

Architektura systemów komputerowych

Podczas realizacji poniższych zadań, oprócz instrukcji/dokumentacji podanych na końcu ćwiczeń (sekcja „Materiały”), można korzystać z dowolnych źródeł internetowych. Szczególnie polecany przez prowadzącego jest kurs dostępny na stronie: <https://forbot.pl/blog/kurs-stm32-l4-wstep-spis-tresci-dla-kogo-jest-ten-kurs-id48575>.

Uwaga: Po wykonaniu każdego zadania należy zapisać, kody i inne informacje do sprawozdania (zrzuty ekranu, odpowiedzi na pytania), o ile są potrzebne w danym zadaniu. Po wykonaniu wszystkich zadań niniejszej instrukcji należy powiadomić prowadzącego.

Instrukcja II

UART, IT, IDLE, DEBUG

Wstęp

Uniwersalny asynchroniczny nadajnik-odbiornik, znany ze skróconej nazwy UART, to jeden z podstawowych sposobów komunikacji urządzeń. Wysyłanie i odbieranie danych może być realizowane w czterech trybach:

- blokującym (polling),
- przerwaniowym,
- przerwaniowym z wykrywaniem stanu bezczynności IDLE (tylko odbieranie),
- DMA (bezpośredni dostęp do pamięci z pominięciem procesora).

Przebieg ćwiczenia

1. W projekcie z poprzednich zajęć ustaw w konfiguratorze komunikację UART (kurs Forbot - lekcja #4). Skorzystaj z modułu LPUART1. Ustaw tryb komunikacji asynchroniczny, prędkość komunikacji (baud rate) na 57600 bit/s. W ustawieniach zaawansowanych wyłącz opcję Overrun. Zapobiegnie to nadpisywaniu nieprzetworzonych danych w buforze.
2. Tryb blokujący (polling):
 - a) Wyślij wiadomość „**Hello from Nucleo!**” na przejściówkę USB-UART wbudowaną na płycie Nucleo64. W celu odebrania wysłanych danych uruchom terminal **hterm** i połącz z dostępnym portem COM. Ustaw parametry komunikacji zgodne z wprowadzonymi w konfiguratorze CubeMX. Wywołaj funkcję wysyłającą dane w taki sposób, żeby długość tekstu była automatycznie obliczana (bez „hardkodowania”). Do czego służy parametr `timeout`? Jaką maksymalną wartość może przyjąć?
Podpowiedź: Nazwę uchwytu portu UART (pierwszy parametr funkcji) można odczytać w kodzie automatycznie wygenerowanej funkcji `MX_LPUART1_UART_Init`.
 - b) Zmodyfikuj kod z zad. 2a, aby wiadomość była wysyłana za każdym razem, kiedy naciśnięty zostanie przycisk (tuż po jego puszczeniu). Odbieranie danych powinno działać w pętli nieskończonej.
 - c) Zmodyfikuj kod z zad. 2b, aby umożliwiał wysyłanie informacji kodem Morse’a. W przypadku krótkiego wciśnięcia przycisku powinien zostać wysłany znak kropki, a w przypadku długiego przytrzymania (co najmniej 500 ms) – znak kreski.
Podpowiedź: Skorzystaj z funkcji `HAL_GetTick` z poprzedniego ćwiczenia.

- d) Odbieraj w pętli 5-bajtowe wiadomości wysłane przez terminal, a następnie odsyłaj je z powrotem. Jak zachowa się program w przypadku wysłania wiadomości krótszej lub dłuższej niż pięć znaków?
 - e) Napisz program odbierający wiadomości wysłane przez terminal. Gdy odebrana zostanie wiadomość „1”, zapalaj diodę. Gdy odebrana zostanie wiadomość „0”, gaś diodę. Każdorazowo odpowiadaj na przesłane dane komunikatem „OK_” lub „ERROR_” (jeśli otrzymano komunikat inny niż 1/0).
 - f) Napisz program odbierający jednoznakowe wiadomości wysłane przez terminal. Odebraną wiadomość każdorazowo konwertuj na liczbę całkowitą, a następnie zaświeć i zgaś diodę (z dowolną częstotliwością) tyle razy, ile wynosiła podana liczba. Jeśli odebrana wiadomość nie jest liczbą, to zamiast migania diody wysyłaj w odpowiedzi komunikat „ERROR_”
3. Tryb przerwaniowy:
- a) Dodaj do projektu obsługę przerw od LPUART1, jeżeli nie jest włączona (kurs Forbot - lekcja #7). W kontrolerze przerw **NVIC** zezwól na globalne przerwanie od wybranego UART.
 - b) Zmodyfikuj kod z zadania 2e o wykorzystanie funkcji wysyłającej dane w trybie przerwaniowym. Tym razem funkcja odbierająca dane nie powinna być wywoływana w pętli (aby nie obciążać niepotrzebnie procesora).
 UWAGA: Przerwania mogą pochodzić od dowolnego modułu UART (w przyszłości możemy wprowadzić ich obsługę), a więc na początku definicji funkcji obsługi przerwania należy sprawdzać, czy pochodzi ono z modułu, z którego korzystamy, czyli domyślnie LPUART1.
4. Tryb przerwaniowy z wykrywaniem stanem bezczynności:
- Napisz program, który podobnie jak w zadaniu 2d odbiera wiadomość wysłaną przez terminal i odsyła ją z powrotem. Tym razem jednak skorzystaj z przerw i zadбай o to, aby można było przysyłać wiadomości o dowolnej długości (nieprzekraczającej 100 znaków). W tym celu należy użyć funkcję odbierającą dane z wykrywaniem stanu bezczynności: `HAL_UARTEx_ReceiveToIdle_IT` i odpowiadającą jej funkcję obsługi przerwania: `HAL_UARTEx_RxEventCallback`. Umożliwiają one łatwy odbiór danych o zmiennej długości.
- W jakich dwóch przypadkach zgłoszone zostaje przerwanie po wywołaniu funkcji `HAL_UARTEx_ReceiveToIdle_IT`?

Funkcje

- **Tryb blokujący**
 - `HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)`
 - `HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)`
- **Tryb przerwaniowy**
 - `HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)`
 - `HAL_StatusTypeDef HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)`
 - `void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)`
 - `void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)`

- **Tryb przerwaniowy z wykrywaniem stanu bezczynności**
 - HAL_StatusTypeDef HAL_UARTEx_ReceiveToIdle_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
 - void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)

Materiały

- „STM32CubeIDE - poradnik” – dokument dostępny na stronie prowadzącego w folderze przedmiotu Architektura systemów komputerowych.
- [STM32G4 Series advanced Arm®-based 32-bit MCUs - Reference manual](#)
- [Datasheet - STM32G491xC STM32G491xE - Arm® Cortex®-M4 32-bit MCU+FPU, 170 MHz / 213 DMIPS, up to 512 KB Flash, 112 KB SRAM, rich analog, math accelerator](#)
- [Description of STM32G4 HAL and low-layer drivers - User manual](#)