Systemy wbudowane

Lista zadań nr 6

19 i 25 listopada 2025

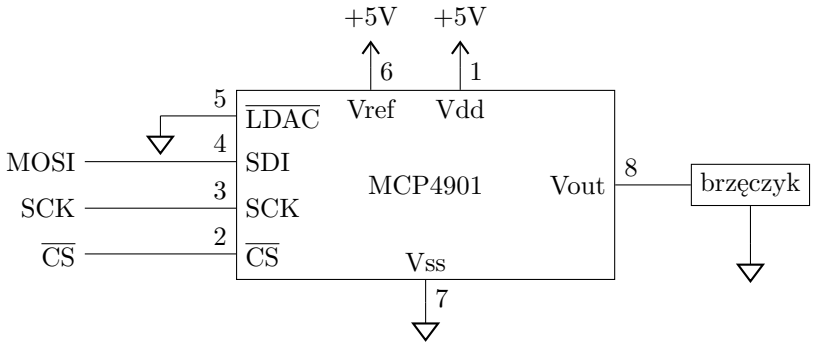
Rozwiązania należy zaprezentować najpóźniej w dniu, w którym odbywa się pracownia. Najpóźniej w tym samym dniu należy również przekazać kod źródłowy rozwiązań na SKOS. Pliki należy nazwać w czytelny sposób, podpisać w komentarzu w treści pliku, oraz przesłać jako oddzielne pliki na SKOS – bez archiwizacji.

1. Napisz jak najprostszy program, który będzie wykonywał za pomocą UART funkcję „echo": każdy odebrany przez układ znak powinien zostać przesłany z powrotem do nadawcy. Nie używaj funkcji z biblioteki standardowej języka C (`printf`, `scanf`, `getchar` itp.). Wykorzystaj przerwania, pętla główna programu powinna wyłącznie usypiać mikrokontroler do stanu bezczynności (głębsze tryby uśpienia wyłączają układ UART).

2. Przykładowa implementacja wejścia/wyjścia przez UART z pliku `helloworld.c` wykorzystuje aktywne czekanie, co ma pewne wady. Podczas nadawania i odbierania procesor większość czasu spędza na sprawdzaniu bitu statusu, a odbieranie działa poprawnie tylko wtedy, gdy program wykonuje funkcje odczytywania. Popraw te wady, implementując rozwiązanie oparte na przerwaniach, wykorzystujące dwa bufory cykliczne: nadawania i odbierania.

   Napisz procedury obsługi przerwań `USART_RX` i `USART_UDRE`[1] oraz nowe implementacje funkcji `uart_transmit` i `uart_receive` unikające aktywnego czekania. Funkcja nadająca może czekać tylko wtedy, jeśli bufor nadawania jest pełny, zaś funkcja odbierająca tylko wtedy, gdy bufor odbierania jest pusty. Pamiętaj o prawidłowej synchronizacji funkcji z procedurami obsługi przerwań przy użyciu maskowania. Przetestuj swoje rozwiązanie.

3. Zbuduj następujący układ wykorzystujący układ DAC MCP4901. Zwróć uwagę na prawidłowe podłączenie układu: nóżka numer 1 jest oznaczona kropką, kolejne nóżki są liczone przeciwnie do kierunku wskazówek zegara. Nieprawidłowe podłączenie linii zasilających **uszkodzi** układ.



   Zapisz w swoim programie krótki, jednokanałowy plik audio w formie ciągu próbek, używając niskiej częstotliwości próbkowania (np. 8 kHz); ze względu na rozmiar dane mogą być umieszczone wyłącznie w pamięci flash mikrokontrolera. Przykładowy plik jest na SKOS – plik `dzwiek.c`. Aby przygotować własny, w systemie Linux można użyć np. następujących komend:

```
$ ffmpeg -i dzwiek.wav -f u8 -ar 8000 -ac 1 -acodec pcm_u8 dzwiek.raw
$ xxd -i dzwiek.raw dzwiek.c
```

   Odtwórz ten plik używając układu DAC sterowanego transceiverem SPI.

   Wgranie nowej próbki do układu MCP4901 polega na: zmianie stanu pinu $\overline{\text{CS}}$ na niski, przesłanie dwóch bajtów przy użyciu SPI (w trybie CPOL=0 CPHA=0) zgodnie z opisem w nocie katalogowej[2], zmianie stanu pinu $\overline{\text{CS}}$ na wysoki. Należy przeczytać ze zrozumieniem opis 16-bitowego rejestru komend w nocie

[1] Przerwanie `UDRE` jest wyzwalane, gdy UART jest gotowy do przyjęcia nowego znaku do nadawania w rejestrze `UDR`.
[2] `http://ww1.microchip.com/downloads/en/DeviceDoc/22248a.pdf`

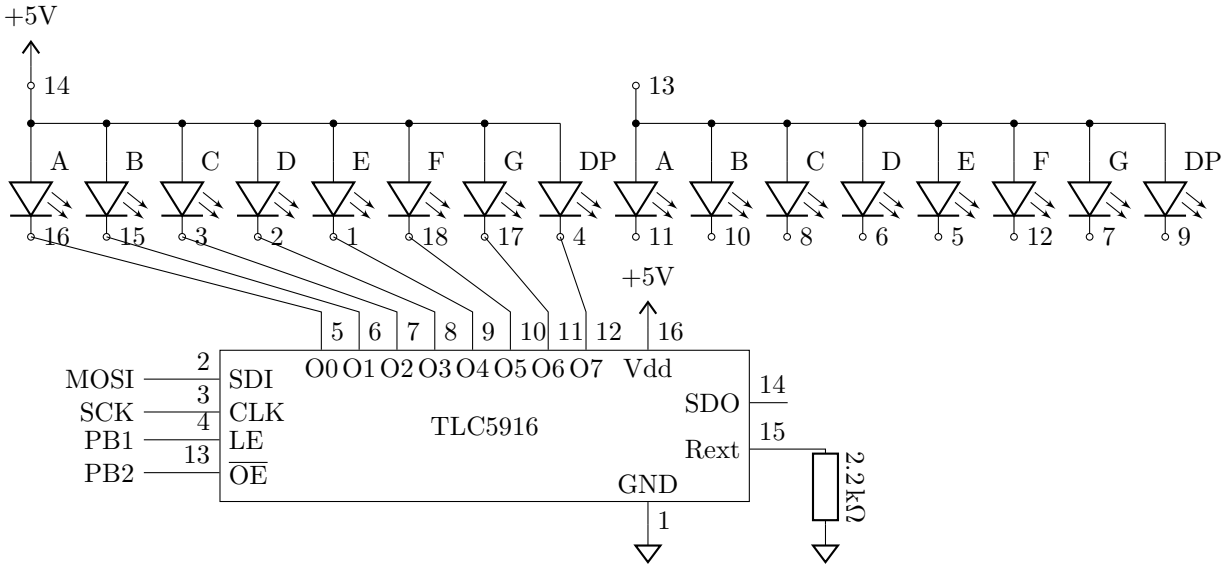
katalogowej MCP4901 – w szczególności nie należy zapomnieć o bitach $\overline{\text{GA}}$ (wzmocnienie powinno być ustawione na 1x) oraz $\overline{\text{SHDN}}$ (układ ma być załączony).

4. Zbuduj następujący układ wykorzystujący wyświetlacz 7-segmentowy oraz układ sterownika LED TLC5916. Wyświetl na wyświetlaczu co sekundę kolejne cyfry od 0 do 9. Wykorzystaj transceiver SPI do sterowania układem sterownika LED. Sposób sterowania tym układem jest następujący:

   - Po przesłaniu stanu diod przy użyciu SPI należy przesłać krótki impuls na pin LE. Stan diod LED jest ładowany z rejestru przesuwnego po zboczu opadającym.
   - Stan wysoki pinu $\overline{\text{OE}}$ wyłącza świecenie diod LED.

   Szczegóły są opisane w nocie katalogowej[3].

   Zwróć uwagę na prawidłowe podłączenie układu. Nieprawidłowe podłączenie linii zasilających **uszkodzi** układ.



[3] https://www.ti.com/lit/ds/symlink/tlc5917.pdf