Ćwiczenie 15: Synchronizacja przerwania z pętlą główną

Instrukcja laboratorium

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

2024-05-07

> Fix the cause, not the symptom.
>
> — *Steve Maguire*

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z:

- wyłączaniem przerwań,
- pojęciem atomowości operacji,
- muteksami.

Uruchomienie programu wyjściowego

1. Podłącz płytkę *LCD Keypad Shield* do *Arduino Uno*.
2. Podłącz termometr LM35DZ do linii *A5*.
3. Na wyświetlaczu widoczny jest odczyt wartości z ADC, mierzącego napięcie z termometru scalonego LM35DZ, obliczona na tej podstawie temperatura oraz liczba błędów odczytu danych z przerwania.

Rysunek 1: Przykładowy stan wyświetlacza

W przerwaniu `ADC_vect`, które wywoływane jest przez przetwornik analogowo-cyfrowy po zakończeniu każdego pomiaru, następuje odczyt tego pomiaru oraz przeliczenie go na temperaturę. Pomiar zapisywany jest w zmiennej globalnej `measurement` zaś temperatura w zmiennej `temperature`. Obie zadeklarowane są z użyciem kwalifikatora typu **`volatile`**, co gwarantuje, że funkcja `printMeasurement()`, wywoływana z pętli głównej, nie będzie cache'ować ich zawartości w rejestrach, ale zawsze dokona odczytu aktualnej wartości z pamięci RAM.

Niestety, nie gwarantuje to odczytu poprawnych wartości. Odczyt należy rozumieć tu jako skopiowanie bajtów z obszaru pamięci operacyjnej, w której przechowywane są obie zmienne globalne, do obszaru pamięci zmiennych lokalnych lub do wybranych przez kompilator rejestrów. Nie jest możliwy odczyt zmiennej typu `uint16_t` lub **`double`** w jednej instrukcji, gdyż zajmują one więcej niż jeden bajt. Tym samym możliwe jest, że nastąpi skopiowanie części danych z pomiaru w chwili t_0, następnie proces

kopiowania zostanie przerwany procedurą `ADC_vect`, po czym będzie kontynuowany, ale z użyciem nowego pomiaru z chwili t_1.

Ponieważ w przypadku pojedynczych zmiennych trudno jest ocenić, czy odczyt jest poprawny (złożony z jednego pomiaru) czy błędny (złożony z dwóch pomiarów), szczególnie, gdy każda wartość z zakresu zmiennej może być prawidłowa. Z tego powodu w przykładowym programie sprawdzamy, czy pomiar i obliczona temperatura są ze sobą zgodne (odczyt z jednego pomiaru) czy nie (odczyt z dwóch pomiarów).

W celu zwiększenia prawdopodobieństwa wystąpienia błędu wprowadzono zmienną `slowDown`, z użyciem której zmniejszono częstotliwość odświeżania wyświetlacza, dzięki czemu średnia szybkość wykonywania pętli głównej znacząco wzrosła, a wraz z nią — prawdopodobieństwo przerwania odczytu przez procedurę `ADC_vect`.

Zadanie podstawowe

Celem zadania podstawowego jest poprawienie działania programu przez użycie bloku atomowego.

Wymagania funkcjonalne

1. Funkcjowanie urządzenia nie zmienia się.
2. Liczba błędów spada do zera.

Modyfikacja programu

Czasowe blokowanie przerwań

Użyj funkcji `cli()` i `sei()`, by wyłączyć przerwania w funkcji `printMeasurement()` na czas odczytu i włączyć je z powrotem po zakończeniu tego procesu. To powinno całkowicie wyeliminować błędy odczytu, gdyż odczyt danych nie będzie przerywany obsługą przerwania.

Użycie bloku atomowego

Zastosowanie pary `cli()` / `sei()` ma trzy wady (wymienione od najmniej ważnej):

1. Blok wykonywany przy wyłączonych przerwaniach nie jest wyraźnie wydzielony w strukturze kodu.

2. Kompilator nie jest świadomy zależności między nimi, nie wyłapuje więc błędu polegającego na wyłączeniu przerwań na stałe (braku wywołania `sei()`).
3. Jeżeli nie znamy sposobu obsługi przerwań w obrębie całego programu, możemy doprowadzić do niezamierzonego włączenia przerwań, gdy użyjemy tej pary w sytuacji, gdy przerwania są już wyłączone.

Problemy te można rozwiązać, stosując makro `ATOMIC_BLOCK()`. Przyjmuje ono jako argument dwie opcje:

ATOMIC_FORCEON Po zakończeniu bloku przerwania są zawsze włączane, tak samo jak w naszym rozwiązaniu z parą `cli()` / `sei()`.

ATOMIC_RESTORESTATE Po zakończeniu bloku przerwania są przywracane do stan zapisanego przed wejściem do bloku.

Użyj drugiej opcji — bardziej uniwersalnej choć wymagającej więcej instrukcji.

W programowaniu operacja atomowa to operacja niepodzielna — wykonuje się w całości lub w ogóle. Przykładem takich operacji są operacje w bazach danych czy operacje bankowe — przelew między kontami nie może wykonać się częściowo, np. zmniejszając saldo nadawcy, ale nie zwiększając salda odbiorcy.

Zadanie rozszerzone

Celem zadania rozszerzonego jest synchronizacja przerwania z pętlą główną za pomocą zmiennej pełniącej rolę semafora.

Wymagania funkcjonalne

1. Funkcjowanie urządzenia nie zmienia się.

Modyfikacja programu

Usunięcie bloku atomowego

Cofnij zmiany wprowadzone w zadaniu podstawowym — usuń makro `ATOMIC_BLOCK()`, pozostawiając odczyt zmiennych globalnych w funkcji `printMeasurement()`.

Komunikacja za pomocą semafora

Dodaj do programu zmienną globalną typu, który mieści się w jednym bajcie, np. `uint8_t`. Taką zmienną można uznać za atomową, ponieważ zarówno odczyt jak i zapis wykonuje się w jednym takcie zegara.

Instrukcje w procedurze obsługi przerwania należy obwarować zależnością np. od stanu **`false`** zmiennej, a po ich wykonaniu zmienić stan zmiennej na przeciwny, np.:

```
1 if (semaphore == false) {
2     // Interrupt instructions.
3     semaphore = true;
4 }
```

Zapis do zmiennych z wartością pomiaru i temperatury nastąpi więc tylko, gdy semafor będzie miał wartość **`false`**, a zmiana jego stanu na **`true`** poinformuje pętlę główną, że może dokonać odczytu:

```
1 if (semaphore == true) {
2     // Main loop instructions.
3     semaphore = false;
4 }
```