Table of Contents

1. Zephyr threads 2

1. Zephyr threads

In this part, we will show methods of exchanging data between threads. In the example from the previous part, threads were defined using K_THREAD_DEFINE()macro.

```
K_THREAD_DEFINE(thread0_id, STACKSIZE, thread0, NULL, NULL, NULL,
        THREAD0_PRIORITY, 0, 0);
```

The arguments of the macro are:

- the name of the thread
- size of the stack that is created for individual use of the thread
- address of the function of the thread
- three initial arguments of the thread which can be passed at the start of the thread (here all NULL)
- constant that specifies the priority of the thread
- thread options (here 0)
- delay to start the thread (0 meaning no delay)

As we can see every thread has its own stack, which means that if there is a need to exchange data between threads we have to use some shared variable visible for all threads interested in using it. In multitasking environments, especially running on a multi-core processor, there is a need to synchronise the access to shared resources. Zephyr offers some mechanisms for inter-thread data exchange and synchronisation.

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:aih:lab:nrf52threads2?rev=1700758119

Last update: **2023/11/23 17:48**