Systemy wbudowane

Lista zadań nr 8

9, 10 i 11 grudnia 2024

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 program używający FreeRTOS wykonujący dwa zadania:

   - Wyświetlanie na diodzie LED stanu naciśnięcia przycisku sprzed 1 sekundy (zadanie 1 z listy 2),
   - Efekt przewijającej się diody LED (zadanie 3 z listy 1).

   Stan przycisku wystarczy regularnie odpytywać, nie trzeba używać przerwań.

2. Napisz program używający FreeRTOS wykonujący dwa komunikujące się zadania, wykonujące następujące operacje:

   - Odczytywanie wartości liczbowych przez UART i wysyłanie ich na kolejkę FreeRTOS (funkcje o nazwach rozpoczynających się od `xQueue`),
   - Odbieranie z kolejki wartości liczbowych i zapalanie diody przez tyle milisekund, ile wynosi odebrana wartość, pomiędzy zapaleniami powinna być widoczna przerwa.

   Ze względu na prostą przykładową implementację UART używającą aktywnego czekania, należy ustawić priorytety tak, aby zapalanie diody nie czekało na UART.

3. Funkcje obsługi UART dołączone do przykładowego programu FreeRTOS wykorzystują aktywne czekanie. W systemie wielozadaniowym jest to marnotrawstwo zasobów: proces czekający na wejście będzie spowalniać wykonanie procesów o tym samym priorytecie i uniemożliwiać wykonanie procesów o priorytecie niższym. Zaimplementuj w FreeRTOS funkcje `uart_transmit` i `uart_receive` tak, aby uniknąć aktywnego czekania. Wskazówki:

   - Wykorzystaj przerwania UART (`USART_RX`, `USART_UDRE`);
   - Do komunikacji pomiędzy procedurami obsługi przerwań a zadaniami FreeRTOS wykorzystaj kolejki FreeRTOS (używaj funkcji o nazwach rozpoczynających się od `xQueue`);
   - Niedopuszczalne jest aktywne czekanie oraz blokowanie w procedurach obsługi przerwań! Zamiast funkcji `xQueueReceive`/`xQueueSend` korzystaj tam z funkcji `xQueueReceiveFromISR`/`xQueueSendFromISR`.

   Sprawdź poprawność swojego rozwiązania przy użyciu dwóch zadań. Pierwsze z nich, o wyższym priorytecie, powinno wykonywać operacje wejścia i wyjścia (np. `scanf` i `printf`). Drugie, o niższym priorytecie, powinno migać wbudowaną diodą. W prawidłowo działającym rozwiązaniu operacje wejścia i wyjścia powinny nie wpływać na miganie diody.

4. Zaimplementuj mechanizm współdzielenia dostępu do ADC nie wykorzystujący aktywnego czekania. W tym celu napisz funkcję `uint16_t readADC(uint8_t mux)`, która, wywołana z zadania FreeRTOS, zwróci wynik pomiaru ADC z wejścia o numerze `mux`. Na czas wykonywania pomiaru zadanie powinno zostać zablokowane (przejść do stanu Blocked). Zaimplementuj też procedurę obsługującą przerwania ADC, współpracującą z funkcją `readADC`, która będzie budzić zadanie oczekujące na wynik. Funkcja powinna działać w sytuacji, gdy wiele zadań zażąda pomiaru ADC (być może z różnych wejść); w takiej sytuacji pomiary powinny zostać wykonane po kolei. Synchronizację zaimplementuj przy użyciu semaforów i/lub muteksów, nie używaj kolejek.

   Program testowy powinien uruchamiać trzy zadania, z których każdy wykonuje w różnych odstępach czasu pomiary ADC na różnych wejściach (np. z podłączonym potencjometrem, termistorem i fotorezystorem) i wypisujące ich wyniki przez UART.