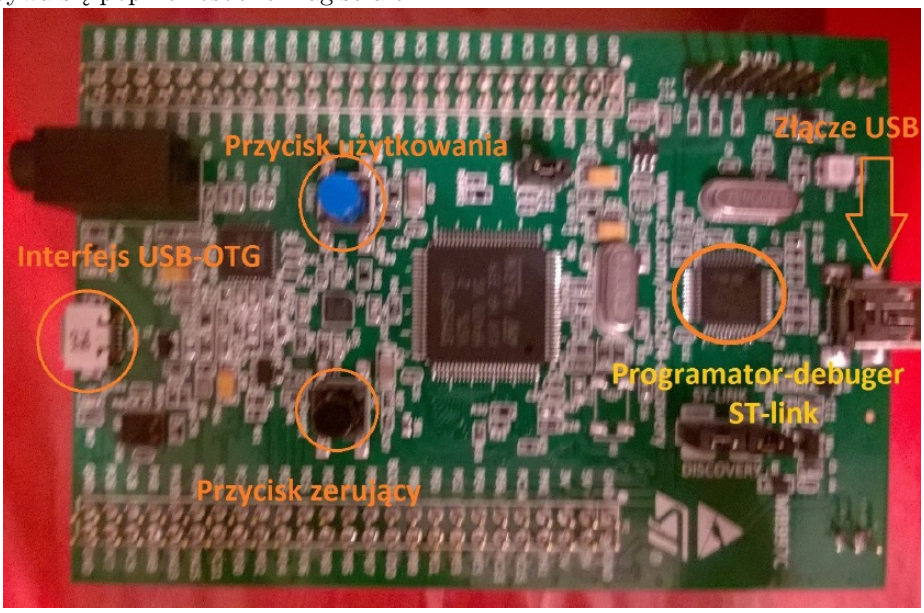
.NET Micro Framework STM32F4 Discovery

Wojciech Duda

2016.4.21

1 Teoria

Rdzeń CortexM4F wykorzystuje architekturę ARMv7M. Pod względem organizacji pamięci jest to architektura harwardzka, tzn. pamięć zawierająca kod programu (Flash) i pamięć danych (SRAM) są rozdzielone i dostęp do nich odbywa się poprzez osobne magistrale.



2 Instalacja

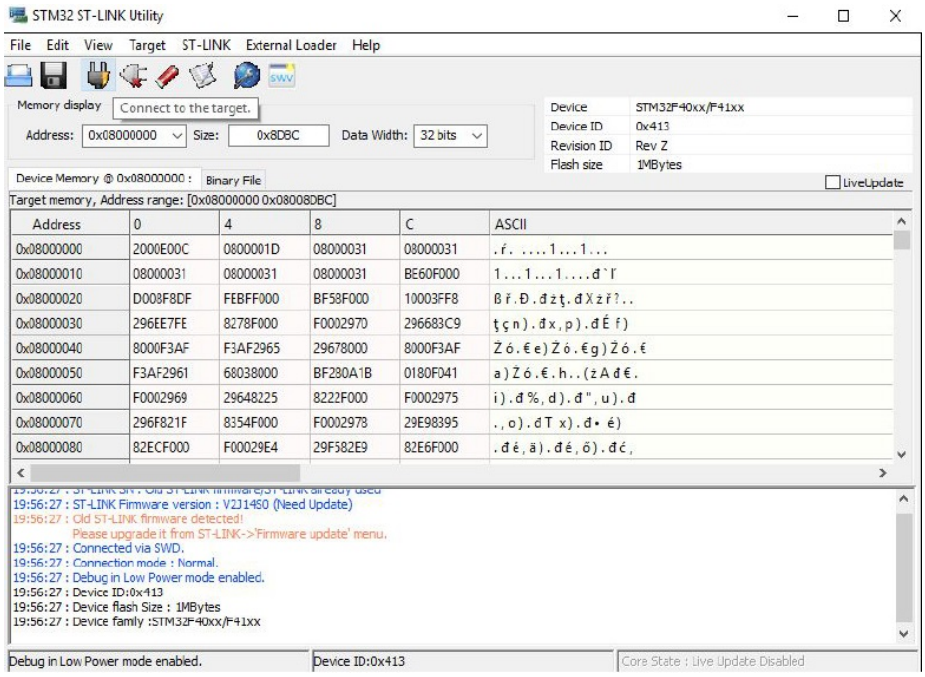
2.1 Narzędzia:

- mikrokontroler STM32F4 Discovery
- kable USB Micro oraz USB Mini
- Visual studio
- STM32 ST-LINK Utility
- sterwonik USB
- bootloader oraz pliki hex
- .NET MicroFramework SDK

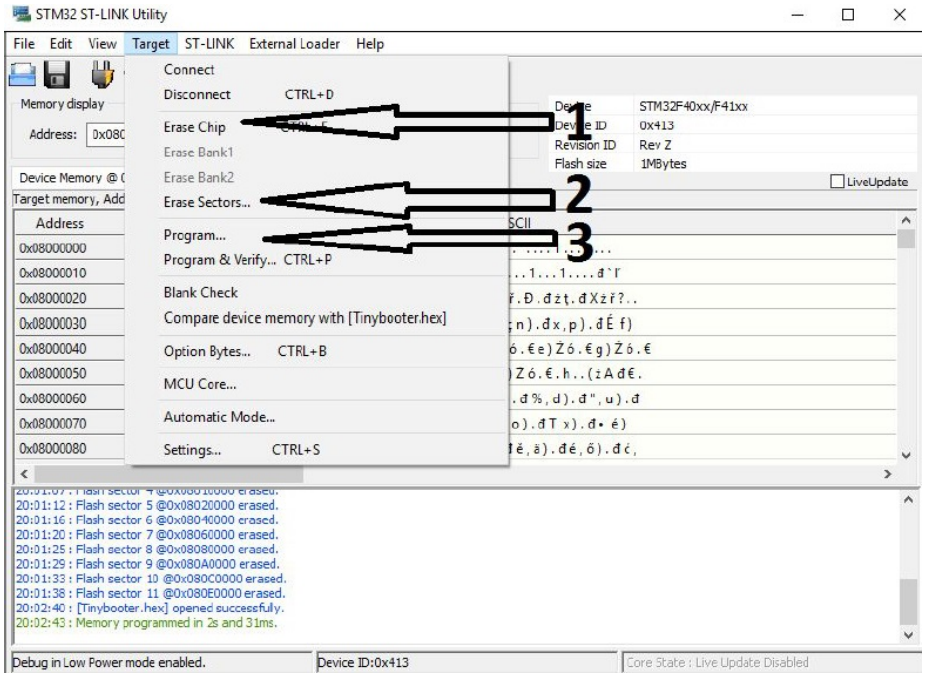
2.2 Zainstaluj STLINK, oraz SDK, resztę pilków rozpakuj.

2.3 Podłącz kabel USB Mini (do wejścia oznaczonego jako "Złącze USB" na zdjęciu powyżej.)

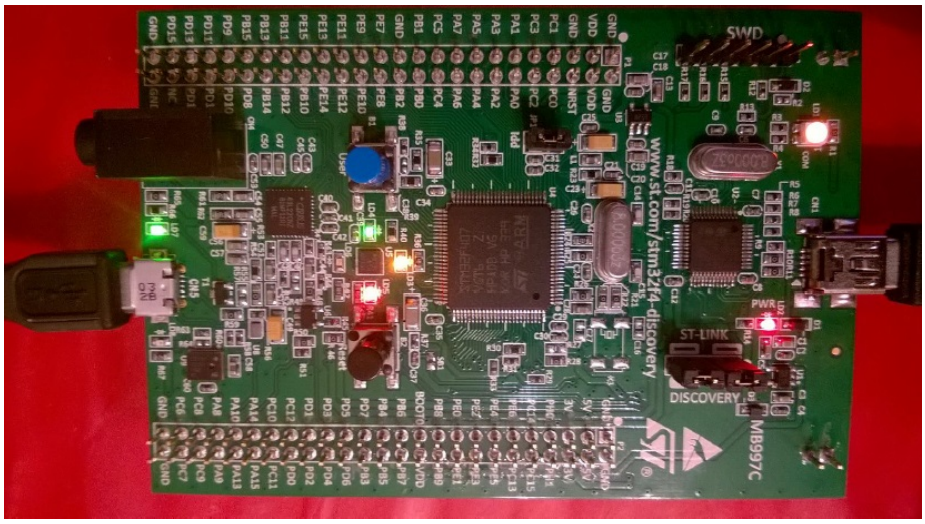
2.4 Włącz STLINK Utility , a następnie połącz się z stm32f4 poprzez przycisk: "Connect to the = target"



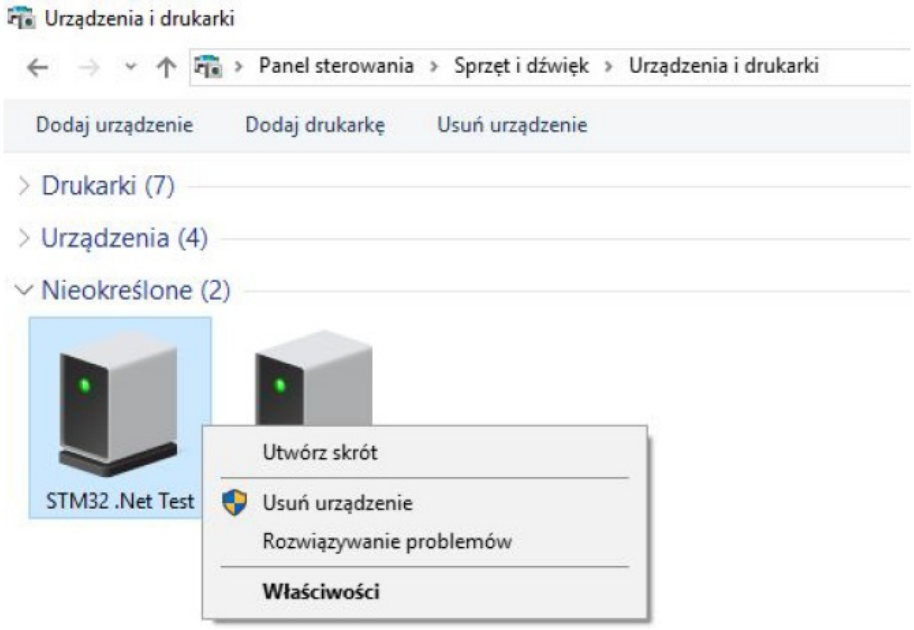
2.5 Następnie wybierz Target >Erase Chip oraz Target>Erase Sectors, wybierz wszystkie i potwierdź. Wybierz Target >Program. . . , wybierz ścieżkę Tinybooter.hex a następnie wybierz start. Zresetuj płytkę poprzez przycisk zerujący.



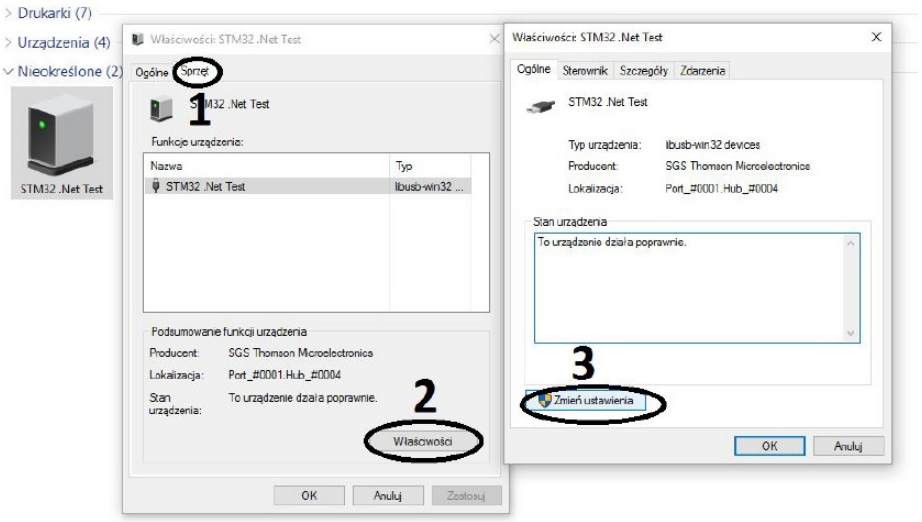
2.6 Jeżeli wszystko przebiegło prawidłowo powinny zapalić się 3 diody użytkowe. Podłącz kabel micro USB (muszą być podłączone oba).

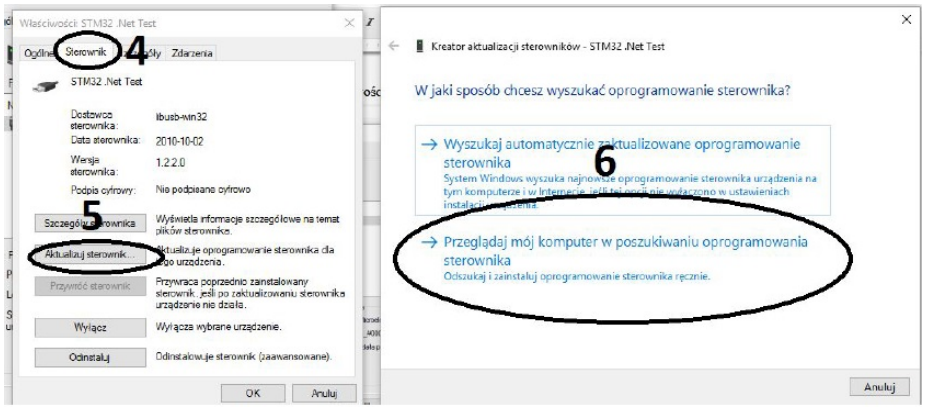


2.7 Przejdz do “urządzenia i drukarki”. Tam w obszarze“ nieokreślone” kliknij prawym przyciskiem myszy w “STM .Net Test” i wybierz właściowości.



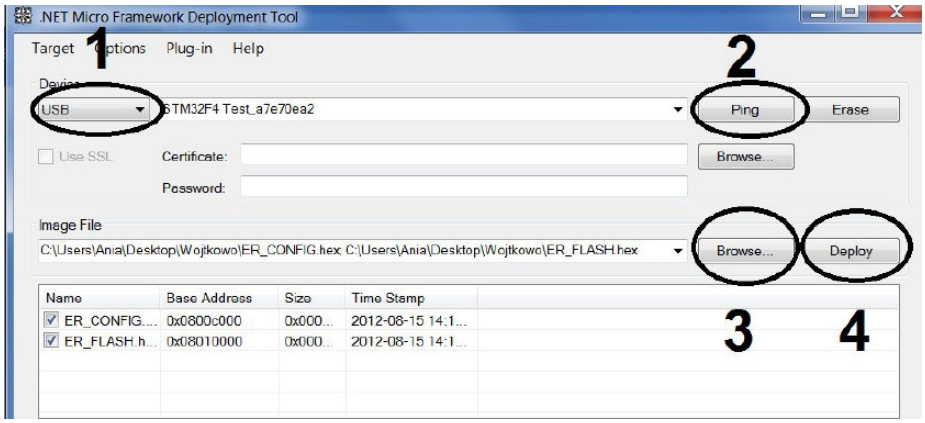
2.8 Wejdź w sprzęt >właściowości >zmień ustawienia >sterownik >Aktualizuj sterownik...



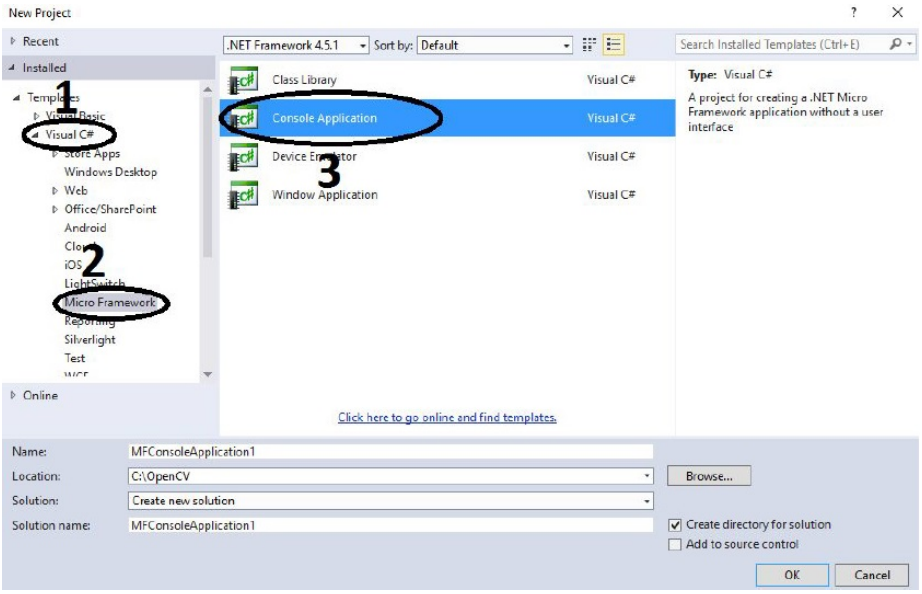


2.9 Wybierz "Przeglądaj mój komputer w poszukiwaniu oprogramowania sterownika" i wybierz ścieżkę gdzie rozpakowałłeś na początku sterownik. Podczas instalacji zignoruj ostrzeżenia.

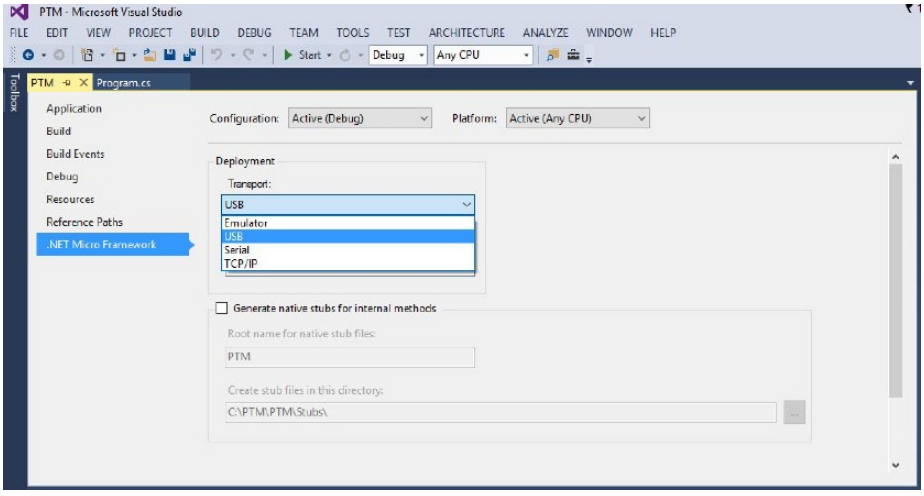
2.10 Teraz uruchom MFDeploy. Wybierz Device: USB. Naduś przycisk Ping. Następnie drugie od góry Browse... , wybierz ścieżkę pozostałych dwoch lików hex: ER_CONFIG.hex, ER_FLASH.hex oraz wybierz Deploy.



2.11 Włącz Visual studio utworz nowy projekt i wybierz C# >Micro Framework >Console Application.



2.12 W utworzonym projecie, w Solution Explorer kliknij prawym przyciskiem myszy na projekt i wybierz "Properties". Tam wybierz .NET Micro Framework i Transport ustaw na USB.



3 Przykłady

Wszystkie przykłady wymienione poniżej są dostępne na: https://github.com/PUT-PTM/STM-Csharp-tutorial. Każdy posiada dwie referencje: Microsoft.SPOT.Native oraz mscorlib, są one niezbędne do działania. Mscorlib zawiera zbiór przestrzeni nazw.

3.1 Przycisk

Obsługa przycisku oraz LED jest zaimplementowana w przykładzie "Button". Posiada on dodatkową referencję Microsoft.SPOT.Hardware, która definiuje klasy takie jak:

- InterruptPort - wykorzystana jako obsługa przycisku.
- OutputPort - wykorzystana jako obsługa LED.
- Cpu - wykorzystana do inicjacji przycisku oraz LED, poprzez określenie ich pinów.

Działanie programu

Sprawdza stan przycisku,w przypadku wciśnięcia przycisku zapala niebieską oraz czerwoną diodę, w innym przpadku zapala pomarańczową oraz zieloną.

Kluczowe funkcje

- void Write(bool) - Metoda klasy OutputPort(true-włącz LED, false-wyłącz LED)
- bool Read() - Funkcja klasy InterruptPort, zwraca true jeżeli przycisk jest wciśnięty.

3.2 LED

Obsługa LED zaimplementowana jest w przykładzie "LED_GPIO". Posiada on dodatkową referencję Microsoft.SPOT.Hardware, która definiuje klasy takie jak:

- OutputPort - wykorzystana jako obsługa LED.
- Cpu - wykorzystana do inicjacji LED, poprzez określenie ich pinów.

Działanie programu

W nieskończonej pętli wykonnuje natępujące czynności: włączenie wszystkich LED, czekanie na wykonanie pętli for, wyłączenie wszystkich LED, czekanie na wykonanie pętli for.

Kluczowe funkcje

- void Write(bool) - metoda klasy OutputPort(true-włącz LED, false-wyłącz LED)
- for (int i = 0; i ¡ 100000; i++) - prosta pętla służąca do zatrzymania programu.

3.3 PWM

Obsługa PWM oraz usypaiania wątku zaimplementowana jest w przykładzie "LED_PWM". Przykład posiada dwie dodatkowe referencje: Microsoft.SPOT.Hardware i Microsoft.SPOT.Hardware.PWM. Definiują one takie klasy jak:

- PWM - wykorzystana jako obsługa LED.
- Cpu - wykorzystana do inicjacji LED, poprzez określenie ich pinów.
- Thread - Wykorzystywana do usypiania wątku.

Działanie programu

W nieskończonej pętli, po kolei cykl pracy impulsów dla każdego LEDa ustawiany jest na liczbę 1, zmieniejszając go o 0,1 co sekundę, aż do 0.

Kluczowe funkcje oraz zmienna

- double DutyCycle() - zmienna klasy PWM. Określa cykla pracy impulsów jako ułamek. Liczba powinna być z przedziału od 0,0 do 1,0.
- void Sleep(int) - metoda klasy Thread. Usypia wątek na tyle milisekund ile zostało podanych w argumencie.
- void Start() - metoda klasyw PWM. Uruchamia port PWM na czas nieokreślony.

3.4 Zegar czasu rzeczywistego

Obsługa zegaru czasu rzecziwistego oraz debuggera zaimplementowana jest w przykładzie "RTC".

Działanie programu

W nieskończonej pętli spradzane są sekundy z aktualnego czasu. Raz na sekundę wykonuje pewne akcje. Jeżeli czas jest podzielny przez 2, pobiera aktualną liczbę tyknięc zegara, w następnej sekundzie sprawdza ile tyknięć wykonało się w czasie tej sekundy.

Kluczowa metoda oraz zmienne

- void Debug.Print(string) - metoda wyświetlająca podany argument w Visual Studio podczas debugowania
- int DateTime.Now.Second - zmienna przechowująca sekundy z aktualnego czasu. Może przyjmować wartości od 0 do 59
- long DateTime.Now.Ticks - zmienna przechowująca a ilość tyknięć zegara, które reprezentują datę i godzinę tej instancji. Może przyjmować wartości od DateTime.MinValue.Ticks do DateTime.MaxValue.Ticks.

3.5 SPI-Akcelerometr

Obsługa akcelerometru za pomocą SPI, LED oraz obsługa watków zaimplementowane są w przykładzie "SPI_Accelerometer". Posiada on dodatkową referencję Microsoft.SPOT.Hardware, która definiuje klasy takie jak:

- OutputPort - wykorzystana jako obsługa LED.
- Cpu - wykorzystana do inicjacji LED, poprzez określenie ich pinów.
- Thread - Wykorzystywana do usypiania wątku.
- SPI - wykorzysytwana do konfiguracji akcelerometru oraz do obsługi czytania z akcelerometru i zapisywania do akcelerometru.

Działanie programu

Odczytywanie aktualnego ułożenia przestrzennego urządzenia, i na podstawie tych dannych włączenie odpowiednich LED. Przechylenie w dół któregoś boku skutkuje zapaleniem odpowiedniego LED. Odwrócenie urządzenia o 180° spowoduje zapalenie wszystkich LED.

Kluczowe funkcje

- void Write(bool) - metoda klasy OutputPort(true-włącz LED, false-wyłącz LED)
- void Sleep(int) - metoda klasy Thread. Usypia wątek na tyle milisekund ile zostało podanych w argumencie.
- void WriteRegister(byte, byte) - metoda służy do zapisywania do akcelerometru. Pierwszy argument to adres rejestru, drugi to dane do zapisu.
- byte ReadRegister(byte) - funkcja służy do czytania odczytywania z akcelerometru. Zwraca dane z rejestru o adresie podanym w argumencie.

3.6 Timer

Obsługa timera oraz LED zaimplementowana jest wprzykładzie "Timer_Function". Posiada on dodatkową referencję Microsoft.SPOT.Hardware, która definiuje klasy takie jak:

- OutputPort - wykorzystana jako obsługa LED
- Cpu - wykorzystana do inicjacji LED, poprzez określenie ich pinów.

Działanie programu

Co sekundę timer się uruchamia i zmienia stan LED(włączone, wyłączone). Nieskończona pętla służy do tego aby program się nie zakończył.

Kluczowe metody oraz klasa

- void Write(bool) - metoda klasy OutputPort(true-włącz LED, false-wyłącz LED)
- void funTimer(object) - metoda wywoływana przez timer.
- class Timer - klasa definiująca timer. W konstruktor przyjmuje argumenty takie jak: metoda którą ma wywoływać, obiekt przechwoujący dane, opóźnienie z jakim ma być wywoływana metoda(w milisekundach), czas między wywoływaniami(w milisekundach).