Ćwiczenie 8: Tryby oszczędzania energii w mikrokontrolerze

Instrukcja laboratorium

Mariusz Chilmon <mariusz.chilmon@ctm.gdynia.pl>

2024-01-28

> First, solve the problem. Then, write the code.
>
> — *John Johnson*

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z:

- trybami oszczędzania energii,
- aktywowaniem oszczędzania energii,
- wychodzeniem z trybu oszczędzania energii.

Uruchomienie programu wyjściowego

1. Podłącz płytkę *WPSH209* do *Arduino Uno*.
2. Wyświetlacz odlicza wartości począwszy od `0000`.

Zadanie podstawowe

Celem zadania podstawowego jest włączenie trybu oszczędzania energii *Power-down*.

Wymagania funkcjonalne

1. Po dwóch sekundach odliczanie zatrzymuje się, a procesor przechodzi w stan *Power-down*.
2. Przed przejściem procesora w stan *Power-down* gaszony jest wyświetlacz.

Ponowne uruchomienie systemu możliwe jest po wciśnięciu przycisku `RESET`. Odliczanie zaczyna się jednak wtedy znów od wartości `0`.

Modyfikacja programu

Mikrokontroler przechodzi w stan oszczędzenia energii (uśpienia) po wywołaniu instrukcji `SLEEP`, zgodnie z konfiguracją rejestru `SMCR`.

Odczytaj w opisie rejestru `SMCR`, jakie bity należy ustawić przed wywołaniem instrukcji `SLEEP`, aby wejść w tryb najgłębszego oszczędzania energii, czyli *Power-down*.

Instrukcję asemblerową `SLEEP` wywołuje w C++ funkcja `sleep_cpu()`.

Mikrokontrolery posiadają na ogół wiele trybów oszczędzania energii, co pozwala pracować niektórym peryferiałom, a inne wyłączyć. Możliwe jest też wyłączenie niektórych elementów mikrokontrolera bez wchodzenia w tryb oszczędzenia energii. W trybie *Power-down* wyłączone są wszystkie sygnały zegarowe, dzięki czemu zużycie energii jest minimalne.

Samo uśpienie procesora nie zawsze wystarcza, by znacząco zredukować zużycie energii. Należy także zadbać o wyłączenie innych elementów systemu mikroprocesorowego. W przypadku naszego urządzenia kluczowe jest zgaszenie wyświetlacza. Aby to zrobić, należy wysłać do rejestrów przesuwnych bajt wypełniony zerami, dzięki czemu segmenty zostaną wygaszone. Bajt sterujący poszczególnymi cyframi można wypełnić wartościami 1, co też odpowiada wyłączeniu poszczególnych cyfr. Na koniec należy pamiętać o zatrzaśnięciu wpisanej wartości na wyjściach rejestrów przesuwnych:

```
1 shifter.shift(0xff); // Segmenty.
2 shifter.shift(0x00); // Cyfry.
3 shifter.latch(); // Zatrzask.
```

Procedurę usypiania umieść w funkcji `shutdown()`. Odmierzanie opóźnienia 2 sekund możesz zrealizować za pomocą funkcji bibliotecznej `_delay_ms()`. Pamiętaj, że jej użycie wymaga dołączenia odpowiedniego pliku nagłówkowego:

```
1 #include <util/delay.h>
```

Zadanie rozszerzone

Celem zadania rozszerzonego jest wyjście z trybu oszczędzania energii za pomocą przerwania.

Wymagania funkcjonalne

1. Po wciśnięciu przycisku *S1* procesor kontynuuje odliczanie.
2. Po dwóch sekundach od wybudzenia procesor ponownie się usypia.

Modyfikacja programu

Wznowienie działania nie jest możliwe z poziomu samego programu, gdyż jego wykonywanie jest wstrzymane. W trybie *Power-down* oscylator taktujący rdzeń procesora i peryferiały takie jak timery są całkowicie wyłączone, więc jedynym sposobem na interakcję z mikrokontrolerem jest dostarczenie sygnału ze świata zewnętrznego, np. w postaci impulsu na pinie wejściowym.

Impuls ten musi być, oczywiście, obsłużony przez przerwanie, gdyż pętla główna jest zatrzymana i nie można jej użyć do sprawdzania stanu wejścia.

Aby włączyć przerwanie od przycisku `S1` należy w funkcji `gpioInitialize()` włączyć przerwanie `PCINT1` dla wejścia `PCINT9`[1]:

```
1  PCMSK1 |= _BV(PCINT9);
2  PICR |= _BV(PCIE1);
```

Pamiętaj o zdefiniowaniu procedury obsługi przerwania `PCINT1_vect`, gdyż domyślna obsługa przerwania resetuje procesor.

[1] Tak, w tym mikrokontrolerze `PCINT` raz oznacza przerwanie od GPIO, a raz pin wywołujący to przerwanie…