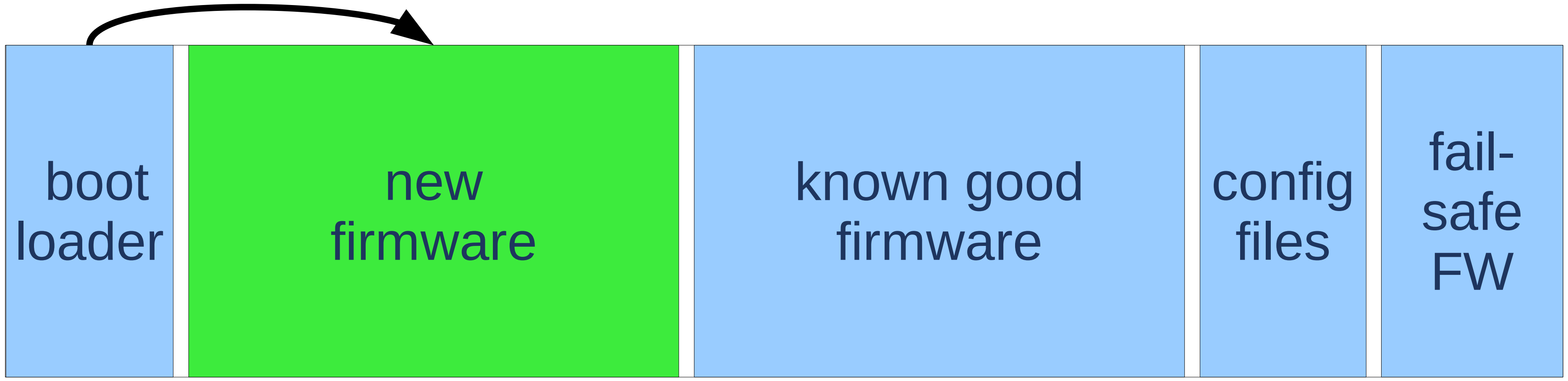
Solutions:

- Fall back on previous (known good) firmware
- Fail-safe firmware that can do upgrades
- Upgrade script included in upgrade image
- Watchdog reboot +
  boot fail-safe after bad boot

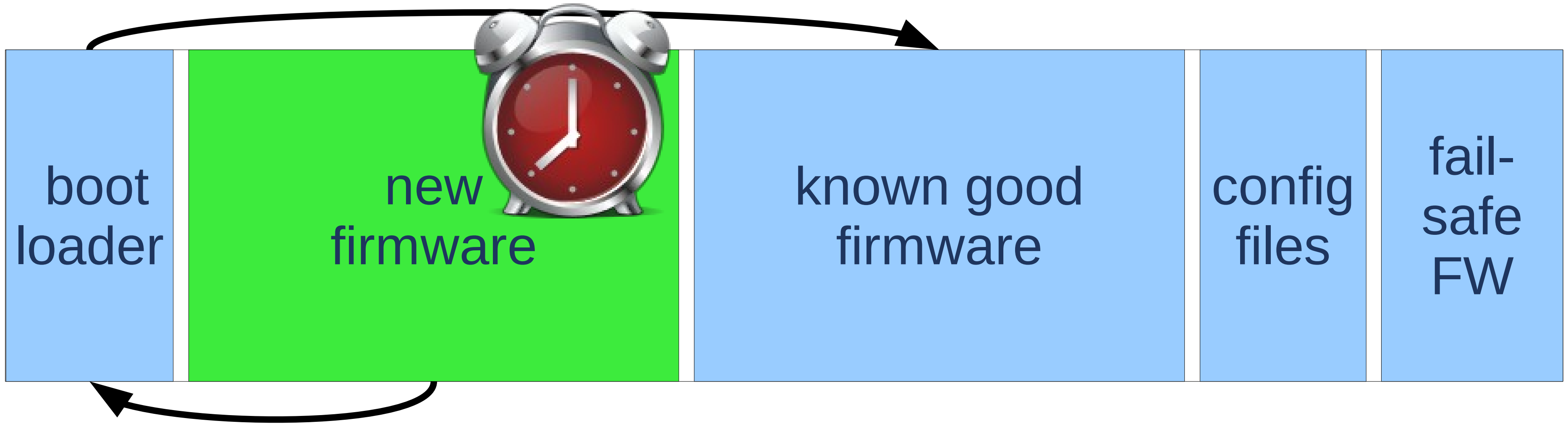
# Typical flash layout with known good and fail-safe firmware



# Boot procedure with watchdog



## Boot procedure with watchdog



Reboot when watchdog timer expires
Reset watchdog if firmware runs well
Force reboot if firmware does not run well

## Overview

1 Failure mechanisms

- Bad firmware
- Power failure
- Flash corruption
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

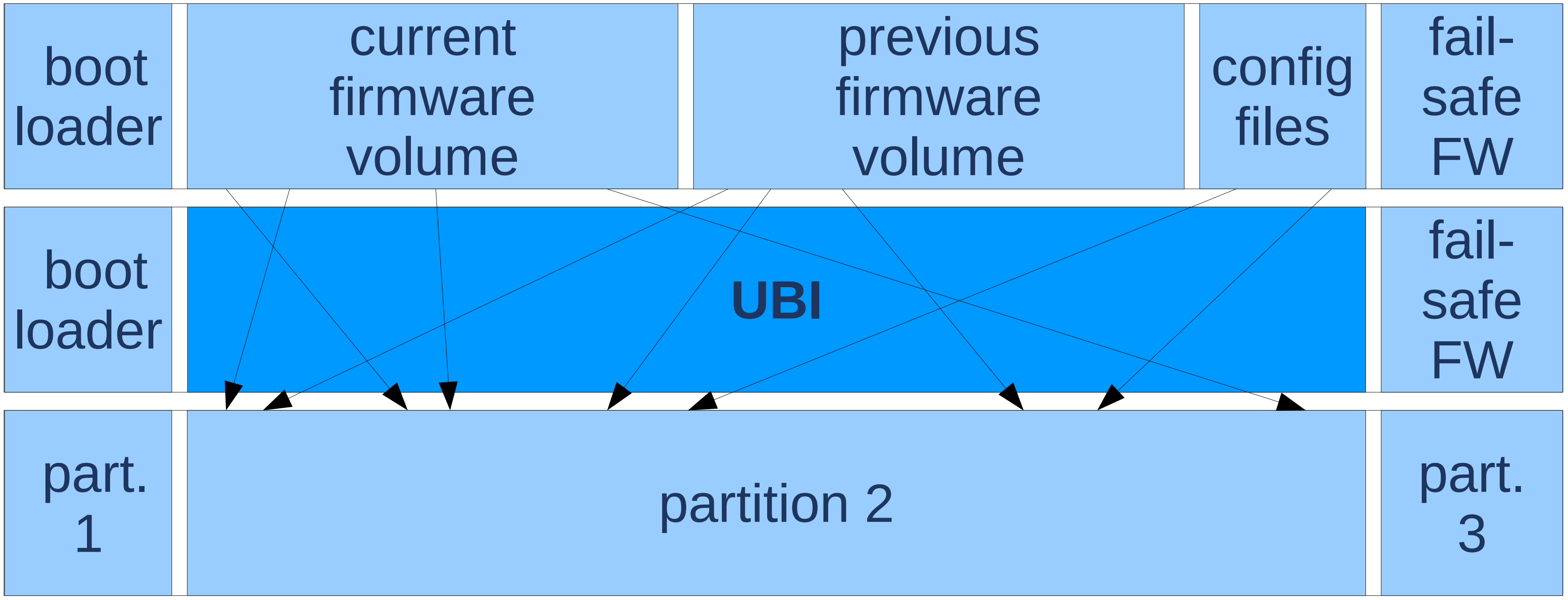
## Flash corruption

Flash storage is **unreliable**:
each individual bit becomes unusable after N writes

- Error correcting codes (ECC):
  detect & correct bit errors
  *when reading*
- Wear levelling:
  don't reuse the same block all the time
- Bad blocks:
  stop using a block if too many errors

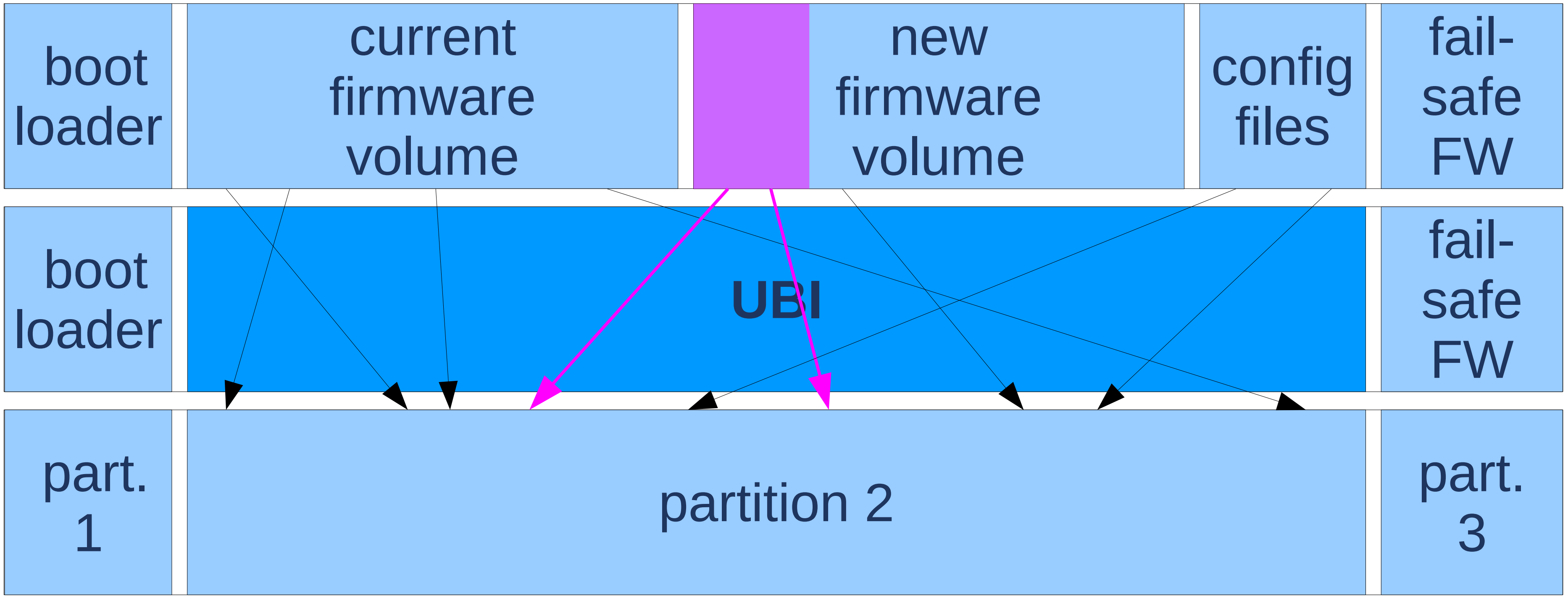
Flash **filesystem** must handle these problems

## UBI provides safe NAND writing



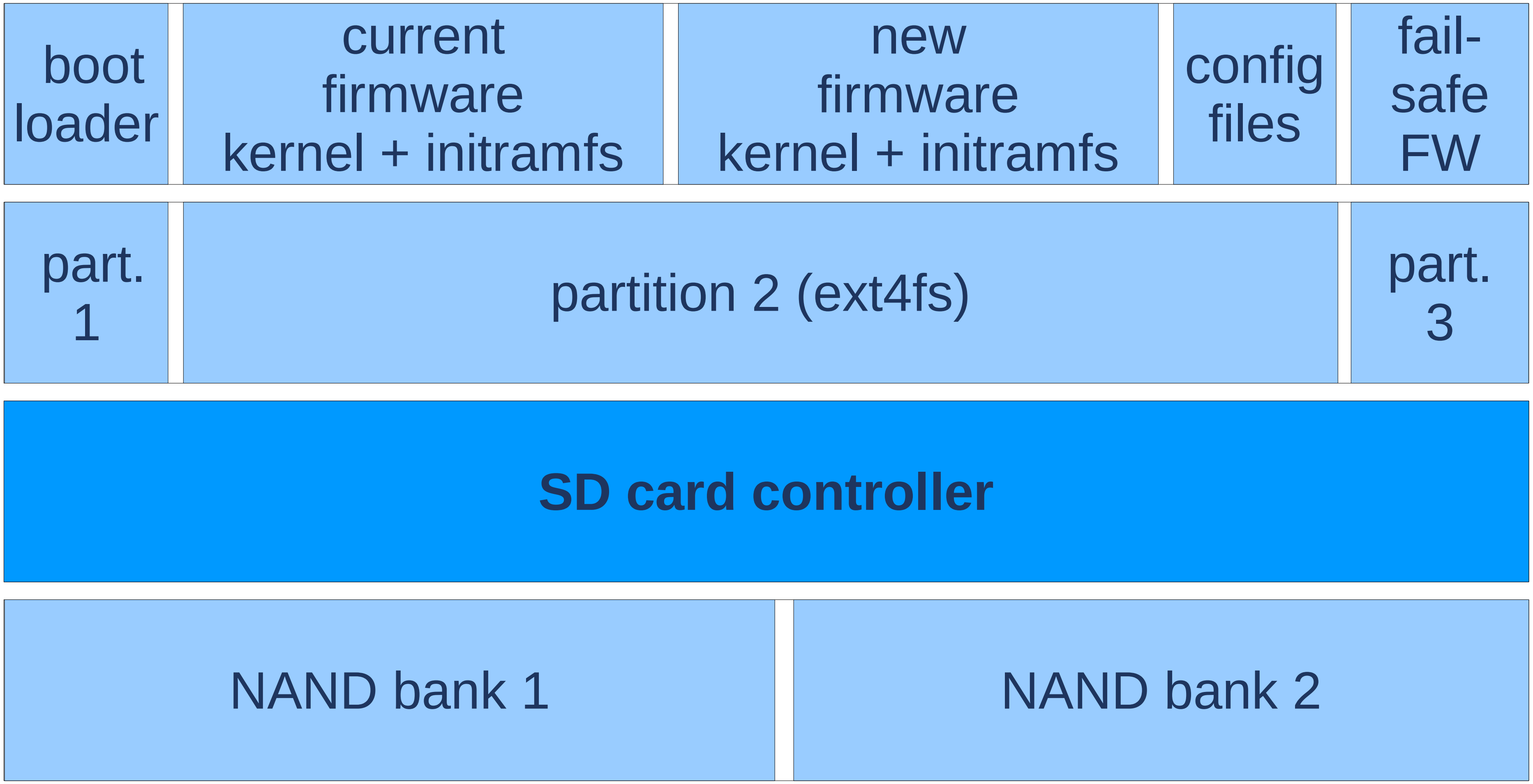
MTD device (NAND Flash)

# UBI provides safe upgrade

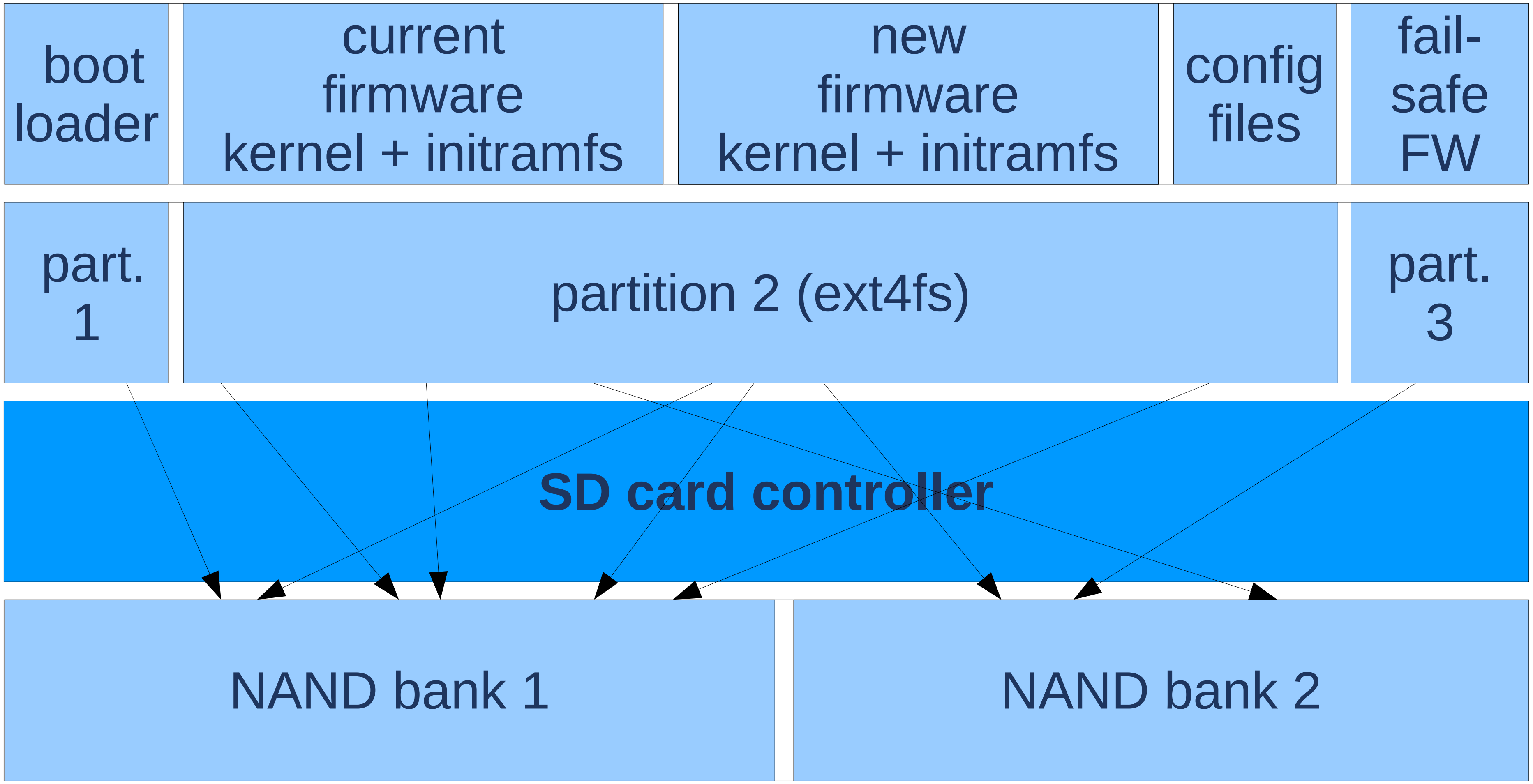


MTD device (NAND Flash)

## Intermezzo: SD cards etc. are bad news

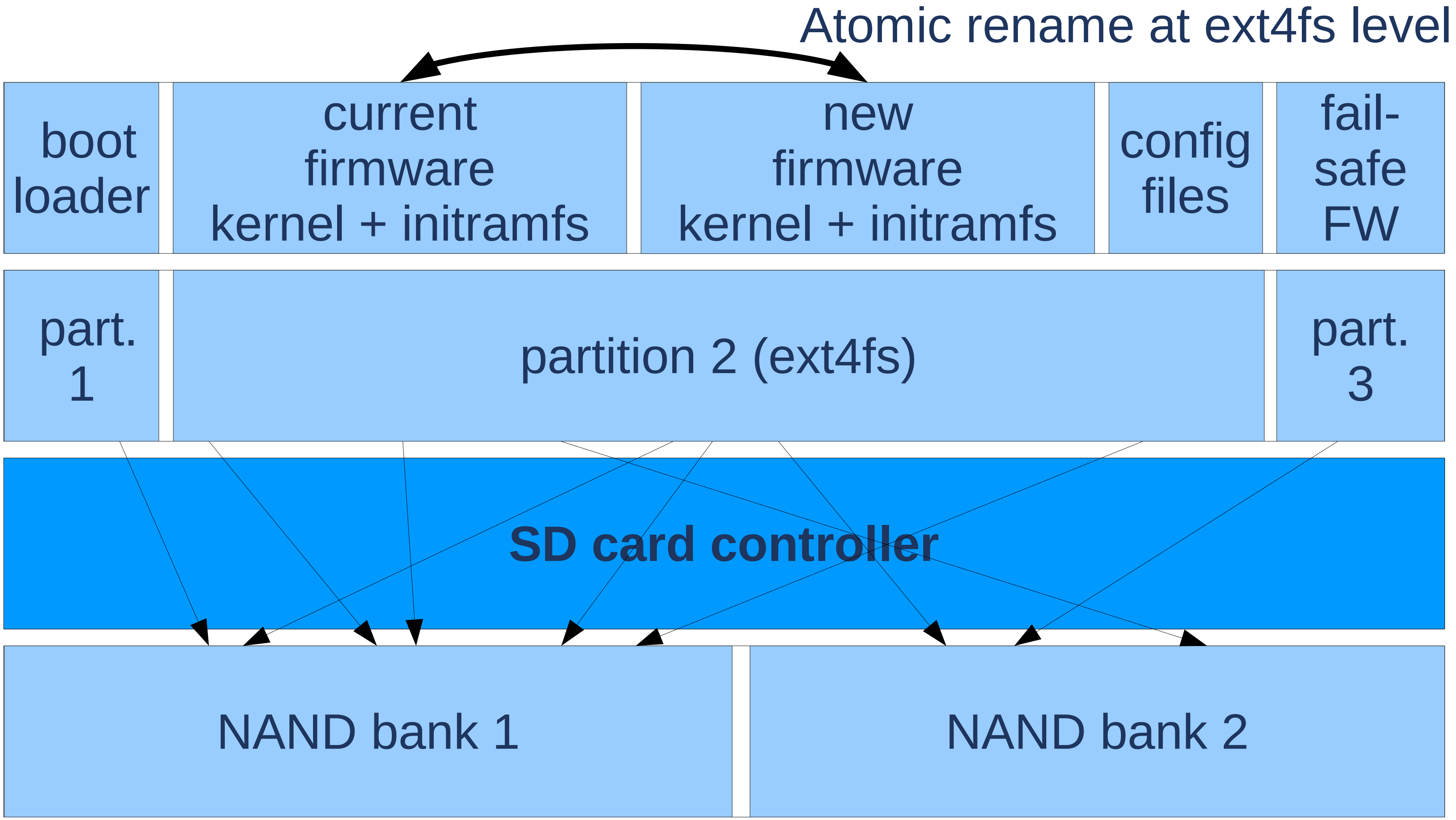


## Intermezzo: SD cards etc. are bad news



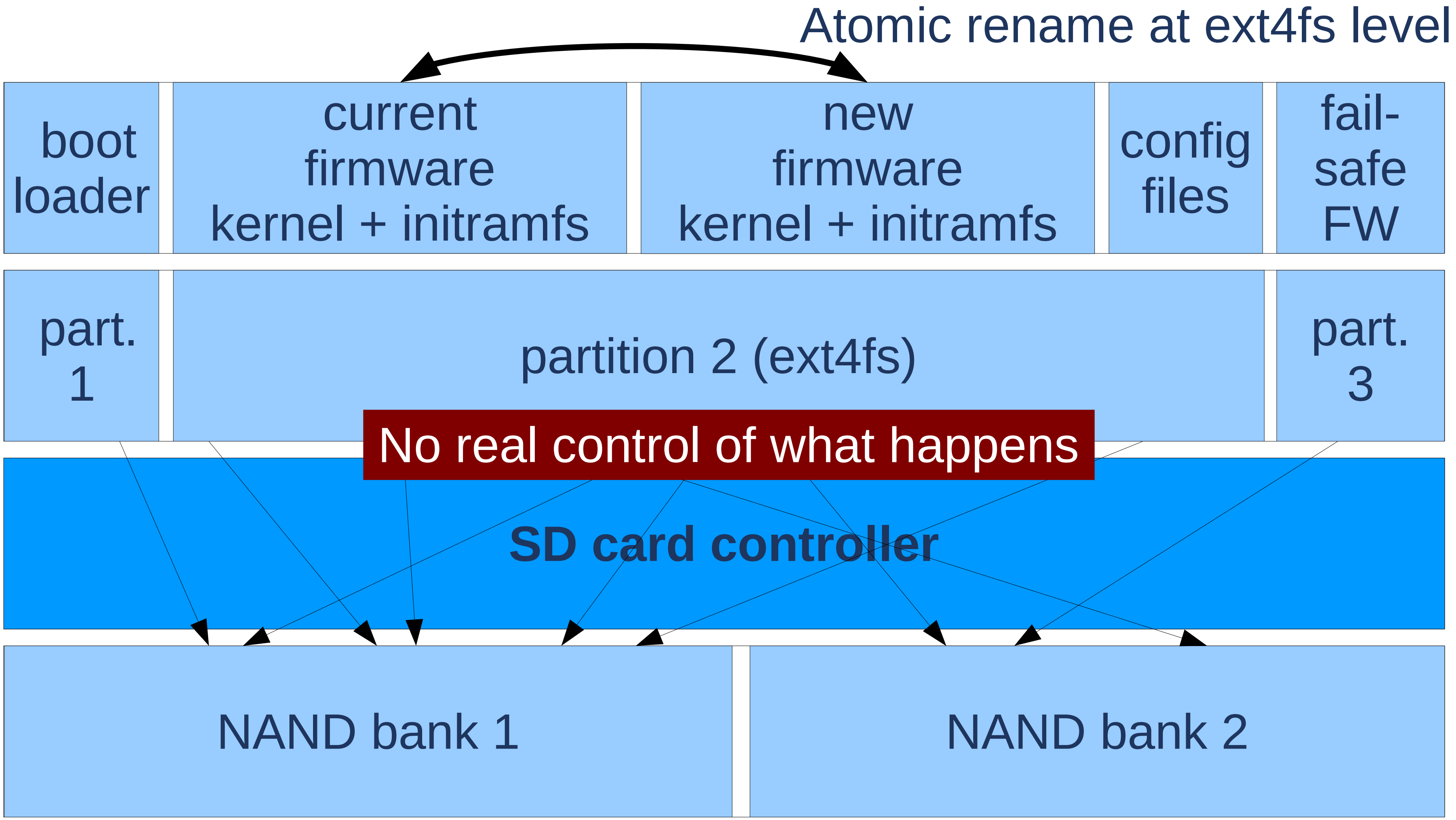
See http://elinux.org/images/4/49/Elc2011_bergmann.pdf

# Intermezzo: SD cards etc. are bad news



See http://elinux.org/images/4/49/Elc2011_bergmann.pdf

## Intermezzo: SD cards etc. are bad news


Atomic rename at ext4fs level
boot loader
current firmware kernel + initramfs
new firmware kernel + initramfs
config files
fail-safe FW
part. 1
partition 2 (ext4fs)
part. 3
No real control of what happens
SD card controller
NAND bank 1
NAND bank 2

See http://elinux.org/images/4/49/Elc2011_bergmann.pdf

## Overview

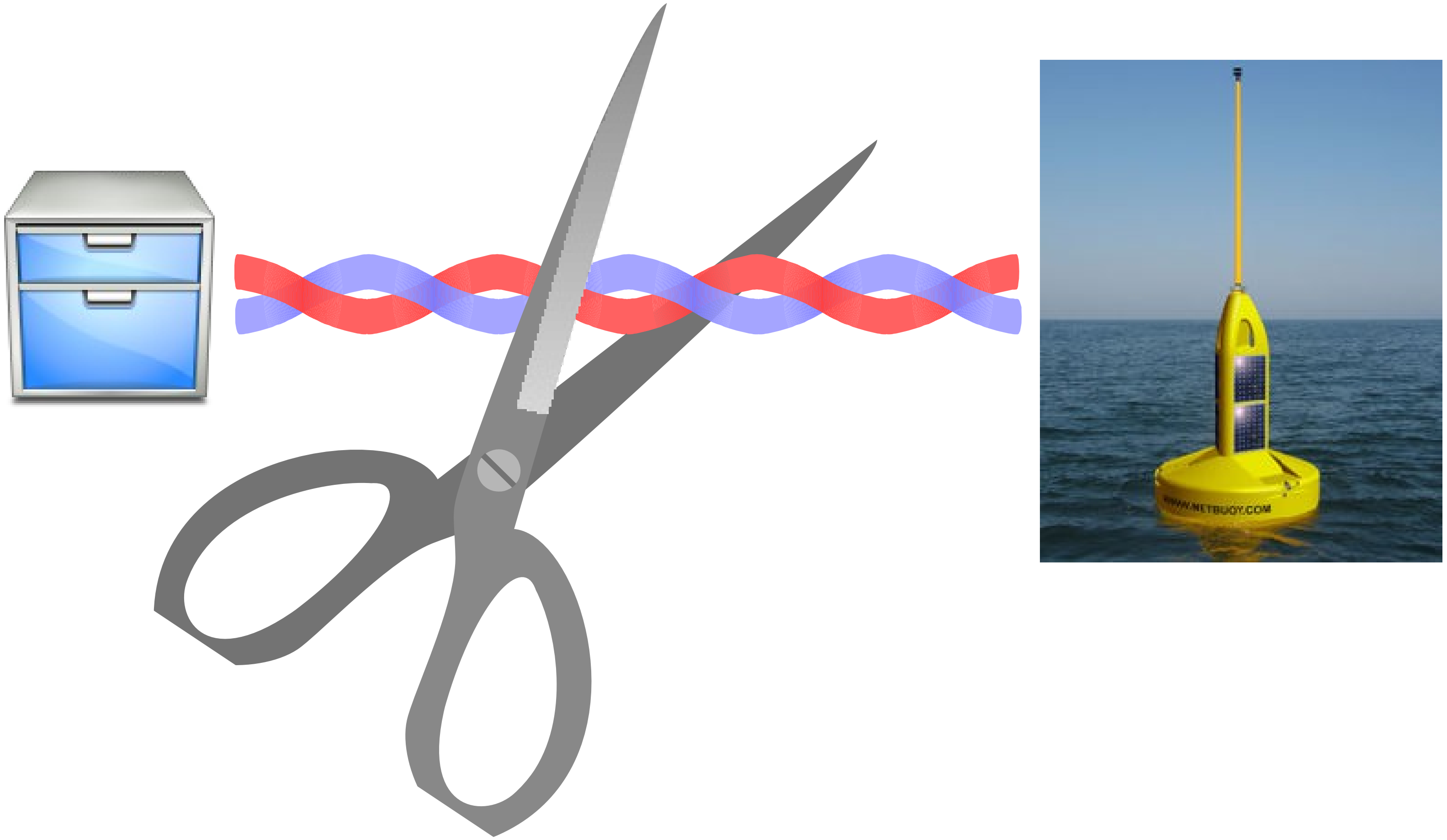
1 Failure mechanisms

- Bad firmware
- Power failure
- Flash corruption
- Communication errors

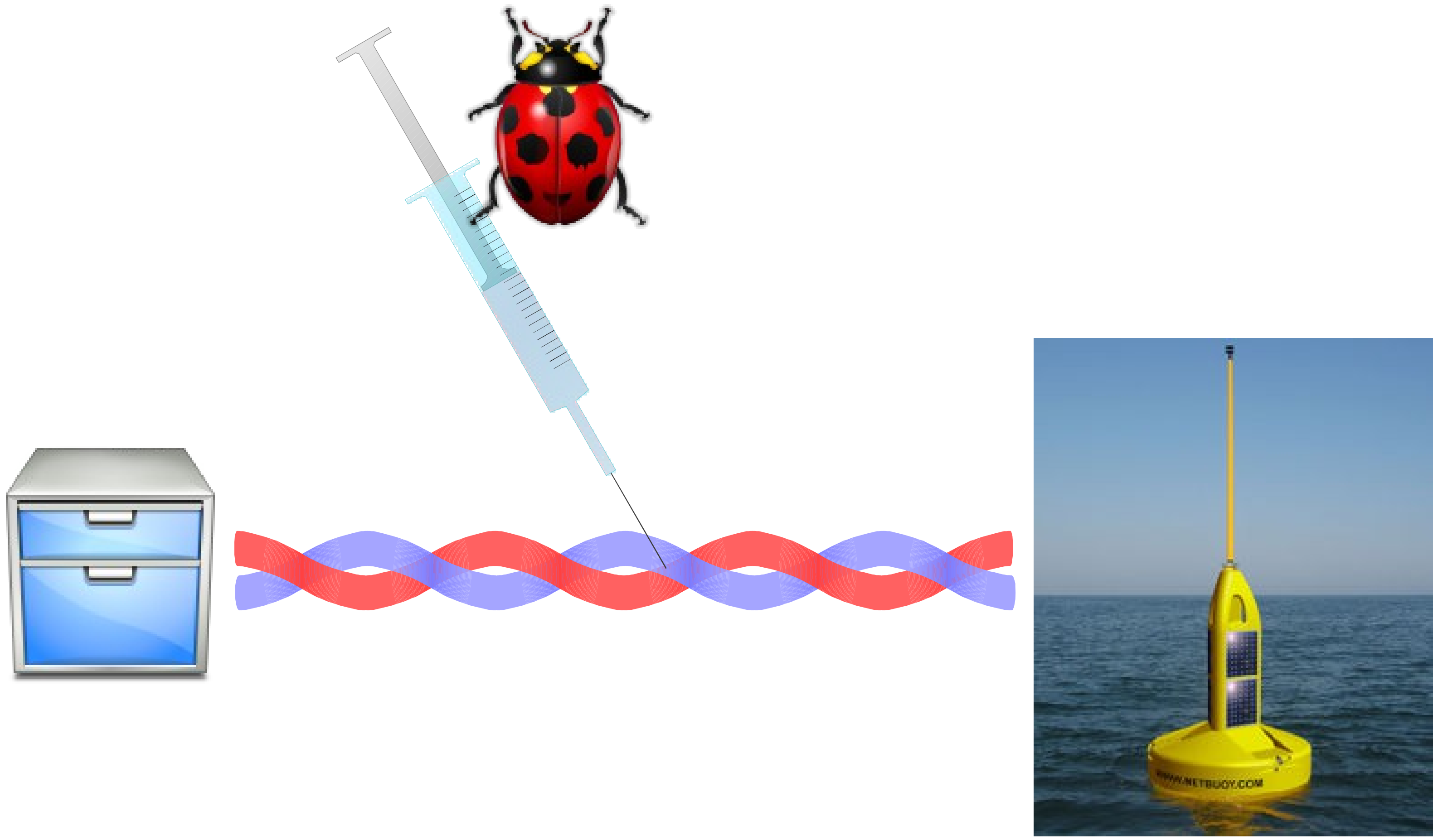
2 Boot loader upgrade

3 Package-based upgrade

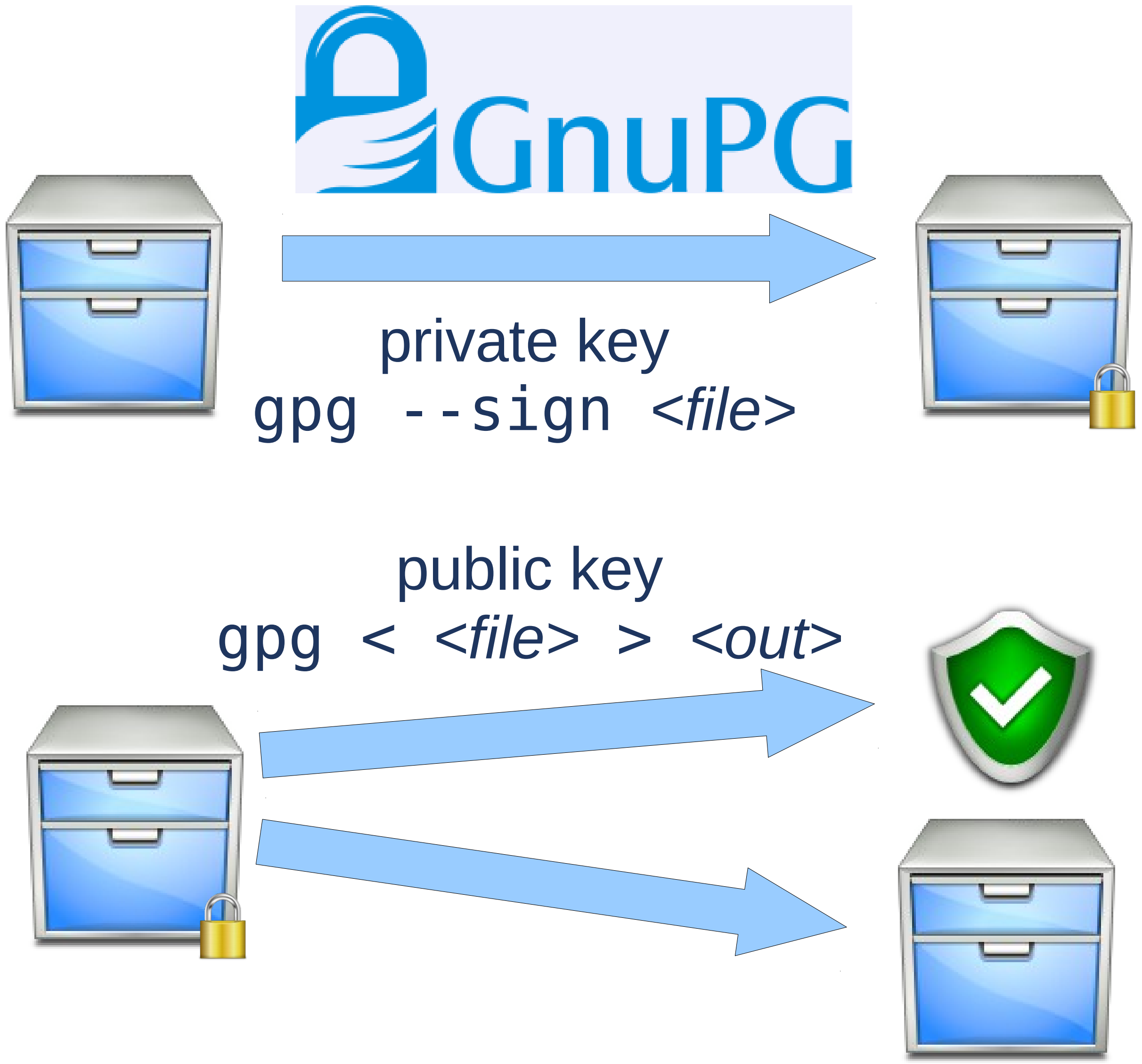
# Communication failures: Incomplete upgrade file



## Communication failures: False upgrade file injection



# Solution for communication failures: verify data before writing



private key

```
gpg --sign <file>
```

public key

```
gpg < <file> > <out>
```

## Take care with signed upgrade files

- Make it possible to install new public keys
  - Signer key may expire
  - Give third parties possibility to create upgrades
  - Avoid tivoization
- Make it possible to install revocations
  - Signer key may be stolen
- Make new keys and revocations accessible to fail-safe
- If upgrade file doesn't fit in memory:
  - Split it in chunks
  - Add an index (to check integrity )

## Overview

1 Failure mechanisms

- Bad firmware
- Power failure
- Flash corruption
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

## Upgrade of boot loader is never safe

If boot loader is broken

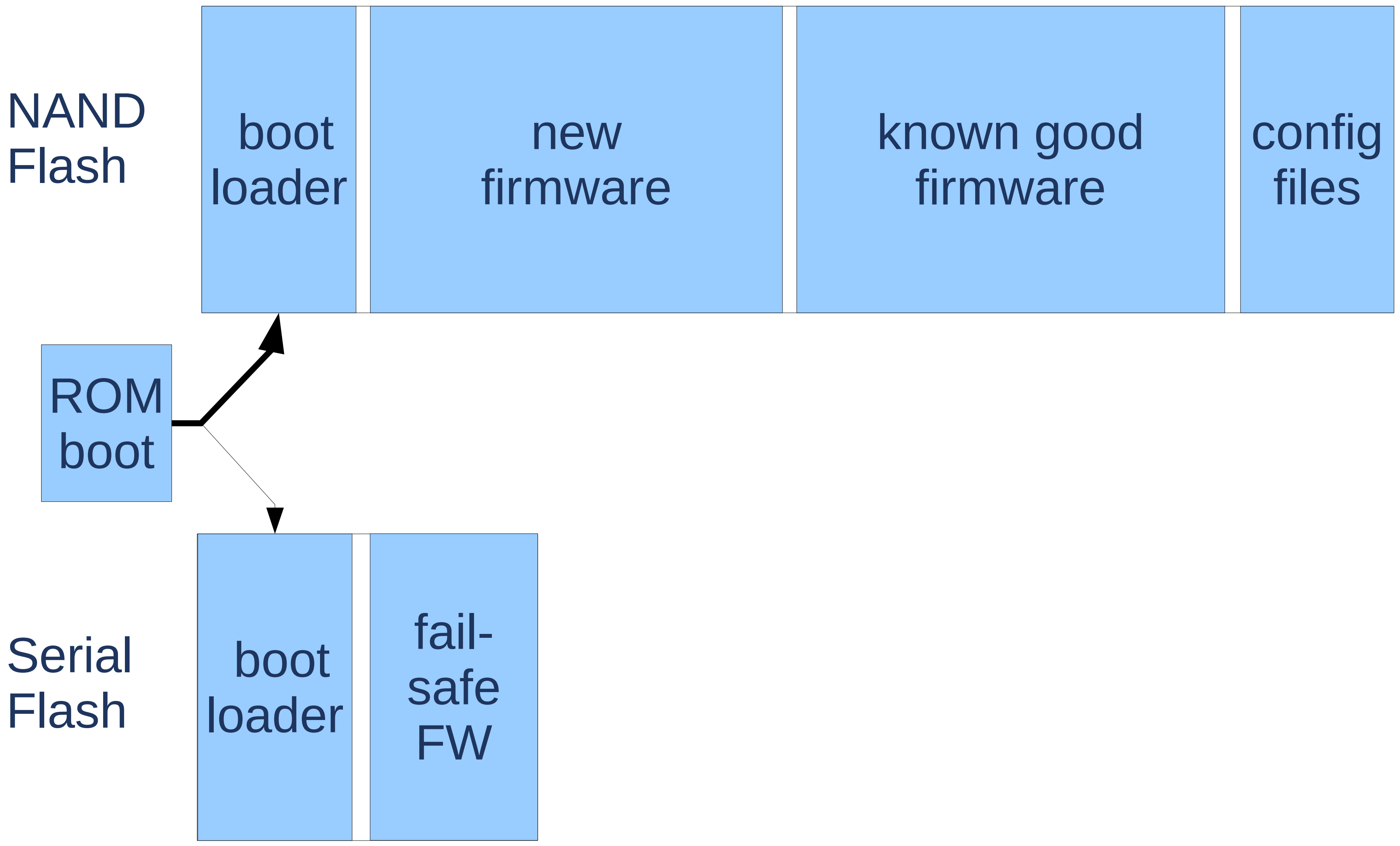
**No recovery is possible**

(unless a ROM boot loader comes first)

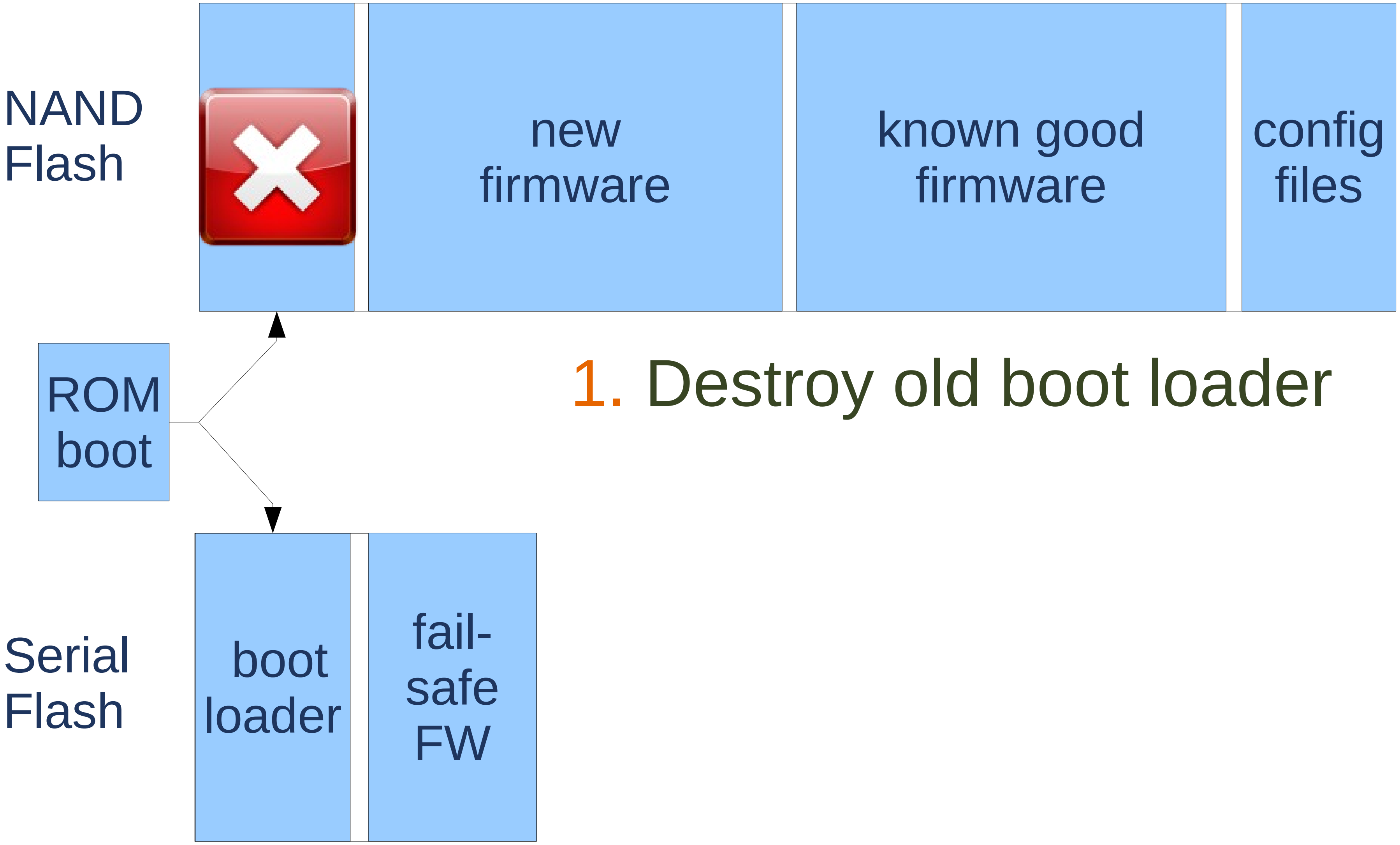
⇒ don't put bugs in the boot loader

⇒ don't put features in the boot loader

## Upgrade of boot loader with backup media

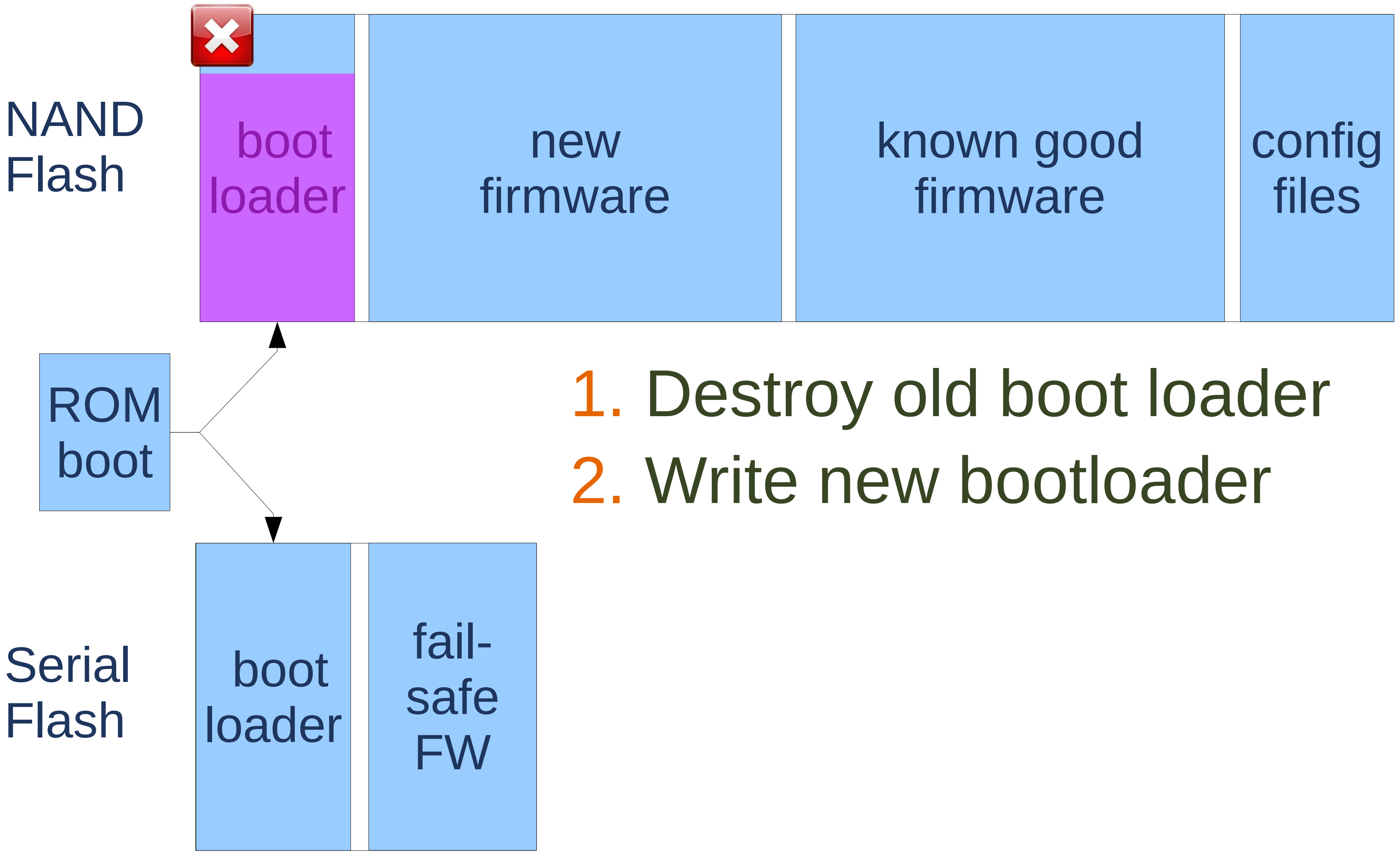


# Upgrade of boot loader with backup media



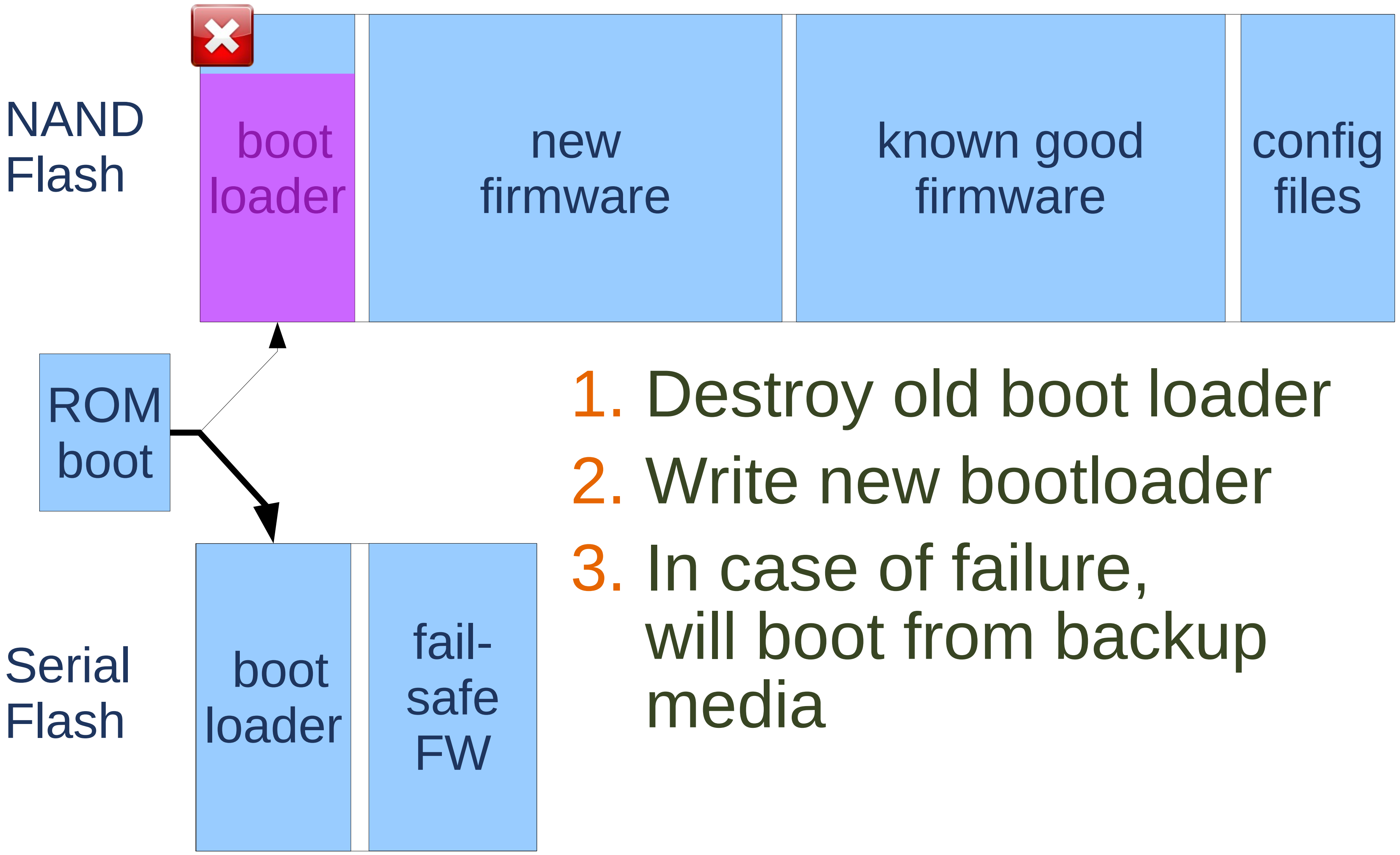
1. Destroy old boot loader

# Upgrade of boot loader with backup media



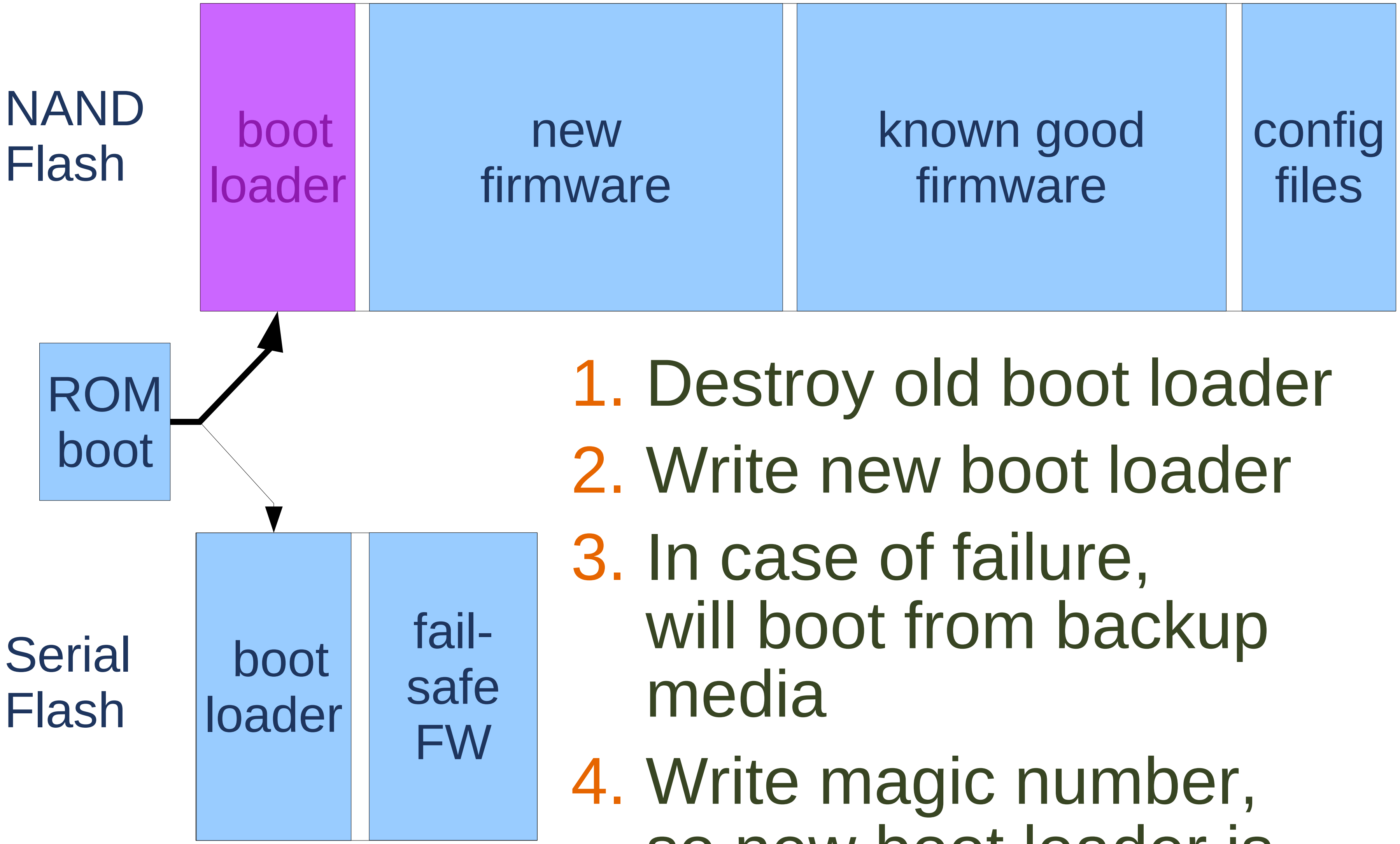
1. Destroy old boot loader
2. Write new bootloader

# Upgrade of boot loader with backup media



1. Destroy old boot loader
2. Write new bootloader
3. In case of failure, will boot from backup media

# Upgrade of boot loader with backup media



1. Destroy old boot loader
2. Write new boot loader
3. In case of failure, will boot from backup media
4. Write magic number, so new boot loader is

## Overview

1 Failure mechanisms

- Bad firmware
- Power failure
- Flash corruption
- Communication errors

2 Boot loader upgrade

3 Package-based upgrade

## Packaged-based upgrades are not ideal for embedded systems

Use a package manager (ipkg, opkg, dpkg, rpm) and upgrade individual packages

Advantage: smaller upgrade files

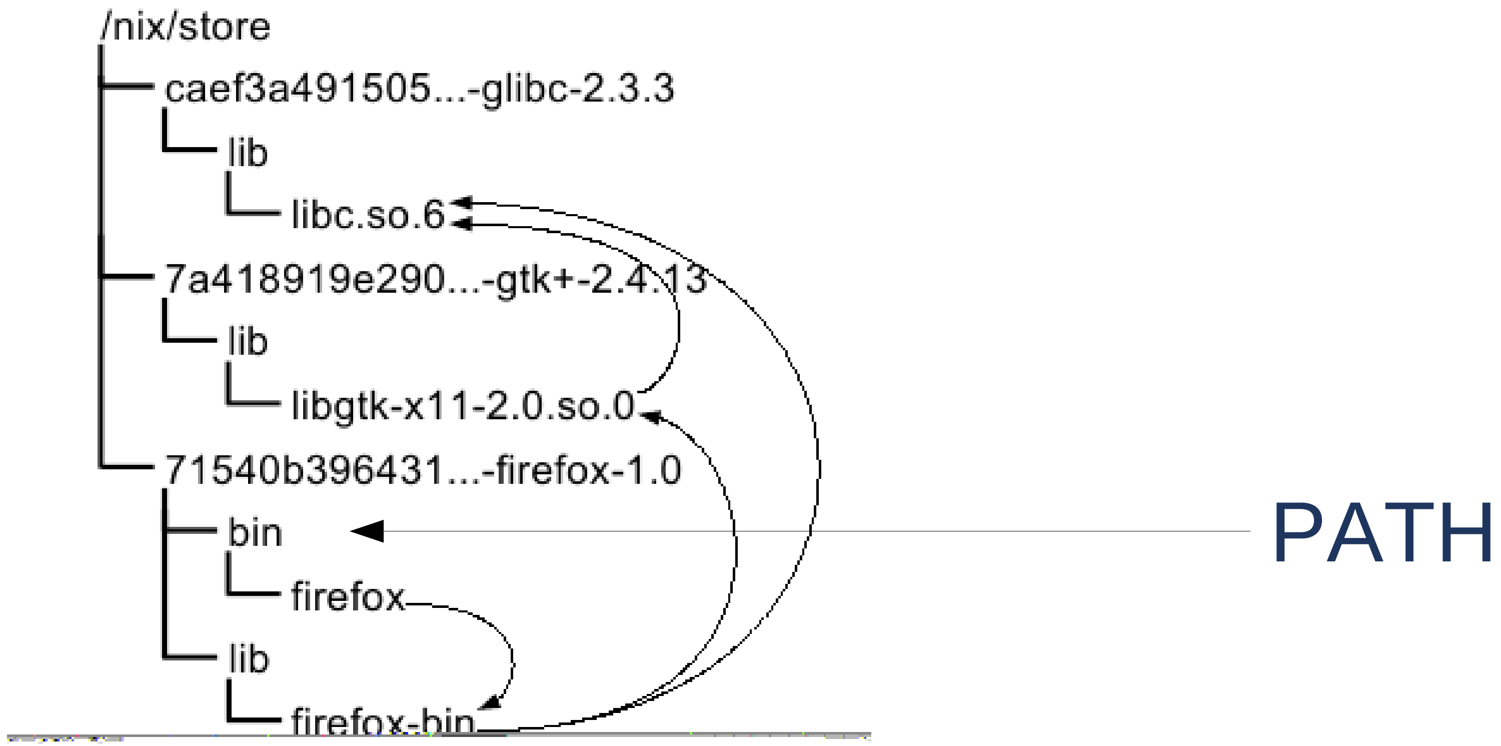
Disadvantages:

- Difficult to predict what is installed exactly
  ⇒ don't rely on version numbers,
  but use manifest with exact package versions
- More places where something can go wrong (Murphy)
- No package manager is truly atomic
  closest: http://nixos.org

## In a typical package-based system things can go wrong

1) Execute removal script
   - Shut down daemon
   - Remove some generated files
2) Remove old files
3) Upgrade dependencies
4) Install new files
5) Execute install script
   - Create new users
   - Create new directories
   - Start daemon

# Nix package manager is largely atomic


/nix/store
caef3a491505...-glibc-2.3.3
lib
libc.so.6
7a418919e290...-gtk+-2.4.13
lib
libgtk-x11-2.0.so.0
71540b396431...-firefox-1.0
bin
firefox
lib
firefox-bin
PATH

Eelco Dolstra. Efficient Upgrading in a Purely Functional Component Deployment Model. In George Heineman et al. (Ed.), Eighth International SIGSOFT Symposium on Component-based Software Engineering (CBSE 2005), volume 3489 of Lecture Notes in Computer Science, pages 219–234, St. Louis, Missouri, USA. Springer-Verlag, May 2005. © Springer-Verlag.

## Conclusions

- Take into account different failure mechanisms: bad firmware, power failure, communication failure, flash corruption
- No single ideal upgrade mechanism exists
  Some things really depend on the hardware
- No (open source) upgrade software exists

## Take your time to get the upgrade system right!

- Take into account different failure mechanisms: bad firmware, power failure, communication failure, flash corruption
- No single ideal upgrade mechanism exists
  Some things really depend on the hardware
- No (open source) upgrade software exists

http://mind.be/content/Presentation_Safe-Upgrade.pdf or .odp

www.mind.be

www.essensium.com

**Essensium NV**
**Mind - Embedded Software Division**
**Gaston Geenslaan 9, B-3001 Leuven**
**Tel : +32 16-28 65 00**
**Fax : +32 16-28 65 01**
**email : info@essensium.com**