Laboratorium wprowadzające (0) Mikrokontrolery ARM

Lucjan Bryndza Politechnika Warszawska

28 listopada 2019

Celem laboratorium wprowadzającego jest zapoznanie studenta z zestawami ewaluacyjnymi STM32F411E oraz STM32F469I discovery, środowiskiem deweloperskim opartym o kompilator *GCC* oraz *Visual Studio Code*. Zapoznaniem ze sposobami uruchamiania i debugowania kodu w środowisku deweloperskim. Zadaniem studenta będzie również zapoznanie się z podstawowymi funkcjami systemowymi systemu **ISIX**.

1. Zadania do wykonania na ćwiczeniach

1. Pobrać przykład 0 (*LED blink*) dla zestawu STM32F411 oraz STM32F469 za pomocą repozytorium kodu **GIT**.
2. Skonfigurować i skompilować projekt *led0_blink* dla obu płytek ewaluacyjnych.
3. Zaprogramować oba zestawy ewaluacyjne.
4. Sprawdzić czy mruga dioda LED a na diagnostycznym porcie szeregowym wypisywane są komunikaty informujące o stanie diody LED.
5. Korzystając z przykładu **lab0_led_blink** sort napisać funkcję sortującą liczby typu **int** z wykorzystaniem algortmu: **selection sort** dla obu zestawów ewaluacyjnych.
6. Przy uruchamianiu przykładu sortowania należy skorzystać z możliwości debugowania zestawu ewaluacyjnego (GDB).
7. Korzystając z przygotowanej funkcji sortowania dokonać serii 20 pomiarów czasów sortowania 4096 liczb typu int z tablicy int **array_for_sort**[] dla obu zestawów ewaluacyjnych. Zapisać wyniki pomiarów.
8. W domu na podstawie zebranych wyników opracować sprawozdanie. Szczegółową instrukcję wykonania sprawozdania możemy znaleźć w rozdziale: 3.

2. Materiały pomocnicze

2.1. Podłączenie zestawu STM32F411-DISCOVERY do portu szeregowego

Ponieważ zestaw **STM32f411E-DISCO** nie posiada wyprowadzonego portu szeregowego, umożliwiającego bezpośrednie dołączenie zestawu do komputera, musimy skorzystać dodatkowej przejściówki RS232(TTL) → USB. Możemy wykorzystać dowolną przejściówkę realizującą to zadanie. W opisie zaprezentowano konwerter z układem FT232RL o oznaczeniu WSR-04502, ale w przypadku innego konwertera sposób postępowania będzie podobny. Układ należy połączyć z płytką ewaluacyjną za pomocą przewodów według konfiguracji z tabeli 1:

Konwerter USB	STM32F411E-DISCO
GND (niebieski)	GND
RXD (zielony)	PA2 (*USART2_TXD*)
TXD (czerwony)	PA3 (*USART2_RXD*)

Tabela 1: Połączenie konwertera RS232-USB z zestawem STM32F411E-DISCO

Zestaw STM32F469I-DISCO wyposażony jest w zintegrowany programator STLINK-V2.1, który za pomocą interfejsu USB udostępnia dodatkowy UART diagnostyczny, zatem nie ma potrzeby dołączania dodatkowej przejściówki. Szeregowy port diagnostyczny połączony wewnętrznie z interfejsem **USART3**: PB11(RX), PB10(TX)

2.2. Konfiguracja pobranie oraz uruchomienie projektu

Aby pobrać repozytorium możemy się posłużyć komendą, którą należy uruchomić w wierszu polecenia *GIT Bash*:

```
git clone --recursive -b pub/pw/lab0 https://boff.pl/cgit/
  ↪ public/isixsamples/
```

W kolejnym kroku należy zmienić katalog na *isixsamples* oraz pobrać konfigurację dla środowiska *VSCode*:

```
cd isixsamples
curl http://bryndza.boff.pl/downloads/prv/vscodecfg.zip --
  ↪ output vscodecfg.zip
```

Pobrany plik należy rozpakować bezpośrednio do katalogu *isixsamples*.

Konfiguracja projektu dla zestawu STM32F411 discovery odbywa się za pomocą polecenia:

```
python waf configure --debug --board=
  ↪ stm32f411e_disco
```

Konfiguracja projektu dla zestawu STM32F469 discovery odbywa się za pomocą polecenia:

```
python waf configure --debug --board=
  ↪ stm32f469i_disco
```

Kompilacje projektu możemy wykonać za pomocą polecenia:

```
python waf
```

Natomiast zaprogramowanie zestawu odbywa się za pomocą polecenia:

```
python waf program
```

Polecenie *waf program* odczytuje plik docelowy którym należy zaprogramować mikrokontroler z pliku **config.json**. Przy zmianie zestawu na STM32F469 należy zmienić scieżkę **target** w sekcji **jtag**:

```
{
	"jtag": {
		"target":  "stm32f469i_disco/lab0_led_blink/
		  ↪ led_blink",
		"type":  "stlink-v2-1",
		"speed":  2000,
		"swd": 1
	},
	"configure" : {
		"disable_isix": false,
		"disable_exceptions" : true,
		"debug" : true
	}
}
```

Projekt możemy również konfigurować oraz uruchamiać bezpośrednio z *Visual Studio Code*. Najpierw należy w pliku *.vscode/task.json* zmienić argumenty dla polecenia *waf configure*. Następnie za pomocą polecenia **CTRL+P** otwieramy wiersz polecenia i wpisujemy *task waf configure* w kolejny kroku należy zbudować projekt za pomocą polecenia *task waf*, a w kolejnym kroku zaprogramować poleceniem *task waf program*.

Debugowanie programu następuje po wciśnięciu klawisza **F5** wcześniej jednak należy w pliku *.vscode/launch.json* ustawić odpowiednio ścieżkę do uruchamianego programu.

Opis skrótów klawiaturowych dla środowiska **VSCode** możemy znaleźć tutaj: [5].

2.3. Podstawowe API systemu ISIX do ćwiczenia

Podstawową funkcją diagnostyczną służącą do wypisywania danych na domyślny port diagnostyczny (terminal) jest funkcja:

```
#define dbprintf(fmt, ...) fnd::tiny_printf("%s:%d|" fmt "\r\n",
  ↪ _isix_dbglog_extract_basename(__FILE__),__LINE__,##
  ↪ __VA_ARGS__)
```

Funkcja przyjmuje identyczny zestaw argumentów jak standardowa funkcja **printf** pozbawiona jest jedynie formatowania i wyświetlania liczb zmiennoprzecinkowych.

Do pomiaru czasu wykonania funkcji sortowania możemy wykorzystać funkcję systemową ISIX wykorzystującą timer o rozdzielczości mikrosekundowej:

```
/** Get current sytem ticks
 * @return Number of system tick from system startup in usec
   ↪ resolution
 */
static inline osutick_t isix::get_ujiffies(void);
```

Funkcja ta zwraca 64 bitową liczbę mikrosekund jaka upłynęła od momentu uruchomienia systemu.

Do utworzenia nowego wątku systemu operacyjnego możemy wykorzystać funkcję:

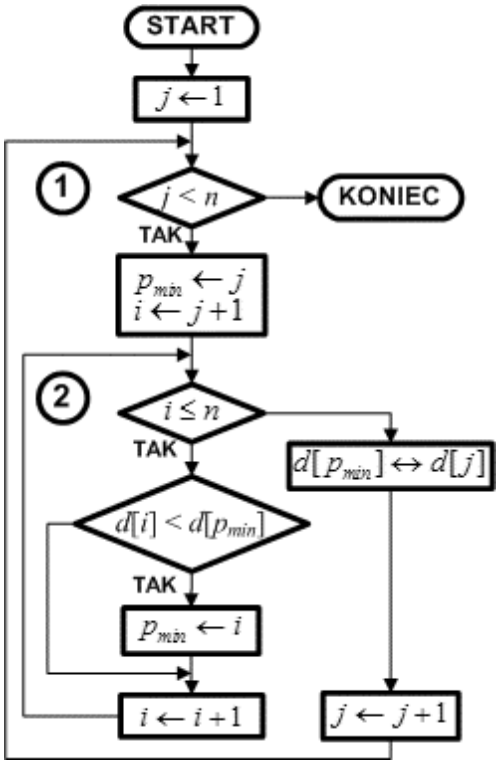
```
ostask_t isix::task_create(task_func_ptr_t task_func, void *
   ↪ func_param, unsigned long stack_depth, osprio_t priority,
   ↪ unsigned long flags=0);
```

Jako pierwszy argument funkcja przyjmuje funkcję, która będzie wykorzystana do realizacji wątku. Jako argument *func_param* możemy przekazać argument przekazany funkcji realizującej wątek. Parametr *stack_depth* określa wielkość stosu dla danego wątku. W przypadku sortowania powinien wystarczyć stos o wielkości 1kB. Jako parametr *priority* należy przekazać priorytet wątku, przy czym 0 oznacza priorytet najwyższy.

2.4. Algorytm selection sort

Sortowanie przez wybieranie polega na wyszukaniu elementu mającego się znaleźć na żądanej pozycji i zamianie miejscami z tym, który jest tam obecnie. Operacja jest wykonywana dla wszystkich indeksów sortowanej tablicy. Algorytm przedstawia się następująco:

- wyszukaj minimalną wartość z tablicy spośród elementów od i do końca tablicy
- zamień wartość minimalną, z elementem na pozycji



3. Instrukcja opracowania sprawozdania

- Napisać identyczny program sortowania na komputerze PC, oraz wyznaczyć średni czas sortowania podobnie jak dla płytek ewaluacyjnych.
- Wyznaczyć średni czas sortowania dla obu zestawów ewaluacyjnych oraz dla komputera PC.
- Wyznaczyć niepewność standardową wyników pomiarów.(Odchylenie standardowe wielkości średniej)
- Wyciągnąć wnioski na temat wydajności obu płytek w porównaniu do komputera PC.
- W sprawozdaniu umieścić informację o konfiguracji sprzętowej komputera PC. Model procesora, częstotliwość zegara.

Literatura

[1] *STM32F411E-DISCOVERY* kit user manual

[2] *STM32F469I-DISCOVERY* kit user manual

[3] Selection sort algorithm

[4] *ISIX-RTOS* API examples

[5] *Visual Studio Code* keyboards shortcuts