Ćwiczenie 4: Multipleksowanie wyświetlacza siedmiosegmentowego

Instrukcja laboratorium

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

2024-01-14

> If the comment and code disagree, both are probably wrong.
>
> — *Bjarne Stroustrup*

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z:

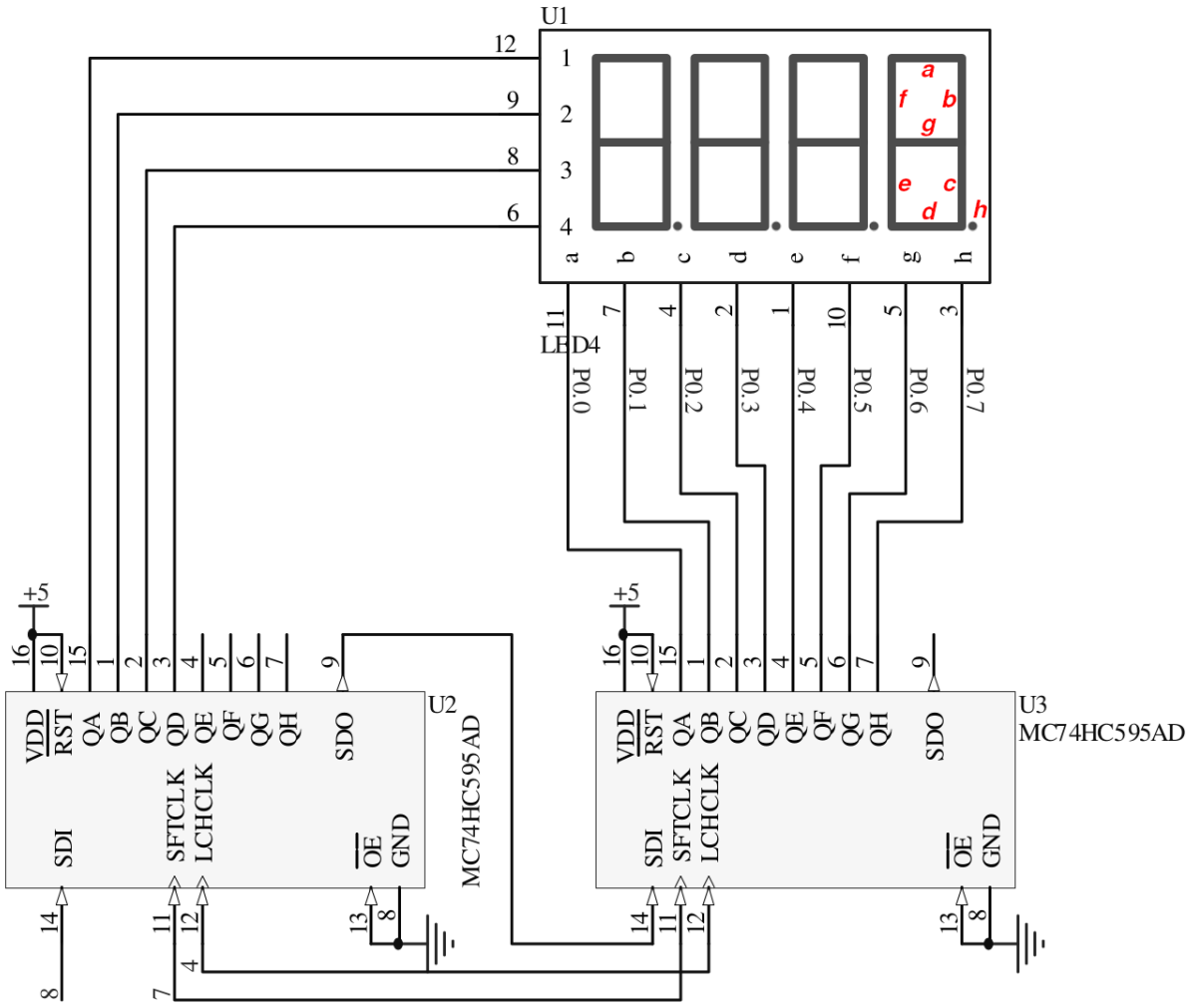
- wykorzystaniem rejestru przesuwnego,
- definiowaniem cyfr na wyświetlaczu siedmiosegmentowym,
- multipleksowaniem wyświetlacza siedmiosegmentowego,
- precyzyjną regulacją częstotliwości pracy timera.

Uruchomienie programu wyjściowego

1. Podłącz płytkę *WPSH209* do *Arduino Uno*.
2. Na kolejnych polach wyświetlacza LED powinny wyświetlać się cyfry.

Multipleksowanie polega na tym, że np. czterocyfrowy wyświetlacz złożony łącznie z $8 \cdot 4 = 32$ segmentów (wliczając kropki dziesiętne) można obsłużyć za pomocą znacznie mniejszej liczby pinów mikrokontrolera. W przypadku wyświetlaczy siedmiosegmentowych LED na ogół wykorzystuje się część pinów do włączania na krótki czas pojedynczych cyfr i 8 pinów do włączania segmentów w aktualnie wybranej cyfrze, co w naszym przykładzie oznacza zapotrzebowanie na $8 + 4 = 12$ pinów.

Poprzez multipleksowanie obsługiwane są różne rodzaje wyświetlaczy, a także klawiatur.

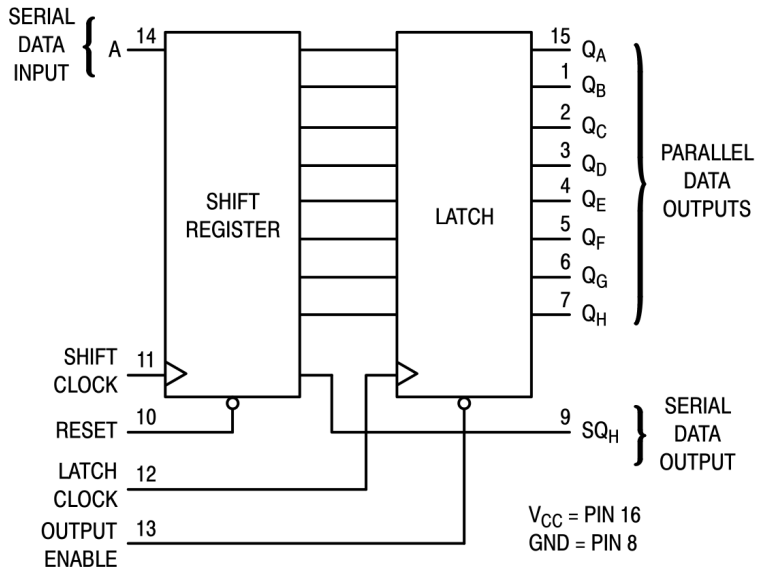


Rysunek 1: Sterowanie wyświetlaczem siedmiosegmentowym

Zapotrzebowanie to można jeszcze znacząco zmniejszyć używając rejestru przesuwnego. Jest to układ cyfrowy, który posiada jedno wejście danych i wiele wyjść. Dane wprowadzane są szeregowo, w takt sygnału zegarowego, a po podaniu sygnału zatrzaskującego *latch* przepisywane są na wyjścia.

Przykładowo, jeżeli rejestr przesuwny ma 8 wyjść, należy podać 8 taktów zegara, by wprowadzić pełny zestaw danych. Rejestr przesuwny posiada wyjście przeniesienia, które można podłączyć do wejścia kolejnego rejestru, dzięki czemu podając 16 taktów zegara możemy wprowadzić łącznie 16 bitów danych do obu rejestrów i wysterować 16 wyjść po podaniu sygnału zatrzaskującego.

Pierwsze rejestry przesuwne zostały skonstruowane w latach 40. XX wieku na lampach elektronowych. Używane są do dzisiaj w szczególności w obwodach SerDes (*Serializer/Deserializer*), obsługujących magistrale komunikacyjne, np. Ethernet i HDMI.



Rysunek 2: Schemat blokowy rejestru przesuwnego MC74HC595A

Połączenie multipleksowania oraz rejstrów przesuwnych pozwala obsłużyć $8 \cdot 4 = 32$ segmenty wyświetlacza za pomocą zaledwie 3 pinów procesora. Oczywiście, ceną za to jest konieczność oprogramowania całego procesu, który trwa co najmniej kilkadziesiąt taktów zegara mikrokontrolera.

Zadanie podstawowe

Celem zadania podstawowego jest uruchomienie wyświetlania multipleksowanego i uzyskanie odmierzania czasu z rozdzielczością 1 s.

Wymagania funkcjonalne

1. Częstotliwość multipleksowania wynosi 200 Hz, a wyświetlane liczby zmieniają dokładnie co sekundę.
2. Wszystkie cyfry są wyświetlane prawidłowo.

Wymagania jakości kodu

1. Wartość rejestru `OCR1A` powinna być zdefiniowana w sposób zależny od częstoliwości taktowania mikrokontrolera `F_CPU` i stałej `TIMER_FREQUENCY`, określającej żądaną

częstotliwość multipleksowania 200 Hz.

Dzięki uzależnieniu nastaw timera od stałych zdefiniowanych w kodzie można zarówno łatwo zmienić częstotliwość uzyskiwaną z timera, jak i zachować dotychczasową częstotliwość po zmianie taktowania całego mikrokontrolera (np. w celu obniżenia poboru prądu).

Modyfikacja programu

Aby precyzyjnie ustawić częstotliwość pracy timera, wykorzystano tryb CTC (*Clear Timer on Compare Match*), który pozwala regulować górną granicę odliczania timera za pomocą rejestru `OCR1A`. W tym trybie wykorzystujemy nie przerwanie przepełnienia timer (`TIMER_OVF`), ponieważ nie dochodzi do przepełnienia timera, ale przerwanie komparatora (`TIMER1_COMPA` w pliku `main.cpp`).

Wzór na częstotliwość przebiegu uzyskiwanego w timerze:

$$f = \frac{f_{clkI/O}}{N(1 + OCR1A)} \tag{1}$$

Należy go przekształcić tak, by uzyskać wzór na wartość rejestru $OCR1A$. Żądana częstotliwość f jest zdefiniowana w stałej `TIMER_FREQUENCY`, N to wartość dzielnika preskalera, zaś częstotliwość zegara systemowego $f_{clkI/O}$ — w stałej `F_CPU`.

W zadaniu podstawowym należy wprowadzić zmiany w pliku `timer.cpp` w obrębie funkcji `timerInitialize()`.

Opis trybu CTC znajduje się w rozdziale *16-bit Timer/Counter1 with PWM* w sekcji *Clear Timer on Compare Match (CTC) Mode* dokumentacji mikrokontrolera.

Zadanie rozszerzone

Celem zadania rozszerzonego jest odmierzanie czasu z dokładnością do 0,1 s.

Wymagania funkcjonalne

1. Liczby są inkrementowane co 0,1 s.
2. Zaświecona jest kropka między pierwszą a drugą liczbą (licząc od prawej).

Modyfikacja programu

Aby zwiększyć częstotliwość odliczania należy zmodyfikować funkcję `stopwatchTick()` w pliku `stopwatch.cpp`. Jest ona wywoływana w przerwaniu timera, a więc z częstotliwością 250 Hz.

W celu włączenia kropki dziesiętnej na odpowiednim wyświetlaczu należy zmodyfikować funkcję `timeToDisplay()` w pliku `stopwatch.cpp`.