SWiIR - debugowanie

Zadanie 1.

Przygotuj środowisko i urządzenie nRF 53 DK do debuggowania z użyciem JLink GDB. Następnie uruchom debugger i przetestuj jego działanie wykorzystując skompilowany przykład z katalogu “Zajecia_nr_1/Zadanie_1”.

1. ~~Podłączamy płytkę nRF 52~~
2. ~~Pobieramy archiwum z oprogramowaniem J-Link (https://www.segger.com/downloads/flasher/JLink_Linux_x86_64.tgz)~~
3. Otwieramy pierwsze okno terminala
    a. Uruchamiamy serwer debuggera JLink komendą:
        ./JLinkGDBServerCLExe -if swd -device nrf52832_xxaa -speed auto
    b. Serwer GDB łączy się z płytką Nrf52 i po chwili jest gotowy do pracy. W konsoli powinien wyświetlić się następujący komunikat:
        Waiting for GDB connection…
4. Otwieramy drugie okno terminala
    a. Przejdź do katalogu gcc-arm-none-eabi-9-2019-q4-major/bin
    b. Uruchom GDB “arm-none-eabi-gdb” poleceniem
        ./arm-none-eabi-gdb
    c. **Pytanie**: Dlaczego nie możemy skorzystać ze standardowego GDB zainstalowanego w komputerze?
    d. Podłącz się z serwerem JLink GDB
        target remote :2331
    e. Załaduj do GDB plik wykonywalny “nrf52832_xxaa.out”, który chcemy uruchomić na nRF52 DK
        file *ścieżka_do_katalogu/*nrf52832_xxaa.out
    f. **Pytanie**: Dlaczego ładujemy plik nrf52832_xxaa.out a nie nrf52832_xxaa.hex?
        file _build/nrf52832_xxaa.out
        file _build/nrf52832_xxaa.hex
    g. Korzystając z komend GDB (tabela poniżej) wykonajcie następujące czynności:
        i. Ustaw pułapkę (breakpoint) na wywołaniu funkcji bsp_board_led_invert() w kodzie programu
        ii. Zrestartuj urządzenie wbudowane
        iii. Uruchom urządzenie tak, aby zatrzymało się w pułapce ustawionej w punkcie 1.
        iv. Wykonaj całą iterację pętli w trybie krokowym i obserwujcie terminal podpięty do UART na płytce
    h. **Pytanie:** Jaka jest różnica pomiędzy “next” i “step”?

Polecenie GDB	**Opis**
file *ścieżka/do/pliku*	Ładuje wybrany plik wykonywalny do sesji GDB

step	Wykonuje następną instrukcję wynikająca z przebiegu kodu źródłowego.
breakpoint *plik.c*:*numer_linii*	Ustawia pułapkę (breakpoint) w podanej linii i pliku
continue	Kontynuuj wykonanie programu aż do następnej pułapki
monitor reset	Ustaw domyślny stan urządzenia (m.in. zrestartuj i wyczyść rejestry MCU)
next	Wykonuje następną linię (wyrażenie) w kodzie źródłowym
load	Programuje urządzenie plikiem wykonywalnym, który został załadowany do sesji GDB
quit	Zakończenie sesji debuggowania i zamknięcie GDB

Zadanie 2

Napisz program, który będzie pozwalał na komunikację pomiędzy PC <-> nRF 52 DK przy użyciu UART. Program ma umożliwiać sterowanie wyświetlanymi komunikatami za pomocą odebranych znaków ASCII. Wysłanie znaków ‘1’, ‘2’, ‘3’ lub ‘4’ spowoduje odesłanie odpowiadającego mu napisowi. Po odebraniu jakiegokolwiek innego znaku nRF 52 powinien odpowiedzieć ciągiem “Bledny znak!”.

1. Przejdź katalogu “Zadanie_2” i w pliku MAKEFILE ustaw zmienną “SDK_ROOT” tak, żeby prowadziła do folderu “nRF5_SDK_14.2.0_17b948a”
2. Otwórz plik main.c i uzupełnij brakujący kod
3. Pobierz narzędzia “nRF5x-Command-Line-Tools-Linux64” ze strony https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK i rozpakuj je w katalogu “Zadanie_2”
4. Zapoznaj się z opcjami narzędzia “nrfjprog” i zastanów się jak można z jego pomocą zaprogramować płytkę nRF52 DK

   ```
   nrfjprog --help
   ```
5. W drugim oknie terminala uruchom program minicom lub mniterm.py

   ```
   minicom -D /dev/ttyACM0 -b 115200
   ```

   lub

   ```
   miniterm.py /dev/ttyACM0 115200
   ```

Skorzystaj z dokumentacji SDK dostępnej w katalogu “https://infocenter.nordicsemi.com/index.jsp?topic=%2Fstruct_sdk%2Fstruct%2Fsdk_nrf5_latest.html&cp=7_1”. Funkcje pozwalające na obsługę UART są opisane w sekcji “API Reference -> SDK common libraries -> UART Module”

Uwaga!

Kod źródłowy napisany w ramach zadania nr 2 musi pod koniec zajęć być umieszczony w Waszym repozytorium na Github. Jest to warunek **konieczny** do zaliczenia tego zadania!

Zadnie 3
Uruchom logowanie do `JlinkRTTClient`
https://www.segger.com/products/debug-probes/j-link/technology/about-real-time-transfer/

https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v11.0.0%2Flib_nrf_log.html

https://infocenter.nordicsemi.com/index.jsp?topic=%2Fug_gsg_ses%2FUG%2Fgsg%2Fconnect_rtt_linux.html

```
JLinkExe -device nrf52832_xxaa -if SWD -speed 4000 -autoconnect 1
```

```
JlinkRTTClient
można też:
telnet localhost 19021
```