Программно-аппаратные платформы Интернета вещей и встраиваемые системы

ПРИМЕР ОТЛАДКИ HARD FAULT

Постановка задачи ☺

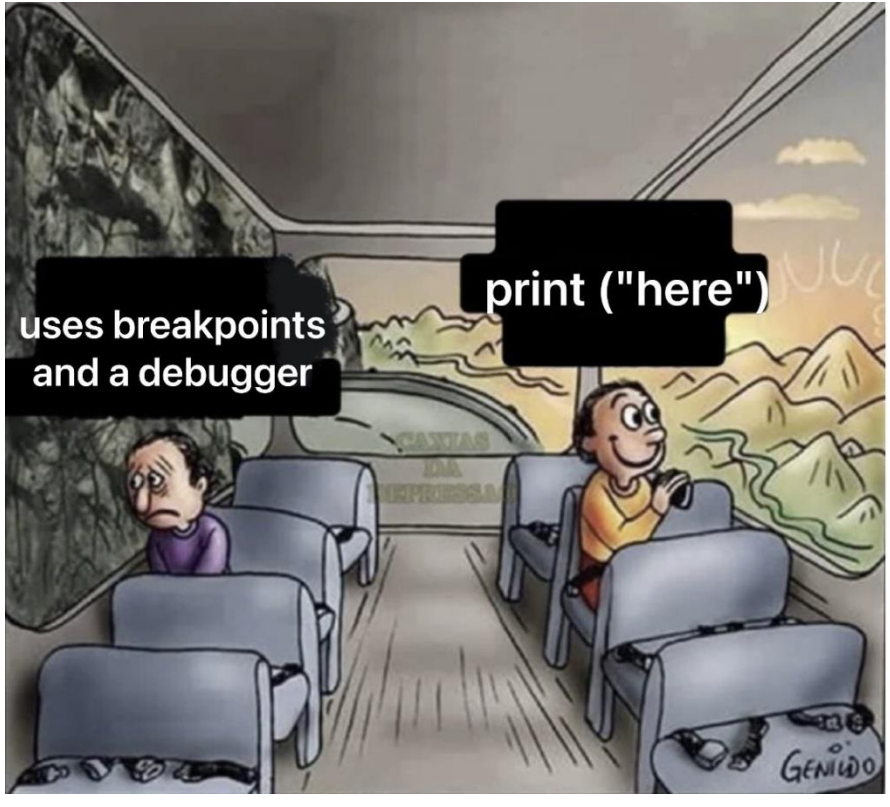
```c
#include <stdio.h>

int main(void) {
    printf("Hello World!");
    return 0;
}
```

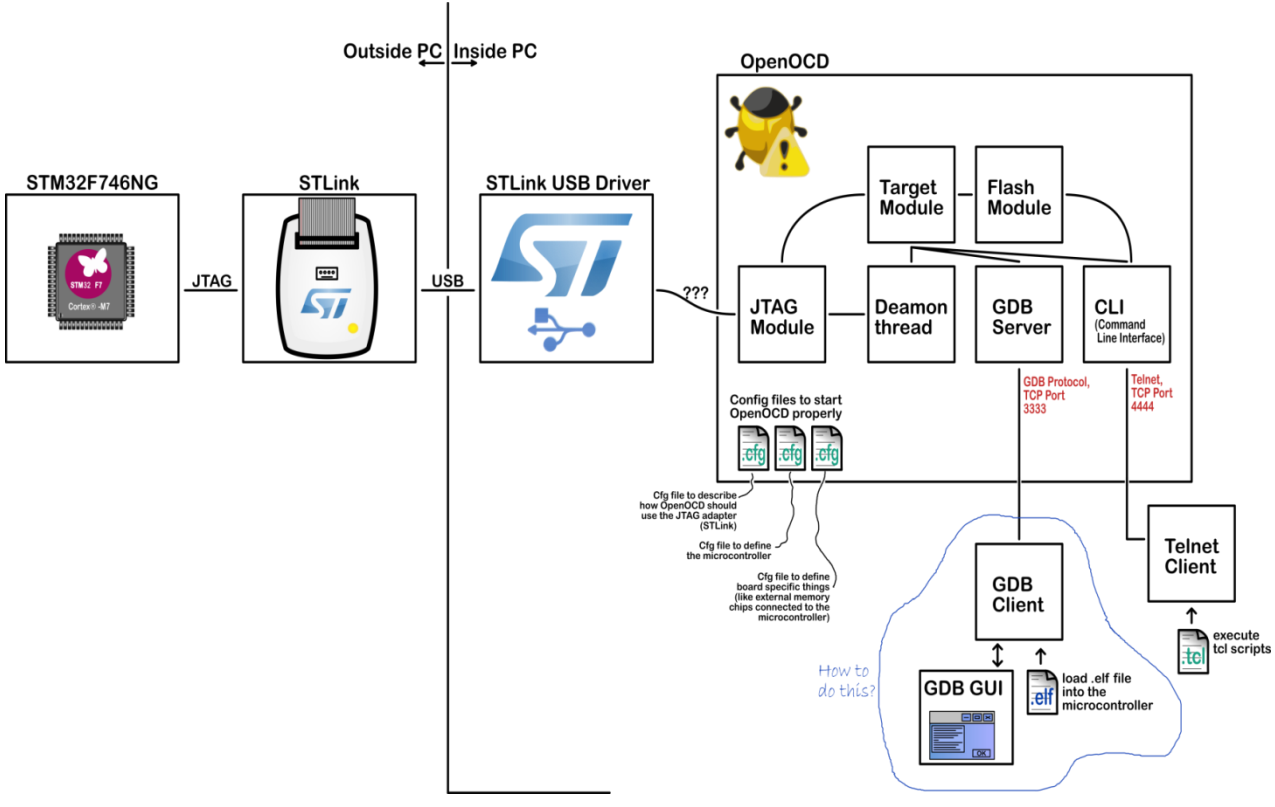
```
APPLICATION = hello-world
BOARD = nucleo-f401re
CFLAGS_OPT = -O3
RIOTBASE ?= $(CURDIR)/../..
DEVELHELP ?= 1
QUIET ?= 1
include $(RIOTBASE)/Makefile.include
```

После `make flash` плата не подает признаков жизни

Способы отладки



STLink, OpenOCD, gdb



Отладчик gdb

- Запускается командой `make debug` в каталоге с вашим проектом
- Основные команды:
  `break` – создать точку останова (breakpoint)
  `step` – пошаговое выполнение программы (step into)
  `next` – пошаговое выполнение программы (step over)
  `stepi`, `nexti` – шаг размером в одну машинную команду
  `continue` – продолжить выполнение программы
  `run` – перезапуск программы; RESET микроконтроллера
  `print` – напечатать значение переменной
  `display` – печатать значение переменной после каждого шага
  `where` – печать стека вызовов
  `disassemble` – печать дизассемблированного машинного кода
- Ctrl+C – остановка программы
- Имеются разнообразные графические оболочки, в том числе встроенные в IDE (Eclipse, VSCode); для них может пригодиться команда `make debug-server`

`hard_fault_handler` в RIOT

- Обработчик прерывания HardFault, возникающего при попытке выполнить некорректную операцию (неизвестная инструкция, доступ к несуществующему адресу в памяти, доступ к регистрам привилегированного режима, и так далее)
- По умолчанию в RIOT выводит содержимое регистров на момент сбоя с помощью `printf`
- На Cortex-M0 применяется также для определения объема памяти - https://habr.com/ru/post/437256/

Пример распечатки hard_fault_handler

```
Context before hardfault:
   r0: 0x00000000
   r1: 0x20001008
   r2: 0x0000000b
   r3: 0x00000000
  r12: 0x00000000
   lr: 0x080005d9
   pc: 0x080005da
  psr: 0x01000000

FSR/FAR:
 CFSR: 0x00008200
 HFSR: 0x00000000
 DFSR: 0x00000008
 AFSR: 0x00000000
 BFAR: 0x08001b1c
Misc
EXC_RET: 0x08002351
Attempting to reconstruct state for debugging...
In GDB:
  set $pc=0x80005da
  frame 0
  bt
```

Но это не наш случай…

Ошибка происходит в `reset_handler_default`, сразу после старта процессора, в почти тривиальном цикле:

```
uint32_t *dst = &_sstack;
while (dst < top) {
    *(dst++) = STACK_CANARY_WORD;
}
```

(кстати, `volatile uint32_t *dst;` «решает» проблему)

На самом деле

Включив в отладчике отображение текущей инструкции, пройдем по шагам (можно сделать и `disassemble`, но так будет проще):

```
break reset_handler_default
display/i $pc
run
nexti
```

Сбой происходит на этой инструкции:

```
0x800053e <reset_handler_default+54>: vldr d7, [pc, #336]
```

Кто виноват?

- При оптимизации компилятор заменил две операции с 32-битным регистром на одну операцию с 64-битным регистром FPU

Что делать?

- Включать сопроцессор как можно раньше:

```
cortexm_init_fpu();
```

https://github.com/RIOT-OS/RIOT/pull/16414