UNIVERSIDAD COMPLUTENSE MADRID

Robustness. Watchdog

Error management

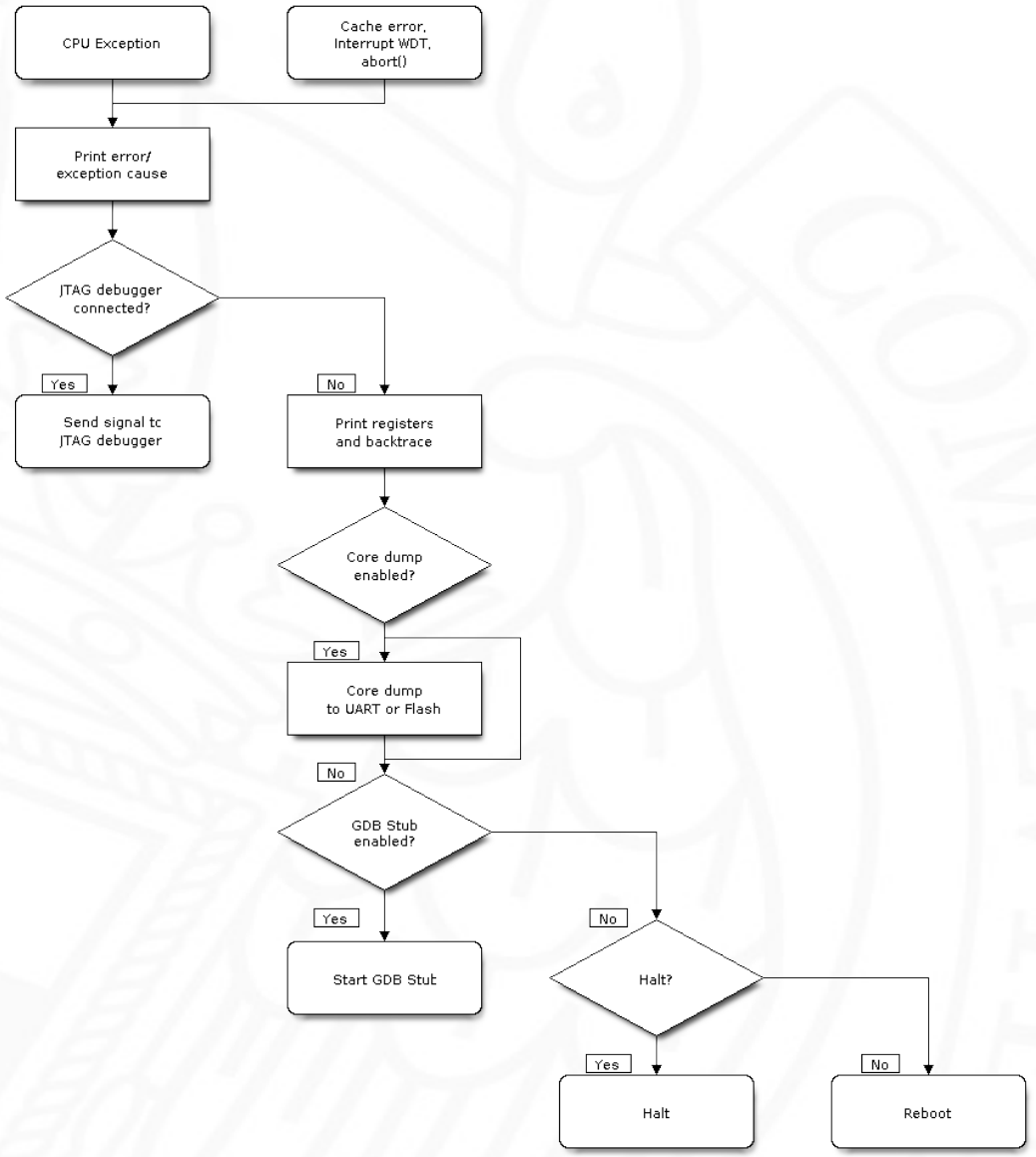
- Our code must check potential errors
  - And try to recover if an error raises
- *Recoverable* errors
  - Function returns an error code: typically an integer (enum)
  - Throw an exception (C++ -> `throw()`)
- Fatal errors
  - Use *`assert()/abort()`*
  - HW exceptions Excepciones HW
  - System level checks: Watchdogs, heap and/or stack corruption detection

Error codes (int)

- The usual way in UNIX world (syscalls)
  - Return ‘0’ if OK
  - Negative if something got wrong
    - Or just, different to 0
    - With a meaning for each of them
    - https://docs.espressif.com/projects/esp-idf/en/stable/api-reference/error-codes.html
- What if our function encounters an error?
  - Try to recover from error: try again, reset a module, a device...
  - Propagate the error to the clling function
    - Return error code / exception (throw() / catch())
    - Important: *undo* the work performed before the error (malloc(), open()....)
  - Make the error *fatal*
    - Using *assert() / abort()*
    - This is not valid for middleware, except during development

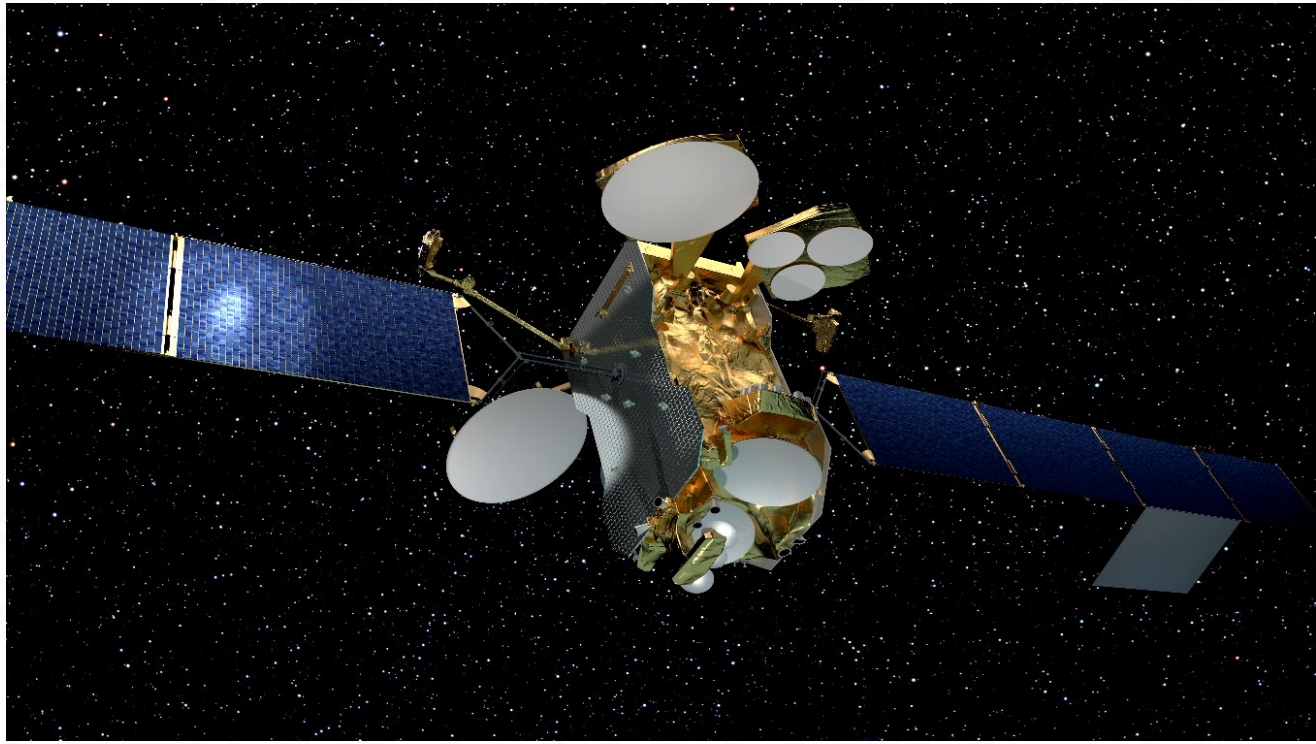
Managing errors. ESP-IDF

- ESP_ERROR_CHECK
  - https://docs.espressif.com/projects/esp-idf/en/stable/api-guides/error-handling.html#esp-error-check-macro
  - Similar to *abort()* but it chekcs the function returned value
  - The printed message includes localization information (source file, line...)
- Panic Handler
  - ESP-IDF manageer for fatal errors (exceptions, *watchdogs, stack overflow...)*
  - Configurable behavior
    - https://docs.espressif.com/projects/esp-idf/en/stable/api-guides/fatal-errors.html
    - Default: *core dump and reset*



What to do if everything goes wrong?

- Reset the system
  - Ctrl-Alt-Supr ????
- But.... What it the system in onboard a satellite?
  - Very likely there will be no keyboard!!



Watchdog

- A *watchdog* is a descendent counter. When it reaches 0, it restes the system
- Every (good) SoC has (at least) a *watchdog hardware (WDT)*
  - The counter is a *special timer*
  - Action configurable, time configurable…
- The main idea is that the code (middleware or application) must *kick the WDT* before it reaches 0
- If a *reset* happens due to WDT, we need to detect it after booting

Using a *watchdog*

- If the device hangs....
  - Memory corruption and the code ends up in a infinite loop
  - HW component not answering a request (GPS, modem...)
  - *Deadloack* due to misuse of sempahores
  - High priority task are execute always and do not let low priority tasks to execute
- How are we using the WDT?
  - Basic (incomplete) idea: low priority task whose only goal is to *kick the wathcdog*
  - If the system hangs, the task will not execute and the watchdog would trigger
  - But, what if a high priority task hangs ? Maybe because a sempahore is never released

ESP-IDF *watchdog* support

- ESP32 (*hardware*) has 3 WDTs
- ESP-IDF (*software*) provides 2 types of *watchdogs*
  - Interrupt watchdog: it checks if the FreeRTOS task scheduler has blocked for too long
    - Infinite loops with interrupts disabled, block in an ISR
    - It uses a hardware WDT
  - Task Watchdog Timer (TWDT): it checks that all tasks are progressing and none of them *abuse CPU* usage
    - By default, it tracks the *Idle Task*
    - Any task can ”ask for surveillance”
      - `esp_task_wdt_add(TaskHandle_t handle)`
    - If included in the surveillanve, the task must kick the TWDT periodically: `esp_task_wdt_reset(void)`
    - It uses another hardware WDT

https://docs.espressif.com/projects/esp-idf/en/stable/api-reference/system/wdts.html