Odtwarzacz WAV

LPC 2148v3 Eduboard + LPC2148 „Expansion Board”

Skład grupy:

Łukasz Włodarczyk 216922 - lider

Mateusz Zawisza 216935

Jakub Grębowicz 216770

Tabela funkcjonalności:

Nazwa Funkcjonalności	Osoba odpowiedzialna za implementacje
Interfejs SPI	Jakub Grębowicz
Obsługa karty SD	Jakub Grębowicz
DAC	Mateusz Zawisza
Timer	Mateusz Zawisza
Obsługa przerwań	Mateusz Zawisza
Hitachi screen 2x16	Mateusz Zawisza
GPIO (Joystick i przycisk)	Jakub Grębowicz
UART	Łukasz Włodarczyk
Silniczek krokowy (wiatraczek)	Łukasz Włodarczyk
Ekran expansion board	Łukasz Włodarczyk

Udział procentowy poszczególnych członków:

Łukasz Włodarczyk – 30 %

Mateusz Zawisza – 40 %

Jakub Grębowicz – 30 %

Spis treści

Dokumentacja użytkownika

Wymagania sprzętowe

Zaprogramowany przez nas czytnik kart obsługuje karty SD sformatowane do FAT16 lub FAT32. Działa zarówno dla kart o niskiej pojemności, jak i tych większych (Testowaliśmy karty do pojemności 2GB).

Aby odtworzyć plik muzyczny, należy go przygotować według następujących wytycznych:

- Format pliku .wav
- Bit resolution 8 bit
- Sampling rate 8000Hz
- Mono (Jest to spowodowane budową mikrokontrolera LPC 2148. Obsługuje on tylko jedne kanał DAC.
- Aby zapewnić wyświetlanie nazwy pliku podczas odtwarzania jego nazwa powinna mieć 8 znaków.

Można to zrobić np. przy pomocy konwertera znajdującego się na tej stronie.

Interfejs użytkownika

Do obsługi urządzenia używamy przycisku oraz joystick'a.

Za pomocą **joystick'a** możemy przełączać się pomiędzy plikami znajdującymi się na karcie sd

- Przesunięcie w prawo – następny plik
- Przesunięcie w lewo – poprzedni plik
- Przesunięcie w górę – odtworzenie aktualnego pliku od początku
- Przesunięcie w dół – powrót do pierwszego pliku na karcie pamięci

Za pomocą przycisku 0.14 możemy uruchomić odtwarzanie oraz zastopować aktualnie odtwarzaną piosenkę. O stanie (start/stop) informuje nas znaczek na ekranie głównym.

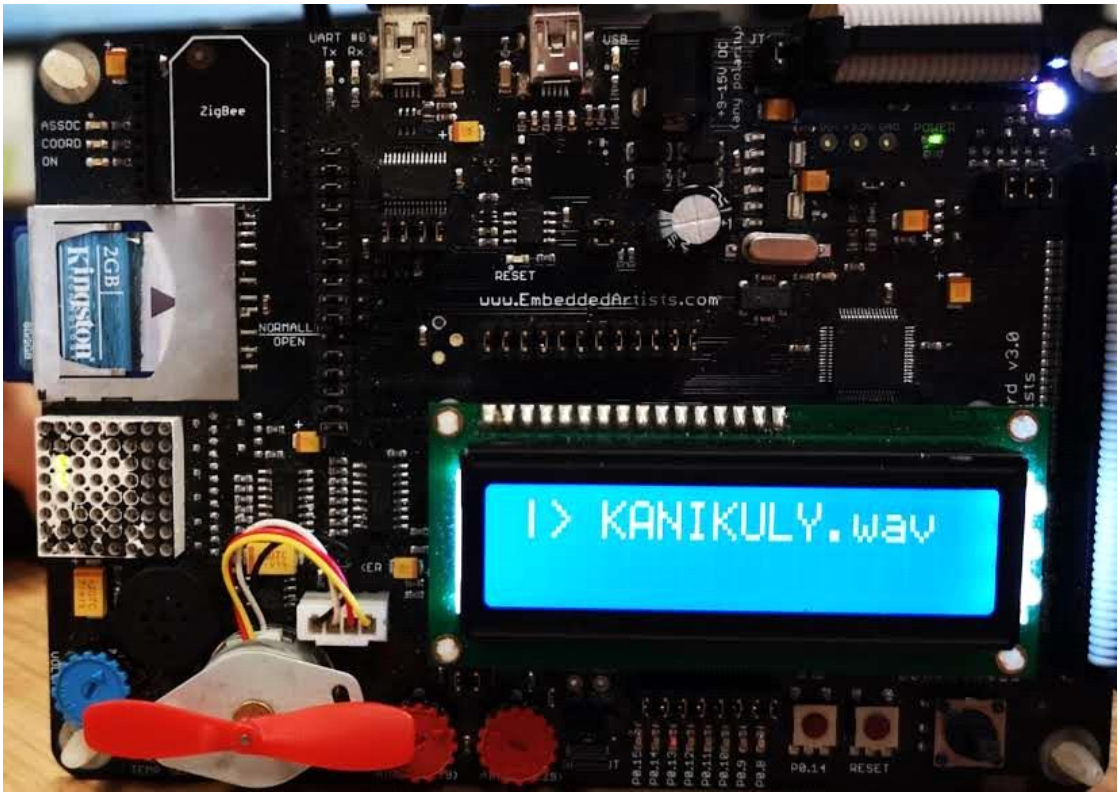
Opis działania programu

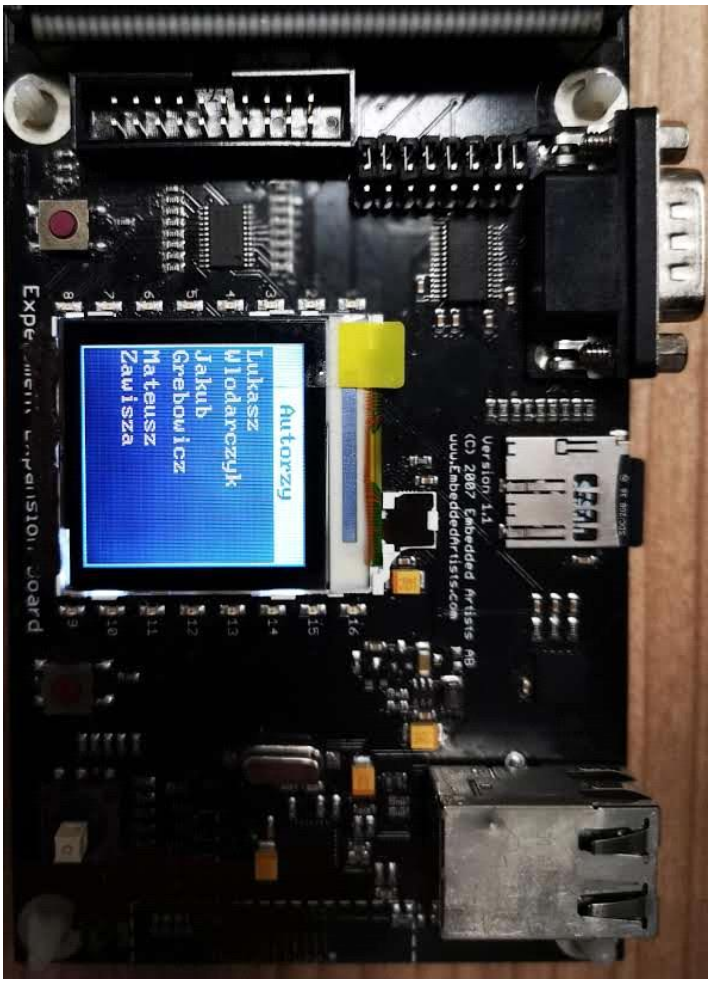
Po uruchomieniu urządzenia następuje inicjacja wszystkich niezbędnych komponentów. Później następuje sprawdzenie dostępności karty SD oraz odczytanie zawartych na niej danych.
W przypadku wykrycia plików wav następuje ich zliczenie oraz automatycznie wybranie pierwszego z nich jako aktywnego i oczekiwanie na uruchomienie odtwarzania poprzez wciśnięcie przycisku. W międzyczasie na ekranie głównym wyświetlane są aktualnie wykonywanie działania. Na ekranie dodatkowym możemy zobaczyć imiona i nazwiska autorów programu.

Kiedy wciśniemy przycisk P0.14 rozpoczyna się odczyt danych z danego pliku na karcie poprzez naprzemienne napełniane dwóch buforów (Podczas odczytywania jednego z nich, drugi jest napełniany. Ich pojemność to 10240 bajtów każdy), z których dane są przetwarzane za pomocą DAC w przerwaniach systemowych.

Dalsze działanie programu uzależnione jest od tego, jakie opcje interfejsu użytkownika wybierzemy (np. Przełączanie piosenek, startowanie, stopowanie itp.).

Po wybraniu dowolnej operacji, następną można wykonać dopiero po czasie 1 sekundy. Służy to uniknięciu błędnych odczytów spowodowanych drganiem zestyków.





Funkcjonalności

Interfejs SPI

Interfejs SPI to dwukierunkowy interfejs synchroniczny. Wykorzystuje 4 Piny:

- **MISO (Master In Slave Out)**
  Wykorzystywany do wysyłania danych
- **MOSI (Master Out Slave In)**
  Wykorzystywany do otrzymywania danych
- **SCK (Serial Clock)**
  Sygnał taktujący
- **SSEL (Slave Select)**
  Wybór układu podrzędnego

W naszym programie interfejs SPI jest wykorzystany do połączenia slotu SD z mikrokontrolerem

Jest on ustawiony tak, aby generować zegar taktującego transfer danych z częstotliwością ok. **15 MHz**.

```
#define SPIF    7

#define SPI_SCK_PIN    4   /* Clock*/
#define SPI_MISO_PIN   5   /* from Card*/
#define SPI_MOSI_PIN   6   /* to Card*/
#define SPI_SS_PIN     11   /* Card-Select*/

#define SPI_PINSEL     PINSEL0
#define SPI_SCK_FUNCBIT   8
#define SPI_MISO_FUNCBIT  10
#define SPI_MOSI_FUNCBIT  12
#define SPI_SS_FUNCBIT    14

#define SPI_PRESCALE_REG  S0SPCCR
#define SPI_PRESCALE_MIN  8

#define  SPI_CS   0x00008000
#define SELECT_CARD()   IOCLR0 = (1<<SPI_SS_PIN) // IOCLR ->low =
wlaczenie
#define UNSELECT_CARD() IOSET0 = (1<<SPI_SS_PIN) // IOSET ->high =
wylaczenie


#define CPHA 3
#define CPOL 4
#define MSTR 5
#define LSBF 6
#define SPIE 7

SPI.c
void initSpi(void)
{
    // setup GPIO
    IODIR0 |= (1<<SPI_SCK_PIN)|(1<<SPI_MOSI_PIN)|(1<<SPI_SS_PIN);
// |= - output, 1<<pin (np P0.11)
    IODIR0 &= ~(1<<SPI_MISO_PIN); // &= - input
                                  // &= (1<<10) | (1<<11)
    // set Chip-Select high - unselect card
    IOSET0 = (1<<SPI_SS_PIN); // IOSET0 = (1<<SPI_SS_PIN)

    // reset Pin-Functions
    SPI_PINSEL &= ~( (3<<SPI_SCK_FUNCBIT) | (3<<SPI_MISO_FUNCBIT)
|
```

```
        (3<<SPI_MOSI_FUNCBIT) | (3<<SPI_SS_FUNCBIT) ); // Pin
function Select register 0 (PINSEL1 - 0xE002 C000)

    SPI_PINSEL |= ( (1<<SPI_SCK_FUNCBIT) | (1<<SPI_MISO_FUNCBIT) |
        (1<<SPI_MOSI_FUNCBIT) );

    // enable SPI-Master
    S0SPCR = (1<<MSTR)|(0<<CPOL);
        // low speed during init 400kHz
    setSpiSpeed(150);
```

Karta SD (Secure digital)

Karty SD mają 9 wyprowadzeń. Dodatkowo karta SD ma na brzegu przełącznik blokujący możliwość zapisu.

W naszym programie kartę SD obsługujemy z pomocą interfejsu SPI.

```
#define SPI_SS_PIN  11
#define SD_Disable() IO0SET = 1 <<11
#define SD_Enable() IO0CLR = 1 <<11
#include "integer.h"
#define CMDREAD     17
#define CMDWRITE    24
#define CMDREADCSD  9


CHAR sdInit(void)
{

    SHORT i;
    BYTE resp;

    spiSend(0xff);
    i=100;
    do
    {
        sdCommand(0, 0, 0); // wyslanie komendy 0
        resp=sdResp8b(); // odczyt wartosci rejestru S0SPDR
    }
    while(resp!=1 &&i--);

    if( resp != 1) // jesli odpowiedz jest rozna od 1
```

```
    {
        if(resp == 0xff)
        {
            return(-1);
        }
        else // sprawdzenie kodu bledu
        {
            sdResp8bError(resp);
            return(-2);
        }
    }

    i=32000;
    do
    {
        sdCommand(1, 0, 0);

        resp=sdResp8b();
        if(resp!=0)
        {
            sdResp8bError(resp);
        }
    }
    while(resp == 1 &&i--);

    if( resp != 0)
    {
        sdResp8bError(resp);
        return(-3);
    }
    return(0);

}
```

UART (Universal Asynchronous Receiver/Transmitter)

UART jest to protokół asynchronicznej komunikacji szeregowej w którym dane są przesyłane bit po bicie.

LPC2148 posiada dwa bloki UART.
W naszym programie używamy UART0, i Bitrate 38400 baud. Do ustawienia powyższych wartości służy plik config.h i linie:

```
#define CONSOL_UART         0
```

```
#define CONSOL_BITRATE      38400UL
```

Następnie ustawiamy piny w GPIO dla UART0 i pozostałe ustawienia potrzebne do inicjalizacji w pliku consol.c:

```
PINSEL0 = (PINSEL0 & 0xfffffff0) | 0x00000005;
//initialize bitrate (by first enable DL registers, DLAB-bit = 1)
 UART_LCR = 0x80;
 UART_DLL = (unsigned char)(UART_DLL_VALUE & 0x00ff);
 UART_DLM = (unsigned char)(UART_DLL_VALUE>>8);
 UART_LCR = 0x00;
 //initialize LCR: 8N1
 UART_LCR = 0x03;
 //reset FIFO
 UART_FCR = 0x00;
 //clear interrupt bits
 UART_IER = 0x00;
```

UART w LPC2148

PIN	TxD(Transmit Data)	RxD(Recieve Data)
UART0	P0.0	P0.1
UART1	P0.8	P0.9

Aby skomunikować się z komputerem i wyświetlić coś na terminalu, używamy UART.

Do terminala dane wysyłamy za pomocą simplePrintf().

Przerwania

Przerwanie powoduje wstrzymanie aktualnie wykonywanego programu i wykonanie kodu procedury obsługi przerwania przez procesor. W naszym programie przerwanie VIRQ jest generowane za pomocą timera co każde 125 mikro sekund.
Na początku przydzielamy przerwanie do VIRQ, a następnie wskazujemy adres funkcji, która to przerwanie obsłuży. Uaktywniamy slot i włączamy przerwanie.
Program może obsługiwać łącznie 18 różnych przerwań (gdzie 16 to VIRQ, 1 FIQ, 1 NVIRQ).

Timer

LPC2148v3 zawiera dwa 32 bitowe timery. Każdy z nich może być używany jako Timer lub Counter. Timer używany jest przez nas do generowania przerwań IRQ. Wykorzystujemy je do odtwarzania dźwięku (korzystając z DAC).
Używając Timera 1 odtwarzamy jedną próbkę co każde 125 mikro sekund. W ciągu sekundy odtworzone zostanie więc 8 000 próbek.
Początkowo uruchamiamy i resetujemy licznik. Następnie zamieniamy częstotliwość na takty i resetujemy flagi przerwań. Generujemy okresowe przerwanie i uruchamiamy timer.

DAC

LPC 2148 posiada 10-bitowy cyfrowo-analogowy przetwornik DAC. Posiada on buforowane wyjście oraz możliwość wyboru między szybkością a poborem mocy.

DAC na LPC 2148 posiada jeden rejestr DACR (Digital to Analog Register).

Bit	Symbol	Wartość	Opis	Wartość po resecie
0-5	-		Zarezerwowane	N/A
6-15	VALUE		Po wybraniu „setting time” VALUE/1024 * Vref jest zapisywana do pinu Aout	
16	BIAS	1 0	„Setting time” ustawiony na 1 µs, maksymalne natężenie to 700 µA „Setting time” ustawiony na 2,5 µs, maksymalne natężenie to 350 µA	
17-31	-		Zarezerwowane	

Rejestr DACR jest zapełniany zdekompresowanymi próbkami w przerwaniu, które znajduje się w pliku.
Na początku programu następuje zainicjowanie, włączenie DAC poprzez zresetowanie, a następnie odpowiednie ustawienie 18 i 19 bitu (odpowiednio 0 i 1) dla P0.25:

```
PINSEL1 &= ~0x000C0000;
PINSEL1 |= 0x00080000;
```

Odtwarzanie muzyki realizowane jest za pomocą przerwania.
Odpowiednia próbka odczytywana jest z bufora, a następnie ustawiona w rejestrze DACR wraz z biasem na odpowiednich miejscach (patrz tabela wyżej, 6-15 bity dla próbki, 16 bit dla biasu).

```
DACR = ((val+128) << 8) | (1 << 16);
```

Joystick

Używamy joysticka do przełączania pomiędzy aktualnie odtwarzanymi utworami. W przypadku skręcenia w prawo – przeskakujemy do następnego pliku. W przypadku skręcenia w lewo – do poprzedniego.

Przycisk (P0.14)

Przycisk ten wykorzystywany jest do zatrzymywania i odtwarzania danego pliku. Po jego wciśnięciu zatrzymujemy bądź wznawiamy obsługę przerwań w programie.

Hitachi wyświetlacz 2x16

Wyświetlacza używamy do wyświetlania komunikatów dla użytkownika oraz nazwy aktualnie odtwarzanego pliku. Wyświetlacz wykorzystuje następujące PINy:

```
#define LCD_DATA        0x00ff0000    //P1.16-P1.23     - obsługa bitów danych
#define LCD_E           0x02000000    //P1.25           - bit kontrolny
#define LCD_RW          0x00400000    //P0.22           - bit zapis/odczyt
#define LCD_RS      0x01000000    //P1.24               - bit adres RS
#define LCD_BACKLIGHT   0x40000000    //P0.30           - bit podświetlenia
```

Po zainicjowaniu wyświetlacza wyłączamy go, a następnie czyścimy. Włączamy tryb pisania, włączamy wyświetlacz i ustawiamy kursor na początku wyświetlacza (cursor home). Wyświetlanie tekstu odbywa się poprzez odpowiednie ustawienie bitów LCD_DATA i LCD_RW w IOCLR0 i IOCLR1. Dane do wyświetlenia przesyłane są na IOSET1. Na koniec ustawiamy bit kontrolny, odczekujemy 250 nano sekund, zwalniamy go i ponownie odczekujemy 250 nano sekund.

Silnik krokowy (wiatraczek)

Płytka posiada mały silnik krokowy 5V dwubiegunowy z zamontowanym śmigłem. Piny P0.12 i P0.21 są wykorzystywane do kontrolowania ruchu silnika krokowego.

W naszym programie wykorzystujemy wzór forward, ustawiamy 2 kombinacje wyżej wymienionych pinów P0.12 i P0.21. Przy przełączaniu piosenek wiatraczek „przeskakuje” z jednej pozycji na drugą, sygnalizując użycie joysticka.

Ekran dodatkowy

Ekran dodatkowy (LCD) komunikuje się poprzez synchroniczny interfejs SPI.
Ustawienie parametrów pracy odbywa się w metodzie initSpiForLcd(void) z pliku lcd_hw.c, gdzie ustalane są następujące wartości:

```
#define LCD_CS     0x00000080

#define LCD_CLK    0x00000010

#define LCD_MOSI   0x00000040

IODIR |= (LCD_CS | LCD_CLK | LCD_MOSI);

PINSEL0 |= 0x00001500;

SPI_SPCCR = 0x08;

SPI_SPCR  = 0x20;
```

.

Używane przez nas funkcje:

- **lcdColor**
  Ustawiamy kolor zawartości i tła.
- **lcdClrscr**
  Funkcja służy do czyszczenia zawartości ekranu. Ustawia punkt początkowy na (0,0) i zapisuje ekran kolorem tła.
- **lcdGotoxy**
  Funkcja służy do określenia współrzędnych punktu początkowego dla wprowadzania informacji na ekran.
- **lcdPuts**
  Funkcja służy do wyświetlenia na ekranie łańcucha znakowego (char tab [n]).Wypisuje ona łańcuch znakowy za pomocą funkcji lcdPutchar(tU8 data).
- **lcdRect**
  Funkcja służy do wyświetlenia na ekranie prostokąta, jako argumenty przyjmuje wartości współrzędnych lewego górnego wierzchołka, wymiary i wartość oznaczającą wybrany kolor, opisaną na ośmiu bitach (po trzy na R i G, dwa na B). Jej działanie ogranicza się do zabarwienia odpowiednich pikseli (w obrębie prostokąta) wskazanym kolorem.
- **lcdInit**
  Funkcja służy do inicjacji ekranu, ustawia np. paletę kolorów, tryb wyświetlania (np. można odwrócić do "góry nogami")

W naszym wypadku wybrany jest tryb kolorów 0x02, który pozwala nam korzystać z 256 kolorów, przy kontraście ustawionym na poziomie 56 (0-127). Dodatkowo używamy następujących makr:

```
#define LCD_CMD_SWRESET   0x01
#define LCD_CMD_BSTRON    0x03
#define LCD_CMD_SLEEPIN   0x10
#define LCD_CMD_SLEEPOUT  0x11
#define LCD_CMD_INVON     0x21
#define LCD_CMD_SETCON    0x25
#define LCD_CMD_DISPON    0x29
#define LCD_CMD_CASET     0x2A
#define LCD_CMD_PASET     0x2B
#define LCD_CMD_RAMWR     0x2C
#define LCD_CMD_RGBSET    0x2D
#define LCD_CMD_MADCTL    0x36
#define LCD_CMD_COLMOD    0x3A
#define MADCTL_HORIZ      0x48
#define MADCTL_VERT       0x68
```

Analiza skutków awarii

Skutki awarii poszczególnych elementów systemu znajdują się w tabeli poniżej.

Element systemu	Skutek awarii	Opis
Mikrokontroler	Krytyczny	W razie awarii, uszkodzenia lub braku zasilania całego mikrokontrolera nie ma możliwości skorzystania z odtwarzacza.
Interfejs SPI	Krytyczny	Awaria interfejsu SPI oznacza brak możliwości odczytu z karty, co oznacza brak możliwości odtworzenia plików.
Slot karty SD	Krytyczny	Awaria slotu karty również uniemożliwia nam odczyt plików z karty, co sprawia że urządzenie nie nadaje się do użycia.
Konwerter DAC	Krytyczny	Awaria konwertera DAC blokuje możliwość uzyskania dźwięku. Brak możliwości odsłuchu wyłącza nasze urządzenie z użycia.
Przycisk	Krytyczny	Brak możliwości wciśnięcia przycisku zablokuje nie tylko możliwość zatrzymywania aktualnie odtwarzanej piosenki, ale również możliwość „wystartowania” całego

		urządzenia, ponieważ aby rozpocząć słuchanie muzyki należy wcisnąć przycisk P0.14
Joystick	Średni	Brak możliwości skorzystania z joysticka zablokuje możliwość przełączania się pomiędzy odtwarzanymi aktualnie utworami, co może sprawiać duże kłopoty, tym bardziej w przypadku gdy na karcie posiadamy wiele plików.
Wiatraczek	Niski	Jest to dodatek do programu. Nie jest istotny dla działania całości.
Wyświetlacz dodatkowy	Niski	Bez dodatkowe wyświetlacza nie zobaczymy nazwisk autorów programu. Nie ma to wpływu na działanie całości.
Wyświetlacz LCD	Niski	Awaria wyświetlacza nie stanowi dużego zagrożenia. Tracimy informację o tytule odtwarzanego aktualnie pliku oraz część interfejsu użytkownika. Nie jest jednak to on na tyle skomplikowany, aby nie poradzić sobie bez komunikatów od urządzenia.