Laboratorium Techniki Mikroprocesorowej II

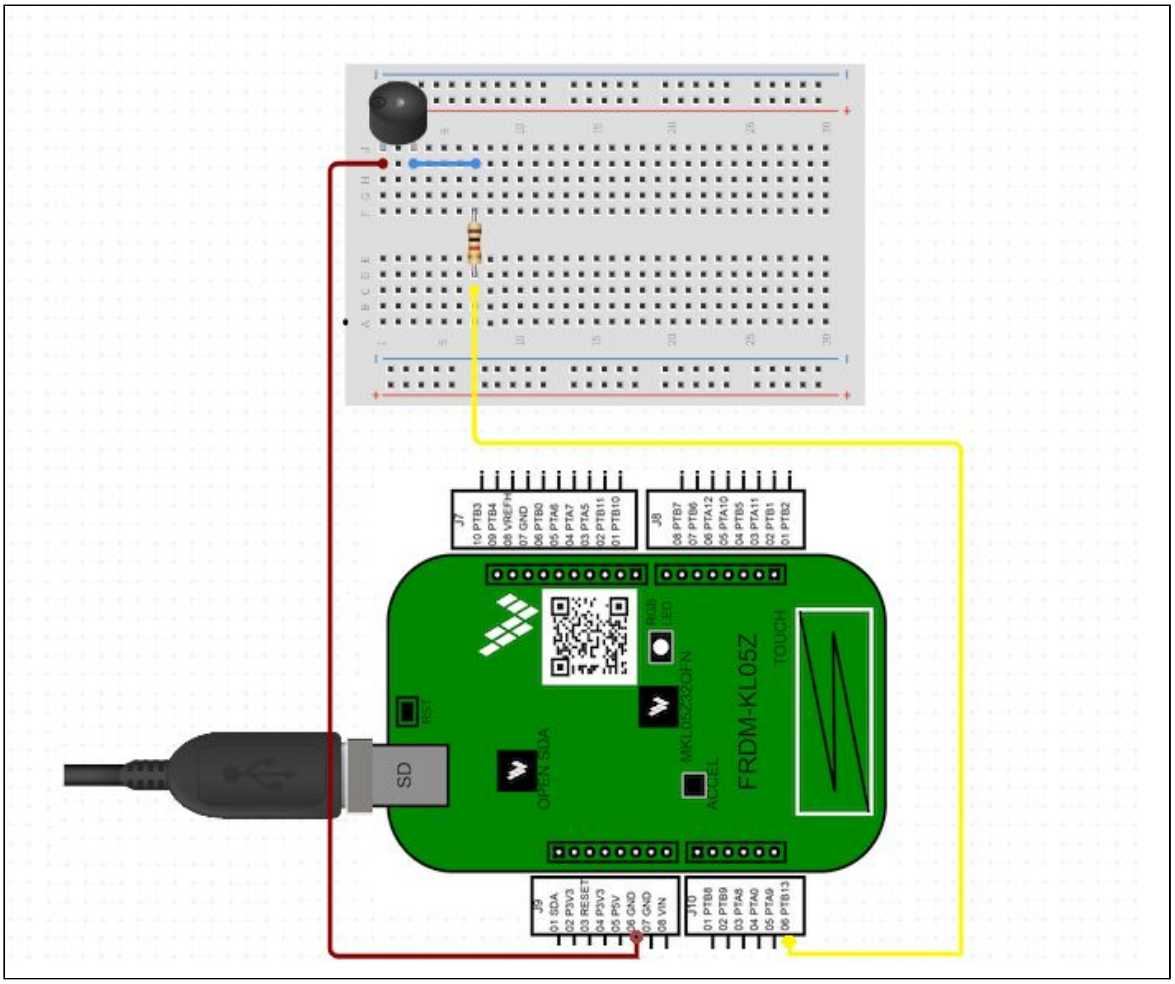
Sprawozdanie z projektu - Programowalny budzik RTC z komunikacją UART

Kamil Kaliś, Radosław Skałbania

20.01.2020

1. Wstęp

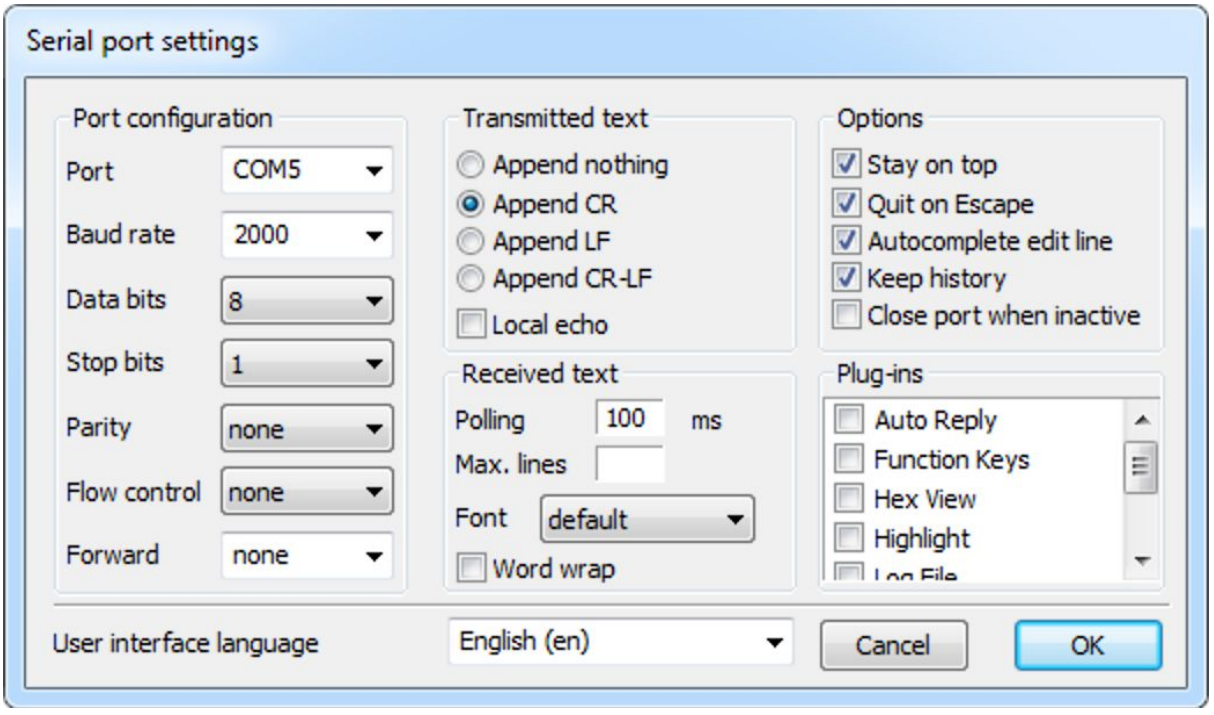
Tematem projektu jest budzik wgrany do pamięci mikrokontrolera FRDM-KL05Z firmy NXP. Jest on sterowany z poziomu terminalu UART, sygnalizujący alarm za pomocą migającej diody led o kolorze czerwonym oraz wydającego dźwięk z buzzera. Całość jest kontrolowana za pomocą programu Termite. Budowa układu jest niezbyt skomplikowana, co potwierdza schemat podany poniżej.



Pierwsza z nóżek buzzera jest podpięta do pinu 6 (GND), który odgrywa rolę masy. Druga, poprzedzona rezystorem 1kΩ jest podpięta do portu B na pinie 13, jest ona przewodnikiem sygnału generowanego z mikrokontrolera. Cały układ zasilany jest przez port USB.

2. Instrukcja obsługi

Do komunikacji i interfejsu graficznego użytkownika używany jest port szeregowy emulowany przez OpenSDA. Do otwarcia portu szeregowego na systemie Windows można użyć programu Termite oraz zmienić ustawienia jak na rysunku poniżej.



Po zestawieniu połączenia płytki z komputerem należy użyć przycisku RESET (Sw1) na płytce.

Program powinien przywitać użytkownika oraz poprosi o wprowadzenie aktualnej daty.
Ważne jest aby zapewnić poprawny format:

<dzień>/<miesiąc>/<rok(ostatnie 2 cyfry)>|<godzina>:<minuty>:<sekundy>

Każde pole powinno zawierać 2 cyfry i zostać rozdzielone dowolnym separatorem.

Przykład poprawnie wprowadzonej daty:
20/01/20|14:20:40

W przypadku niepoprawnie wprowadzonej daty, program poinformuje nas o tym – należy wtedy rozpocząć jeszcze raz wciskając RESET.

Jeśli wszystko zostanie poprawnie skonfigurowane, można rozpocząć użytkowanie budzika.
Użytkownik ma dostępne menu z opcjami – przycisk, którego należy użyć aby wykonać akcję zawarty jest między dwoma znakami, odpowiednio “<” i ”>”.

Przed ustawieniem budzika dostępne są dwie opcje:

- <s> – ustawienie budzika
- <t> – wyświetlanie aktualnego czasu

Użycie <t> będzie wyświetlać co sekundę aktualny czas. Aby przerwać wyświetlanie czasu i wrócić do menu należy użyć <m>.
Użycie <s> wywoła procedurę ustawiania alarmu.
Program poprosi o wprowadzenie daty i godziny wywołania alarmu. Tak jak przy konfiguracji programu, również ważny jest poprawny format:
<dzień>/<miesiąc>/<rok(ostatnie 2 cyfry)>|<godzina>:<minuty>:<sekundy>
Program poinformuje użytkownika jeśli wszystko się powiedzie.

Po ustawieniu alarmu, w menu głównym pojawią się 2 dodatkowe opcje:

- <c> – wyświetla ustawioną datę i godzinę alarmu
- <r> – usuwa ustawiony alarm

Jeśli zegar RTC (Real Time Clock) zrówna się z wartością ustawioną jako budzik, to płytka zacznie migać czerwoną diodą oraz dołączony zewnętrznie buzzer się uruchomi.

Program poinformuje nas o dostępnych opcjach:

- <a> – wyłącza alarm
- <d> – wywołuję drzemkę

Czas drzemki ustawiony jest w programie na czas 15 sekund i 3 powtórzenia. Przy próbie kolejnej drzemki użytkownik informowany jest, że należy już wstawać i ma możliwość jedynie zatrzymania alarmu.
Po tym program wraca do domyślnego stanu, gdzie ponownie można odczytać godzinę oraz ustawić kolejny alarm.

3. Moduły projektu

- leds – moduł do obsługi diody RGB
    - leds.h – plik nagłówkowy zawierający deklaracje funkcji oraz makra RED/GREEN/BLUE odpowiadające za wygodny wybór kolorów diody RGB
    - leds.c plik zawierający implementacje funkcji:
        - initLed(void) – funkcja inicjalizująca porty GPIO
        - blinkLeds(void) – funkcja włączająca po kolei wszystkie ledy
        - ledOn(uint8_t color) – włącza wybraną diodę
        - ledBlink(uint8_t color, uint8_t count) – mruga wybraną diodą *count* razy
        - ledOff(void) – wyłącza diodę RGB
- uart – moduł do komunikacji UART0
    - uart.h – plik nagłówkowy z deklaracjami funkcji
    - uart.c – zawiera funkcje:
        - uart_init(void) – inicjalizuje moduł UART0
        - uart_sendCh(uint8_t data) – przesyła pojedynczy znak na terminal
        - uart_sendStr(char* str) – przyjmuje wskaźnik na tablicę znaków i przesyła ciąg znaków na terminal
        - uart_getChar(void) – odbiera pojedynczy znak przez uart
- buzzer – moduł do obsługi buzzera
    - buzzer.h – plik nagłówkowy z deklaracjami funkcji oraz makro definiujące pin do którego połączony jest buzzer
    - buzzer.c – funkcje:
        - buzzerInit(void) – funkcja inicjalizująca porty GPIO dla buzzera
        - buzzerTone(uint8_t count, uint32_t length) – włącza dźwięk buzzera trwający *length* * 1000 cykli zegara *count* razy
- extra – moduł na dodatkowe komponenty
    - extra.h – plik nagłówkowy
    - extra.c – zawiera funkcje:
        - delay_mc(uint32_t value) – powoduje opóźnienie trwające *value* * 1000 cykli zegara
- alarm – moduł do uruchamiania alarmu
    - alarm.h – plik nagłówkowy
    - alarm.c – funkcje:

        - ringTheAlarm(void) – rozpoczyna wykonywanie się alarmu – miga czerwona dioda oraz buzzer wydaje dźwięk
        - stopTheAlarm(void) – zatrzymuje alarm
- rtc – moduł Real Time Clock
    - rtc.h – plik nagłówkowy
    - rtc.c – zawiera funkcje:
        - rtc_init(void) – funkcja inicjalizująca moduł rtc do pracy
        - rtc_write(uint32_t t) – funkcja wpisująca do rejestru TSR liczbę sekund, którą moduł RTC inkrementuje z częstotliwością 1Hz
        - rtc_read(void) – zwraca aktualną wartość rejestru TSR
        - rtc_set_alarm(uint32_t time) – funkcja wpisująca do rejestru TAR liczbę sekund, przy której jeśli rejestr TSR się wyrówna i inkrementuje, to wystawia przerwanie
- datetime – moduł do wygodnej obsługi i konwertowania formatu daty zebranej od użytkownika
    - datetime.h – plik nagłówkowy zawierający strukturę danych i zdefiniowany typ danych *date_time_t* reprezentujący datę oraz godzinę oraz tzw. Look-up-table dla ilości dni w poszczególnych miesiącach co 4 lata (dla uwzględnienia lat przestępnych). Zawiera deklaracje funkcji.
    - datetime.c – zawiera implementacje funkcji:
        - date_time_to_epoch(date_time_t* date_time) – funkcja służąca do konwersji struktury *date_time_t* na liczbę sekund, która upłynęła od 1 stycznia 1970r od godziny 00:00:00 (patrz: Epoch Time, Unix Time)
        - epoch_to_date_time(date_time_t* date_time, uint32_t epoch) – konwertuje czas *epoch* na strukturę danych *date_time_t*
        - parse_str_to_date(char* str, date_time_t* date_buf) – funkcja przyjmująca wskaźnik na tablicę znaków oraz na strukturę *date_time_t*. Służy do zamiany wprowadzonego przez użytkownika formatu daty na strukturę typu *date_time_t*. Zwraca 1 w przypadku powodzenia, lub 0 w przypadku błędu.
        - date_time_uart_send_str(date_time_t* date_time) – funkcja do wysłania wypełnionej i zmienionej na string struktury typu *date_time_t* przez moduł UART
        - itoa(int value, char* result, int base) – statyczna funkcja biblioteczna do zamiany liczby typu *int* na *“string”*.

        - find_number(uint8_t ch) – statyczna funkcja pomocnicza służąca do konwersji liczby przyjętej przez uart w formacie znaku ASCII na zwykły integer.
- menu – moduł do wyświetlania menu
    - menu.h – plik nagłówkowy
    - menu.c – zawiera funkcję:
        - show_menu(uint32_t is_alarm_set) – wyświetla menu pomocnicze. Przyjmuje argument *is_alarm_set,* który jeśli jest równy zero oznacza, że alarm nie jest ustawiony, a jeśli wartość jest różna od zera, to oznacza że alarm jest ustawiony i wyświetla dodatkowe opcje
- main.c – plik z logiką programu. Zawiera makra dotyczące wielkości bufora na tekst przesyłany z UART, wielkości określające ustawienia drzemki, makro służące do czyszczenia flagi SR[TAF] oraz IDLE do powrotu do stanu "jałowego" po wybraniu odpowiedniej opcji z menu.
    - UART0_IRQHandler(void) – przerwanie z UART0 podczas odbierania wiadomości z UARTa
    - RTC_IRQHandler(void) – przerwanie pochodzące od RTC kiedy wartość TSR zrówna się z TAR i inkrementuje. Funkcja wywołuje alarm.
    - main(void) – główna logika programu, inicjalizacje modułów, konfiguracja programu i logika dotycząca odpowiednich opcji wybranych przez użytkownika

Literatura

- *KL05 Sub-Family Reference Manual*
- *Quick Start Guide for FRDM-KL05Z*
- *FRDM-KL05Z User's Manual*