In-source tests with Ztest

HARDWARIO

2025-09-02

Marek Maškarinec

Ztest

- Zephyr library for writing automated tests

tests/add/src/main.c

```c
#include <subsys/adder/adder.h> // provides int add(int, int)

ZTEST_SUITE(add_tests, NULL, NULL, NULL, NULL, NULL);
ZTEST(add_tests, test_add_simple)
{
  zassert_equal(add(4, 2), 4 + 2);
}
```

- Testing can be done without hardware (native_sim, qemu targets, simulated devices)
- Each test is it's own Zephyr application
- Tests can be managed and run with CLI tool Twister
  - Scalability, support for multiple platforms

Ztest (ii)

```
$ west build -b native_sim
$ ./build/zephyr/zephyr.exe
*** Booting nRF Connect SDK v2.9.0-38914f587059 ***
*** Using Zephyr OS v3.7.99-52fc80b2508c ***
Running TESTSUITE add_tests
===================================================================
START - test_add_simple
 PASS - test_add_simple in 0.000 seconds
===================================================================
TESTSUITE add_tests succeeded

------ TESTSUITE SUMMARY START ------

SUITE PASS - 100.00% [add_tests]: pass = 1, fail = 0, skip = 0, total = 1 duration = 0.000 seconds
 - PASS - [add_tests.test_add_simple] duration = 0.000 seconds

------ TESTSUITE SUMMARY END ------

===================================================================
```

Problem with Ztest

- Ztest is mainly meant to test libraries (modules, subsystems etc.)
- Each test is it's own Zephyr application
  - Tests have to link againts the code they are testing
  - Testing private and application code is complicated

app/src/main.c

```
static int add(int a, int b) {
  return a - b;
}

int main(void) {
  ...
```

Solution

Tests directly in the source files

app/src/main.c

```
static int add(int a, int b) {
  return a - b;
}

ZTEST_SUITE(add_tests, NULL, NULL, NULL, NULL, NULL);
ZTEST(add_tests, test_add_simple)
{
  zassert_equal(add(4, 2), 4 + 2);
}

int main(void) {
  ...
```

Tests directly in the source files (ii)

```
$ west build
...
/usr/bin/ld.bfd: libsubsys__testsuite__ztest.a(ztest.c.obj): in function `main':
ztest/src/ztest.c:1459: multiple definition of `main'; app/libapp.a(main.c.obj)
...
```

1. Kconfig option for the in-source test

app/Kconfig

```
config APP_MAIN_TESTS
  bool "APP_MAIN_TESTS"
  default n
  depends on ZTEST
```

- Default should be set to `n` to prevent duplicate tests

2. Add preprocessor conditions

app/src/main.c

```
 static int add(int a, int b) {
   return a - b;
 }

 +#if defined(CONFIG_APP_MAIN_TESTS)
 ZTEST_SUITE(add_tests, NULL, NULL, NULL, NULL, NULL);
 ZTEST(add_tests, test_add_simple)
 {
   zassert_equal(add(4, 2), 4 + 2);
 }
 +#endif /* !defined(CONFIG_APP_MAIN_TESTS) */

 +#if !defined(CONFIG_APP_MAIN_TESTS)
 int main(void) {
```

3. Create a test application

tests/app_main/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20.0)
find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(app_main_tests)

target_sources(app PRIVATE ../../app/src/main.c)
```

tests/app_main/prj.conf

```
CONFIG_ZTEST=y
CONFIG_APP_MAIN_TESTS=y
```

Recapitulation

- Tests are put directly in the source files
- Source files are included in blank testing applications

Advantages

- Tests are near the implementation
  - they can serve as small samples
- Ability to test private code
- Access to static variable

Disadvantages

- Tests can inflate files a lot
  - possible workaround with `#include`
- Needs a lot of `ifdef`
- Not the official approach

Possible steps forwards

- Entire application could be a conditional test

Q&A