Zastosowanie wyświetlacza LCD wraz z płytką ewaluacyjną Open 1768

Wykonał: Tomasz Kąkol

Spis treści

Lista zastosowanego sprzętu / funkcjonalności / aplikacji

- Płytka ewaluacyjna Open 1768 (ARM Cortex-M3)
- Wyświetlacz LCD
  - wyświetlacz 3,2" TFT (320x240)
  - sterownik *ILI9325*
  - podstawowe procedury zawarte w *lcd_lib.tgz* dołączone do projektu
    - lcd_lib/Open1768_LCD.h
    - lcd_lib/LCD_ILI9325.h
    - lcd_lib/asciiLib.h
- Terminal *Tera term*
- Środowisko *uVision 5*
- Złącza RS232 (interfejs UART)
- Komputer stacjonarny

Opis działania programu – instrukcja użytkownika

Po uruchomieniu układu użytkownik za pomocą klawiatury wypisuje dowolny tekst w terminalu *Tera term*. Następnie tekst ten zostaje wyświetlany na wyświetlaczu LCD w czasie rzeczywistym. Po całkowitym wypełnieniu obszaru wyświetlacza przez tekst następuje automatyczne wyczyszczenie ekranu, umożliwiając czytelne wyświetlanie kolejnych wypisywanych znaków przez użytkownika.

Opis działania programu – opis algorytmu

1. Deklaracja / Definicja zmiennych
2. Konfiguracja i Inicjalizacja wyświetlacza LCD
3. Nadpisanie całego wyświetlacza w jednolitej barwie z wykorzystaniem metody służącej do zapisu wartości do wybranego rejestru LCD

    void lcdWriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValue)

    gdzie:
    - *LCD_Reg: adres wybranego rejestru.*
    - *LCD_RegValue: wartość do zapisu do wybranego rejestru*
4. *Konfiguracja PIN-ów*
5. Wypisywanie pojedynczej liter na wyświetlaczu
    a. Nasłuchiwanie na wypisanie znaku w terminalu *Tera term*
    b. Zamiana wartości i uzyskanie kodu ASCUU zgodnie ideą funkcji:
    - void GetASCIICode(int font, unsigned char* pBuffer, unsigned char ASCII)

    gdzie:

    - znak ASCII odczytujemy z terminalu z użyciem stałej LPC_UART0 zadeklarowanej z pomocą dyrektywy #define w pliku nagłówkowym LPC17xx.h:
        - LPC_UART0->RBR
    - Po odczytaniu znaku, do zmiennych lokalnych Xsize Ysize wyświetlacza dodawane są wartości w celu przygotowania ‘przesunięcia’ do pozycji przygotowanej dla następnej literki
    - Za każdym razem po wyświetleniu znaku sprawdzane jest, czy nie wychodzimy już poza szerokość lub wysokość wyświetlacza. Jeżeli przekroczono szerokość, przechodzimy do nowej linii. Jeżeli przekroczono wysokość, nadpisujemy cały wyświetlacz nową jednolitą warstwą i zaczynamy wypisywanie literek od początku.
    - Wracamy do początku punktu 5 i algorytm jest w pętli nieskończonej.

Opis funkcjonalności

Rejestry UART

Powiązane z LPC1768 (w projekcie korzystałem wyłącznie z UART0):

- RBR - zawiera ostatnio otrzymane dane (Input),
- THR - Zawiera dane do przesłania (Output),
- FCR - rejestr kontrolny FIFO
- LCR - Kontroluje formatowanie ramek UART (liczba bitów danych, bity stopu) (IO)
- DLL - najmniej znaczący bajt wartości baud rate transmisji UART (IO)
- DLM - Najbardziej znaczący bajt wartości baud rate transmisji UART (IO)

Konfiguracja rejestru UART

LPC1768 ma wbudowane 16-bitowe FIFO dla odbiornika Rx / nadajnika Tx. W ten sposób może przechowywać 16-bajtowe dane otrzymane na UART bez nadpisywania. Jeśli dane nie zostaną

odczytane przed wypełnieniem kolejki (FIFO), nowe dane zostaną utracone, a bit błędu OVERRUN zostanie ustawiony.

FCR						
31:8	7:6	5:4	3	2	1	0
RESERVED	RX TRIGGER	RESERVED	DMA MODE	TX FIFO RESET	RX FIFO RESET	FIFO ENABLE

- Bit 0 - FIFO:
  - Ten bit jest używany do włączania / wyłączania FIFO dla danych odbieranych / przesyłanych.
    - 0 - FIFO jest wyłączony.
    - 1 - FIFO jest włączone zarówno dla Rx, jak i Tx.
- Bit 1 - RX_FIFO:
  - Służy do wyczyszczenia 16-bajtowego Rx FIFO.
    - 0 - Bez wpływu.
    - 1 - Wyjaśnia 16-bajtowy Rx FIFO i resetuje wskaźnik FIFO.
- Bit 2 - Tx_FIFO:
  - Służy to do wyczyszczenia 16-bajtowego Tx FIFO.
    - 0 - Bez wpływu.
    - 1 - Czyści 16-bajtowy Tx FIFO i resetuje wskaźnik FIFO.
- Bit 3 - DMA_MODE:
  - Służy do włączania / wyłączania trybu DMA.
    - 0 - Wyłącza DMA.
    - 1 - Włącza DMA tylko, gdy bit FIFO (bit-0) jest ustawiony.
- Bit 7: 6 - Rx_TRIGGER:
  - Ten bit służy do wyboru liczby bajtów danych odbiornika, które mają być zapisane, aby umożliwić przerwanie / DMA.
    - 00-- Trigger level 0 (1 znak lub 0x01)
    - 01-- Trigger level 1 (4 znaki lub 0x04)
    - 10-- Trigger level 2 (8 znaków lub 0x08)
    - 11-- Trigger level 3 (14 znaków lub 0x0E)

Bibliografia

1. https://www.waveshare.com/wiki/Open1768
2. http://www.fis.agh.edu.pl/koidc/staff.php?worker=6&type=2
3. https://exploreembedded.com/wiki/LPC1768:_UART_Programming