SPRAWOZDANIE – REFLEKS LPC1768

SPIS TREŚCI:

1. Nazwiska Autorów

Zespół A07:
Łukasz Gołębiewski 203882 – lider
Barbara Josko 203898
Jakub Mielczarek 203943

2. Lista wykorzystanych funkcjonalności

Płytka LPC1768 z mikrokontrolerem:
- I2C
- SPI
- DAC
- TIMER
- TIMER IRQ

Płytka LPCXpresso Base Board:
- wyświetlacz OLED
- EEPROM
- akcelerometr
- joystick (GPIO)
- głośnik

3. Zakres obowiązków członków zespołu

Zakres obowiązków został podzielony wg opisanych w dokumentacji funkcjonalności:
Łukasz Gołębiewski: I2C, TIMER, TIMER IRQ, GŁOŚNIK, DAC
Barbara Josko: SPI, OLED, FMEA
Jakub Mielczarek: EEPROM, akcelerometr, joystick

Udział procentowy w wykonaniu projektu:
Łukasz Gołębiewski: 44%
Barbara Josko: 28%
Jakub Mielczarek: 28%

4. Opis działania programu – instrukcja użytkownika

Gra po uruchomieniu zagra melodię a na ekranie zacznie pojawiać się przesuwający ciąg losowych liter R, L, U, D. Zadaniem użytkownika jest odpowiednie sterowanie joystickiem w momencie gdy jedna z liter znajdzie się tuż nad wskaźnikiem znajdującym się na środku ekranu. R oznacza, że należy wcisnąć joystick w prawo, L w lewo, U w górę, D w dół. Za poprawną reakcję naliczany jest punkt. Za błędną gra resetuje się. Za brak reakcji użytkownika odejmowany jest punkt. W lewym górnym rogu ekranu widoczny jest czas w sekundach od włączenia gry. Na środku na górze ekranu widoczny jest najlepszy wynik. W prawym górnym rogu widoczny jest aktualny wynik gracza. Potrząsając płytką można zresetować grę – zastosowano funkcjonalność akcelerometru.

5. Opis działania programu – opis algorytmu

Program został napisany w języku C. Najpierw następuje inicjalizacja wszystkich peryferiów i zmiennych. W głównej pętli aktualizowany jest czas, odczytywany jest stan joysticku. Sprawdzana jest pozycja każdej litery ciągu poruszających się znaków. Jeżeli znajduje się nad wskaźnikiem i jest zgodna z kierunkiem joysticku naliczany jest punkt. Aby uniknąć naliczania wielu punktów za jeden znak, każdy znak w swojej strukturze przechowuje flagę która określa czy został za niego naliczony już punkt.
W każdym przebiegu głównej pętli sprawdzany jest również stan współrzędnych akcelerometru. Dopiero po tych czynnościach aktualizowana jest pozycja znaków wykorzystując czas (z timera 0) od ostatniej aktualizacji tak aby zapewnić płynny ruch. Znaki po przekroczeniu lewej krawędzi ekranu są przesuwane za ostatni znak, resetowane są flagi struktury i losowany jest znak L, R, D, U. Do przesuwania znaku za znak ostatni wykorzystano funkcję floor modulo która znajduje indeks znaku najdalej od lewej krawędzi ekranu (czyli znaku poprzedzającego aktualny znak).
Wyświetlanie znaków jest aktualizowane co 50ms a informacji na górze ekranu co 500ms.
Odświeżanie ekranu jest realizowane czyszcząc tylko wybrane strony pamięci ekranu wymagające aktualizacji obrazu.

W momencie końca gry, sprawdzane jest czy wynik jest lepszy od najlepszego wyniku odczytanego z pamięci EEPROM. Jeżeli tak, najlepszy wynik jest nadpisywany.

W grze istnieje możliwość odtwarzanie dwóch dźwięków jednocześnie w tle za pomocą przerwań timerów. W momencie włączenia płytki odgrywane są dwie melodie na raz, jedna bazuje na GPIO i IRQ timera 1, druga na DAC i IRQ timera 3. Podczas naliczania punktów czy końca gry odgrywane są dźwięki bazujące na operacjach na GPIO. Gdy nastąpi poprawienie najlepszego wyniku, odgrywana jest melodia bazująca na DAC z tablicy opartej na zawartości pliku WAV, 8bit, 8kHz.

W przerwaniu timera 1 odczytywana jest długość okresu nuty i ustawiane jest wywołanie następnego przerwania po minięciu połowy tego okresu. Odczytywana jest również długość odgrywania nuty. Następnie ustawiany jest sygnał na pinie połączonym z buczkiem: wysoki jeśli poprzedni był niski, niski jeśli poprzedni był wysoki. Na wykonanie tej operacji trzeba chwilowo przełączyć pin P0.26 na funkcję GPIO, po skończeniu z powrotem na DAC. Pod koniec sprawdzane jest czy upłynął czas odgrywania zadanej nuty (jest to możliwe dzięki timerowi 0), jeśli tak to w następnym wywołaniu przerwania pobrana zostanie następna nuta z tablicy. Jeśli nie będzie następnej nuty, nastąpi ustawienie wywołania następnego przerwania na dłuższy okres, reset rejestru TC i wyjście z przerwania (jest to konieczne, ponieważ odgrywanie dźwięku jest wywoływane, gdy stan joysticka jest niezerowy, przez co długie trzymanie joysticka nie powoduje błędów w odgrywaniu dźwięku, można by nie resetować rejestru TC w przerwaniu, a resetować go w momencie chęci odtworzenia dźwięku ale to wywoływałoby ciągłe resetowanie przy dłużej wciśniętym joysticku).

Przerwanie timera 3 wywoływanie jest tylko podczas odtwarzania dźwięku. Wywoływanie jest w częstotliwości 8kHz. W przerwaniu aktualizowany jest następny bajt z tablicy bajtów opisujących PCM dźwięku do rejestru DACR. Gdy tablica kończy się, następuje wyjście z przerwania. Podczas wywołania funkcji odtworzenia melodii resetowany jest rejestr TC timera 3.

6. Opis działania wykonanego sprzętu

Nie było wykonywanego sprzętu.

7. Opisy funkcjonalności

7. 1 I2C

I2C (Inter-Integrated Circuit) to dwukierunkowa szeregowa magistrala służąca do przesyłania danych pomiędzy układami elektronicznymi wyposażonymi w interfejs I2C. Do przesyłania danych wykorzystywane są tylko dwie linie:
- SCL (Serial Clock line). Linia na której pojawiają się impulsy zegara umożliwiające synchroniczny przesył danych.
- SDA (Serial Data line). Linia po której przesyłane są dane.

Każde urządzenie wyposażone w interfejs I2C posiada swój unikalny adres. Można go odczytać w nocie katalogowej producenta. Układy podłączone do I2C mogą pracować jako urządzenie jedynie odczytujące dane lub odczytujące oraz wysyłające. Dodatkową mogą pracować zarówno w trybie master jak i w trybie slave. Jest to zależne od tego czy urządzenie wykonuje inicjację przesłania danych czy jest tylko adresowane.
W przypadku naszej płytki mikrokontroler działa w trybie master z możliwością wysyłania i odczytu. Wszystkie inne urządzenia podłączone do magistrali działają w trybie slave. Protokoły przesyłania danych w tej konfiguracji wyglądają następująco:
- Przesłanie danych od master do slave. Master wysyła START bit, bajt określający adres urządzenia docelowego oraz bit R/W (kierunek przesyłu danych). Następnie przesyłane są bajty z danymi. Slave wysyła bit potwierdzający po każdym otrzymanym bajcie.
- Odczyt danych od slave do master. Master wysyła START bit, bajt określający adres urządzenia docelowego oraz bit R/W (kierunek przesyłu danych). Następnie slave wysyła bajty z danymi. Master wysyła bit potwierdzający po każdym otrzymanym bajcie. Po ostatnim wysłanym bajcie master generuje STOP lub kolejny START bit.

LPC1768 wyposażono w 3 interfejsy I2C. W naszym programie używamy tylko jednego – I2C2. Inicjalizacja I2C2 polega na włączeniu zasilania do interfejsu za pomocą 32 bitowego rejestru PCONP poprzez ustawienie 1 na 26 bicie tego rejestru (każdy bit rejestru odpowiada jednemu układowi)
Trzeba wybrać również odpowiedni dzielnik zegara PCLK poprzez 32 bitowe rejestry PCLKSEL. Dzielnik można ustawić na 1, 2, 4 lub 8. Każde urządzenie peryferyjne LPC1768 ma przeznaczone dwa bity w rejestrach PCLKSEL. Dodatkowo do rejestrów interfejsu I2C I2SCLH oraz I2SCLL należy wpisać żądaną ilość cykli zegara PCLKI2C dla którego będzie utrzymywał się stan wysoki i niski sygnału prostokątnego zegara podawanego na SCL. I2SCLH określa stan wysoki, I2SCLL stan niski. W przypadku naszego programu obie te wartości są sobie równe a finalna częstotliwość zegara na SCL wynosi 100 kHz.

$$I^2C_{bitfrequency} = \frac{PCLKI2C}{I2CSCLH + I2CSCLL}$$

Następnie do rejestru kontrolnego I2CONSET na 6 bicie (symbol I2EN) należy ustawić 1 aby włączyć interfejs. Wyłączanie analogicznie rejestrem I2CONCLR.

Należy jeszcze zdefiniować które piny na płytce będą pełniły funkcje linii SDA i SCL interfejsu I2C2. Umożliwiają to rejestry PINSEL której kontrolują multipleksery aby umożliwić połączenie pomiędzy pinami a urządzeniami peryferyjnymi mikrokontrolera. W przypadku naszego programu linie I2C2 są ustawione na piny P0.10 i P0.11. W tym celu ustawiamy rejestr 32 bitowy PINSEL0 (odpowiadający P0[15:0]) wartość bitową '10' (druga funkcja alternatywna czyli wyjścia I2C2) dla P0.10 (PINSEL0[21:20] i P0.11 PINSEL0[23:22]). Dla każdego pinu przeznaczone są dwa bity w rejestrze.
Dodatkowo należy ustawić rejestry PINMODE (no pull-up, no pull-down) oraz PINMODE_OD (open-drain) dla używanych pinów. W przypadku naszego programu pozostają wartości domyślne tych rejestrów dla pinów P0.10 i P0.11.

7.2 Głośnik GPIO, DAC

Głośnik podłączony jest do pinów GPIO. Należy tutaj ustawić jedynie kierunki pinów GPIO. Głośnik nie będzie wysyłał żadnych danych więc piny należy ustawić na wyjście. Dokonujemy tego poprzez ustawienie 1 w rejestrach FIODIR0 i FIODIR2 w miejscu odpowiadającym pinom głośnika (każdy bit rejestru odpowiada jednemu pinowi GPIO, każdy port GPIO ma swój rejestr FIODIR). Czyli 1 należy ustawić na 28, 27, 26 bicie

FIODIR0 i 13 bicie FIODIR2 Warto zauważyć, że wszystkie operacje na rejestrach GPIO są bardzo szybkie ponieważ są podłączone do magistrali AHB.

Głośnik LM4811 znajdujący się na płytce LPCXpresso Base Board wymaga kilku pinów z mikrokontrolera. Są to CLK, UP/DN, SHUTDN, VIN1/VIN2. Z noty katalogowej producenta można odczytać że piny CLK, UP/DN odpowiadają za synchroniczne sterowanie głośnością buczka, SHUTDN to pin aktywujący specjalną funkcję shutdown głośnika. Jednakże operacje na tych pinach nie pojawiają się w naszym programie poza inicjalizacją głośnika. Natomiast piny VIN1/VIN2 odpowiadają za generację sygnału wprawiającego membranę buczka w drgania czyli za generowanie dźwięków. Połączenia pinów głośnika do pinów GPIO wyglądają następująco:
P0.26 – VIN1/VIN2, P0.27 – CLK P0.28 – UP/DN, P2.13 – SHUTDN
Podczas inicjalizacji dodatkowo czyszczona jest wartość na pinach P0.27, P0.28, P2.13. Dokonujemy tego poprzez ustawienie 1 w rejestrach FIOCLR dla portu 0 i 2 w miejscu odpowiadającym w/w pinom (analogicznie do FIODIR).

Generowanie dźwięku buczka odbywa się poprzez podawaniu zmiennego napięcia na pin P0.26 tak aby wprowadzić membranę buczka w drgania. Melodię można tworzyć np. poprzez zwykłe operacje na GPIO lub za pomocą DAC.

Pierwszy sposób pozwala na generowanie prostych nut. Chcąc zagrać nutę *a* o częstotliwości 880 Hz, musimy wprowadzić membranę buczka w drgania a takiej częstotliwości. Okres drań *T* będzie wynosił 1136 µs. Stąd należy na pin P0.26 należy podawać stan wysoki przez okres równy T/2 oraz stan niski analogicznie przez T/2. Cykl ten należy powtarzać w zależności jak długo chcemy odtwarzać dźwięk.

Do ustawiania stanów wysokich i niskich na pinach GPIO używa się rejestru FIOSET i FIOCLR. Za generowanie dźwięku odpowiada pin P0.26. Chcąc ustawić stan wysoki na P0.26 należy wpisać 1 na 26 bicie rejestru FIO0SET. Ustawianie zera w tym rejestrze nie ma efektu. Chcą odwołać stan wysoki należy wpisać 1 na 26 bicie rejestru FIO0CLR.

Drugi sposób pozwala na tworzenie bardziej złożonych dźwięków. Sygnał na P0.26 będzie sygnałem analogowym, a nie jak w poprzednim przypadku prostokątnym sygnałem cyfrowym. Wykorzystywany jest w tym celu DAC (Digital Analog Converter) znajdujący się wewnątrz mikrokontrolera.
Inicjalizacja DAC wymaga skonfigurowania jedynie rejestrów PINSEL oraz ustawieniu maksymalnego natężenia dla DAC. Napięcie do DAC podawane jest zawsze, PCLK używa wartości domyślnych. Wyjście z DAC (AOUT) jest ustawione na buczek czyli na P0.26. Należy tutaj użyć alternatywnej funkcji pinu. W tym celu ustawiamy rejestr 32 bitowy PINSEL1 (odpowiadający P0[31:16]) wartość bitową '10' na 21 i 20 bicie tego rejestru (druga funkcja alternatywna dla P0.26 czyli AOUT). Dla każdego pinu przeznaczone są dwa bity w rejestrze.
Do rejestru PINMODE1 należy ustawić '00' na 21 i 20 bicie tego rejestru (odpowiadającym P0.26) czyli pullup. Do rejestry PINMODE_OD0 ustawiamy 0 na 26 bicie tego rejestru (odpowiadającemu P0.26) czyli tryb normalny (not open drain mode).

Do konwersji sygnału cyfrowego na analogowy wykorzystywany jest rejestr DACR. Na bitach 15-6 (VALUE) wpisywana jest binarna wartość która zostanie wysłana jako sygnał analogowy na AOUT. Na bicie 16 można ustawić maksymalny setting time – czas po którym VALUE zostanie przekazana na AOUT oraz maksymalne natężenie na AOUT. W naszym programie maksymalny setting time to 1µs a natężenie na maksymalne natężenie wynosi 700µA. Jako źródło dźwięku podajemy plik WAV 8 bitowy, 8kHz. Jest prostym formatem do takiego odtwarzania ponieważ dźwięk jest zapisany jako PCM. Wystarczy jedynie pominąć nagłówek pliku i odczytywać kolejne bajty. (przyjmując że format pliku WAV jest poprawny). Odczytane bajty należy wysyłać do rejestru DACR w częstotliwości 8kHz.

7.3 Timery i IRQ

LPC1768 wyposażono w 4 timery: timer0, timer1, timer2, timer3. W naszym programie wykorzystujemy 3 z nich. Timer0 wykorzystywany jest do czasu gry, timer1 do przerwań pozwalających odtwarzać muzykę w tle za pomocą głośnika i operacji na GPIO, timer3 do przerwań pozwalających odtwarzać muzykę w tle za pomocą głośnika i DAC. Najważniejsze rejestry potrzebne do konfiguracji timera0:
- Zasilanie jest dostarczane domyślnie
- PCLK_TIMER0 = 0. Dzielnik PCLK ustawiony na 4. PCLK_TIMER0 znajduje się w rejestrze PCLKSEL0[3:2]

- rejestr IR. Odczyt tego rejestru sprawdza czy oczekiwane są przerwania timera. Może być wyzerowany w celu wyczyszczenia oczekujących przerwań timera.
- rejestr TCR. Wyzerowany, powoduje reset TC.
- rejestr TC. Licznik inkrementowany co PR+1 cykli PCLK
- rejestr PR. Kiedy rejestr PC jest równy wartości w PR to następuje inkrementacja TC. PR jest ustawiony na ilość cykli PCLK odpowiadającej długości 1ms.
- rejestr PC. Inkrementowany co każdy cykl PCLK. Gdy równy PR jest resetowany do zera.
- rejestr MR0 – kiedy wartość TC będzie równa MR0 wywołane zostanie przerwanie IRQ. Timer 0 nie ma włączonych przerwań więc ten rejestr nie jest używany.
- rejestr MCR. Zawiera konfigurację określającą czy generowane jest przerwanie, oraz czy resetować wtedy wartości rejestrów tj. TC itp.

Timer1 jest inicjalizowany analogicznie do timera0. Różnice to:
- rejestr IR. Jest wyzerowywany w obsłudze przerwania do wyczyszczenia oczekujących przerwań.
- rejestr TCR. Jest używany w obsłudze przerwania do resetowania TC dopiero po wykonaniu poszczególnych operacji w przerwaniu.
- rejestr PR. Ustawiona ilość cykli odpowiadająca 1µs.
- rejestr MCR. Bit 0 tego rejestru ustawiony na 1 czyli włączenie przerwań.
- rejestr MR0. Wartość w µs określająca okres aktualnie granej nuty.

Timer3 jest inicjalizowany analogicznie do timera1. Różnice to:
- należy włączyć zasilanie do timera3 za pomocą rejestru PCONP poprzez ustawienie 1 na 23 bicie tego rejestru (każdy bit rejestru odpowiada jednemu układowi).
- rejestr MR0. Wartość w µs odpowiadająca okresowi dla jednego bajtu odczytanego z pliku WAV 8kHz. Czyli jest to 1000000µs / 8000.

Zarządzaniem przerwaniami zajmuje się Nested Vectored Input Controller (NVIC), będący częścią procesora Cortex-M3. Tablica wektorów przerwań posiada 35 identyfikatorów. ID przerwania timera1 to 2, timera3 to 4. Każde urządzenie generujące przerwania ma swój własny identyfikator. Podczas wystąpienia przerwania NVIC identyfikuje przerwanie a następnie wykonuje skok do adresu znajdującego się w tablicy wektorów dla określonego ID przerwania. Adres określa początek procedury obsługi przerwania. Oczywiście wykonywane są jeszcze procedury związane z samym przerwaniem tj. odłożenie na stos rejestrów procesora itp.
Aby włączyć możliwość generowania przerwać przez układ peryferyjny mikrokontrolera należy ustawić bit w rejestrze ISER kontrolera przerwań NVIC. Aby włączyć przerwania timera1 i timera3 należy ustawić 1 odpowiednio na 2 i 4 bicie rejestru ISER0.

7.4 SPI

SPI (*Serial Peripheral Interface*) - to szeregowy interfejs urządzeń peryferyjnych, umożliwiający komunikację między mikroprocesorem a urządzeniami peryferyjnymi. Umożliwia synchroniczną, szeregową, pełno-dupleksową komunikację oraz programowalną długość przesyłanych danych.
Płytka wyposażona jest również w dwa kontrolery SSP (*Synchronous Serial Port*), SSP0 i SSP1 – kontrolery umożliwiające operacje na SPI, SSI oraz magistrali Microwire. Posiadają one kolejkowanie FIFO, obsługę wielu protokołów oraz możliwość używania z wykorzystaniem General Purpose DMA.

Interfejs SPI znajduję się w LPC1768, choć jest przestarzałym rozwiązaniem, ponieważ był używany w modelach z serii LPC2xxx i łatwiej przetransferować z jego pomocą oprogramowanie do nowych urządzeń z serii LPC17xx, niż przy pomocy SSP0. W związku z tym, że interfejs SSP jest nowszy i bardziej wszechstronny, w naszym programie skorzystaliśmy właśnie z niego, zgodnie z tym co zaleca dokumentacja urządzenia. Używamy tylko jednego z dwóch interfejsów – SPP1, podłączonego do slave group 0 (APB0) jako dwunaste peryferium o adresie bazowym 0x4003 0000.

Ogólna zasada działania: komunikacja w trybie SPI odbywa się synchronicznie za pomocą czterech linii – MOSI (*Master Output Slave Input*) czyli dane do układu peryferyjnego, MISO (*Master Input Slave Output*) czyli dane z układu peryferyjnego, SCLK (*Serial CLocK*) czyli sygnał zegarowy (taktujący) oraz SS(*Slave Select*) do aktywacji wybranego układu peryferyjnego, która wskazuje urządzenie podrzędne (Slave). Choć może obsługiwać wiele połączeń slave oraz mater, lecz tylko jeden master i jeden slave może komunikować się przez magistralę w czasie przesyłu danych. Przesył danych w trybie SPI przebiega pełnodupleksowo, za pomocą ramek - od 8 do 16 bitów od master do slave, od slave do master w tym samym czasie zawsze przesyłany jest jeden bit. W praktyce najczęściej tylko w jedną stronę przenoszone są dane istotne.

Konfiguracja SSP1 w naszym programie przebiega w następujący sposób:
- Inicjalizację zasilania interfejsu zapewnia rejestr PCONP. Za SSP1 odpowiada w nim PCSSP1 bit nr 10, należy ustawić go na 1; wartość przy resecie to również 1, co oznacza, że przy resecie SSP1 jest aktywowane.
- Uruchomienie zegara koordynującego transfer danych obsługuje rejestr PCLKSEL0, gdzie należy ustawić bity 20 i 21 (symbol bitów: PCLK_SSP1).
- Ustawienie rejestru funkcji pinów na 2.
- Należy wybrać tryb pinów 0 (PINMODE0) tzn. tryb pull-up oraz w trybie OpenDrain wybrać not open drain (0).
- Należy ustawić w rejestrze PINSEL0 (port 0 – P0[15:0]) piny: P0.7, P0.8, P0.9.
Pin P0.7 odpowiada za SCK1 – zegar dla SSP1, który koordynuje transfer danych.
Pin P0.8 odpowiada za MISO1 – Master Wejście Slave Wyjście dla SSP1, używany do przesyłania danych z slave do master.
Pin P0.9 odpowiada za MOSI1 – Master Wyjście Slave Wejście dla SSP1, używany do przesyłania danych z master do slave.
- Należy zmienić ustawienie rejestru funkcji pinów na 0 (binarnie 00) .
- Następnie ustawiamy pin służący do wyboru Slave dla SSP1 (SSEL1) co realizuje rejestr PINSEL2 (port 2 – P2[15:0]) pin P2.2, który jest pinem General Purpose I/O.
- Jako ostatnia następuje inicjalizacja SSP1 przez 32-bitowe rejestry SSP1CR0 i SSP1CR1.

SSP ma również w naszym programie ustalone następujące wartości domyślne:
-taktowanie zegara wynosi 1 MHz,
-dane są przenoszone w 8-bitowych ramkach,
-tryb domyślny to MasterMode.

Główną zaletą pracy SSP w trybie SPI jest programowalność nieaktywnych stanów i faz zegara SCK poprzez bity CPOL (Clock Polarity – polaryzacja sygnału taktującego) i CPHA (Clock Phase – fazę próbkowania) oraz rejestr kontrolny SSPCR0.
Możliwe wartości CPOL i CPHA: (0,0), (0,1), (1,0), (1,1) – cztery kombinacje. Przepływ informacji przy każdej parze wartości przebiega zgodnie z opisem w user manual dostarczonym przez producenta. W programie używaliśmy CPOL i CPHA ustawionych domyślnie na 0, co oznacza następujący proces przesyłu danych: zegar ma wymuszony stan niski, SSEL ma wymuszony stan wysoki, jeśli w aktywowanym SSP zawarte są dane, SSEL ustawia sygnał master na stan niski, dane od slave przechodzą przez linię MISO, master ma aktywną linie MOSI; w połowie taktu zegara dane od mastera są przekazywane do linii MOSI; ponieważ slave i master przekazały swoje dane, pin zegara przechodzi w stan wysoki po upływie kolejnej połowy taktu zegara. Wniosek: dane są transferowane, gdy przednie zbocze impulsu wzrasta, a odbierane, gdy tylne zbocze impulsu maleje.

7.5 OLED

Wyświetlacz OLED, z którego korzystamy, to OEL Display Module, model: UG-9664HSWAG01.
Jego najważniejsze parametry:
- rozdzielczość 96x64 (sterownik dodany do wyświetlacza pozwala na obsługę rozdzielczości 132x64),
- monochromatyczny (biały).

Wyświetlacz jest podłączony do GPIO. Do jego inicjalizacji należy:
- ustawić odpowiednie piny na output: korzystamy z szybkiej magistrali AHB i rejestrów FIODIR, więc wstawiamy jedynkę na bit 1 i 7 FIODIR2(FIO2DIR) oraz jedynkę na bit 6 FIODIR0(FIO0DIR),
- sprawdzić czy zasilanie jest wyłączone: za pomocą FIOCLR2 – wstawiamy w tym rejestrze jedynkę na bit 1,
- zrealizować rekomendowany przez producenta inicjalizacyjny fragment kodu,
- włączyć wyświetlacz: wstawiamy do FIOSET2(FIO2SET) jedynkę na bit 1.

Obsługa wyświetlacza i transfer danych między nim a mikrokontrolerem realizowana jest za pomocą interfejsu SPI (zrealizowanego na SSP1, opisanym w poprzednim podrozdziale). Jak wspomniano w opisie SSP1, na wyświetlacz można wysyłać 8 bitów. Z tego powodu w trybie szeregowym wyświetlacz nie obsługuje wyświetlania pojedynczych pikseli, minimalna wartość do przekazania 8 pikseli, czyli chcąc wyświetlić 1 piksel na biało, pozostałe 7 należy ustawić na czarno. Z powodu pracy w trybie szeregowym wyświetlanie na ekran jest wolniejsze i uzależnione od szybkości magistrali SPI.

7.6 EEPROM

EEPROM to elektrycznie usuwalny rodzaj komputerowej pamięci nieulotnej wykorzystywanej w urządzeniach elektronicznych do przechowywania niewielkiej ilości danych, której poszczególne bajty mogą zostać usunięte i nadpisane.

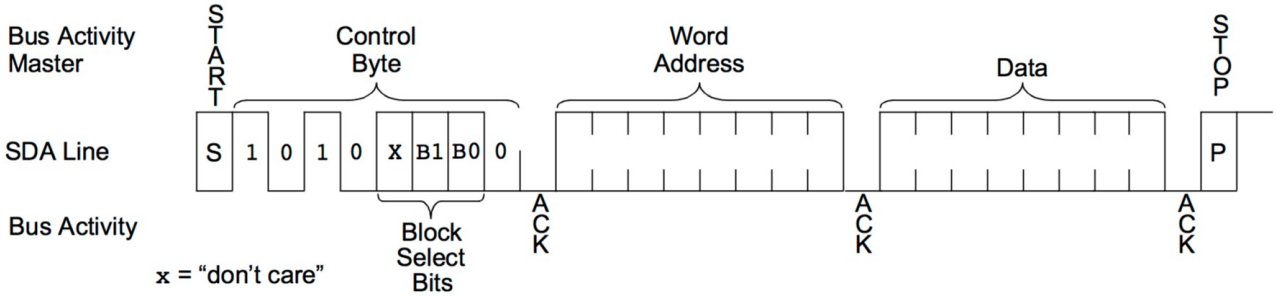
LPC1768 wyposażono w dwa rodzaje pamięci EEPROM. Nasz program wykorzystuje 8 kbit pamięć EEPROM o numerze seryjnym 24LC08B. Składa się ona z czterech bloków pamięci o rozmiarze 256 x 8-bit wykorzystujących do przesyłu danych magistralę I2C. Adres I2C: 0x50-0x53.

Mikrokontroler pracuje w trybie master, natomiast 24LC08 jako slave, jednak oboje mogą działać jako transmiter oraz odbiorca. Dany stan oraz warunki rozpoczęcia i zakończenia zapisu są określane przez mastera (wykorzystującego do tego linie SCL).

Pierwszym etapem (Start condition) poprzedzającym operacje odczytu/zapisu jest odbiór bajtu odpowiadającego za przygotowanie do przyszłych operacji. Pierwsze cztery bity zawierają kod (1010), który informuje o planowanej operacji zapisu/odczytu. Kolejne trzy bity decydują, który z czterech bloków pamięci zostanie wykorzystany, natomiast ostatni bit odpowiada za ostateczny wybór operacji („1" - odczyt, „0" - zapis).

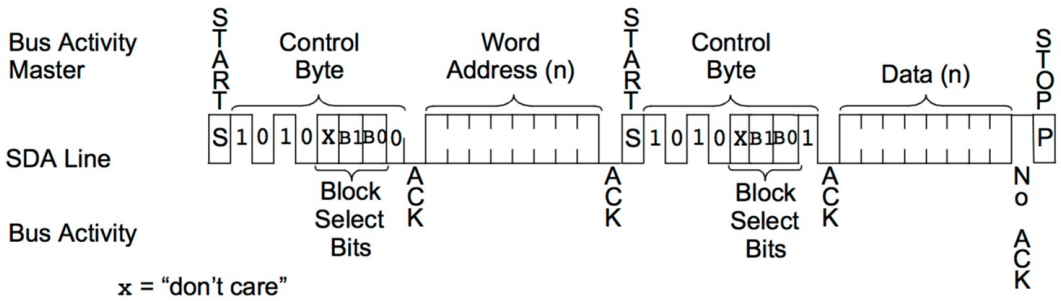
ZAPIS

Po otrzymaniu sygnału potwierdzającego (ACK), master wysyła kolejny bajt zawierający adres urządzenia, Po kolejnym potwierdzeniu następuje właściwy zapis pojedynczego bajtu w pamięci EEPROM do momentu przekazania bitu zatrzymania (Stop Condition).



ODCZYT

Operacja odczytu jest poprzedzona procedurą przygotowującą do operacji zapisu. Po przekazaniu adresu urządzenia oraz potwierdzeniu master wysyła kolejny bajt kontrolny, jednak tym razem ostatni bit kontrolny informuje o planowanej operacji odczytu. W tym momencie następuje transmisja 8-bitowego słowa aż do przekazania bitu zatrzymania.



7.7 AKCELEROMETR

MMA7455L to pojemnościowy akcelerometr pozwalający na dobór odpowiedniej czułości (+-2G, +-4G, +-8G dla 8-bitowej wersji). Do przesyłu informacji korzysta z interfejsu I2C lub SPI (u nas I2C). Oferuje cztery sposoby odczytu wartości na osiach. Funkcjonalność akcelerometru została przez nas wykorzystana do manualnego resetu gry.

Zastosowany w programie „Measurement mode" pozwala na ciągły odczyt wartości z trzech osi X,Y,Z. W momencie gdy odczyt jest zakończony (sygnalizacja flagi DRDY), kolejny pomiar jest już gotowy. Czułość została przez nas określona na poziomie 2G.

Operacja odczytu i zapisu jest standardową procedurą zgodną z protokołem I2C. (jedynym dodatkowym warunkiem MMA7455L do pracy w trybie slave jest ustawienie adresu urządzenia jako 0x1D).
Aktualne wskazania akcelerometru są przekazywane w 8-bitowych rejestrach wyjściowych:
- dla osi X w rejestrze XOUT8 o adresie 0x06
- dla osi Y w rejestrze YOUT8 o adresie 0x07
- dla osi Z w rejestrze ZOUT8 o adresie 0x08
Gdy pomiary dla wszystkich osi zostaną zakończone wspomniana wcześniej flaga DRDY (rejestr STATUS, adres 0x09) przyjmuje logiczne „1" aż do momentu gdy chociaż jeden z rejestrów _OUT8 jest w trakcie odczytu.

Reset gry następuje w momencie potrząśnięcia mikrokontrolerem. Aby do tego doszło iloczyn wskazań wszystkich osi akcelerometru musi przekroczyć 120 000.

7.8 JOYSTICK

Joystick, podobnie jak głośnik jest podłączony do pinów GPIO. Jednak tym razem to joystick wysyła dane do mikrokontrolera, więc przy jego inicjalizacji wszystkie piny zostaną ustawione na wejście. Zatem ustawiamy „0" na 15,16 i 17 bicie rejestru FIODIR0 oraz na 3 i 4 bicie rejestru FIODIR2. Podłączone piny odpowiadają stanom wciśnięcia joysticka P0.15 – DOWN, P0.16 – RIGHT, P0.17 – CENTER, P2.3 – UP, P2.4 – LEFT.

Aby móc korzystać z joysticka niezbędnym jest odczytanie jego położenia. Stan joysticka sprawdzany jest przez porównanie adresu portu z odpowiednimi maskami (wykorzystano koniunkcję bitową) tak aby odczytać stan pinów do których podłączony jest joystick. Funkcja odczytująca zwraca status, czyli sumę logiczną bitów odpowiadających danemu kierunkowi. (JOYSTICK_CENTER – 0x01, JOYSTICK_UP – 0x02, JOYSTICK_DOWN – 0x04, JOYSTICK_LEFT – 0x08, JOYSTICK_RIGHT – 0x10). Umożliwia to również przekazywanie kierunków pośrednich.
Następnie status jest przekazywany do autorskiej funkcji handle_input(), gdzie sprawdzamy warunek:

```
if((status & JOYSTICK_KIERUNEK) != 0)
```

Spełnienie warunku rozpoczyna proces weryfikacji prawidłowości wskazania użytkownika i w przypadku prawidłowej reakcji dodaje punkt do wyniku gracza.

7.9 FMEA

Każdy projekt realizowany z myślą o jego interakcji z użytkownikiem, powinien być zaopatrzony w FMEA (*Failure mode and effects analysis)* analizę rodzajów i skutków możliwych błędów systemu. Również w tym projekcie wdrożono taką analizę.

Awarie, które mogą wystąpić w systemie, cechują się różnymi prawdopodobieństwami i różnymi skutkami dla realizowania założeń programu. W związku z tym awariom można przydzielić różne stopnie krytyczności i prawdopodobieństwa. W tabeli 1 przedstawiono przewidywane możliwe awarie systemu z tymi współczynnikami.

Awaria głośnika – brak poprawnych dźwięków w grze, nie zmniejsza możliwości komunikacji między programem a użytkownikiem.
Awaria akcelerometru – brak lub błędne odczyty przeciążeń na osiach, może uniemożliwić rozgrywkę (ciągłe resetowanie gry), jeśli odczytywane wyniki będą znacząco zawyżane.
Awaria EEPROM – nie będzie możliwy zapis punktów zdobytych przez gracza do pamięci EEPROM, przez co zdobycie nowego najwyższego wyniku nie będzie poprawnie zapisane, najlepszy wynik nie będzie poprawnie odczytywany.
Awaria joysticka – uniemożliwi komunikację z graczem; jego reakcje na elementy wyświetlane na ekranie nie będą możliwe do sprawdzenia, w związku z czym cel gry przestanie istnieć.

Awaria OLED – uniemożliwi komunikację z graczem; jego reakcje na elementy wyświetlane na ekranie nie będą możliwa, ponieważ nic nie będzie się wyświetlać lub wyświetli się tylko część pikseli, w związku z czym cel gry przestanie istnieć.

Tabela 1.

Ryzyko	**Prawdopodobieństwo**	**Znaczenie**	**(Samo)wykrywalność**	**Reakcje**	**Iloczyn**
Awaria głośnika	0.1	2 (Nieznaczące)	1 (łatwa), wykrycie przez przerwanie obwodu (gdy awaria wynika z uszkodzenia cewki głośnika)		0.2
Awaria akcelerometru	0.1	2 (Nieznaczące)	1 (łatwa), wykrycie poprzez nieotrzymywanie sygnału potwierdzającego z magistrali I2C (ACK) po wysłaniu danych przez master.	Użycie alternatywnego interfejsu I2C; użycie innych pinów jako wyjście I2C; migotanie diodą jeśli powyższe nie zlikwiduje awarii.	0.2
Awaria timer0	0				0
Awaria timer1	0				0
Awaria timer3	0				0
Awaria IRQ	0				0
Awaria EEPROM	0.1	5 (Średnie)	1 (łatwa), wykrycie poprzez nieotrzymywanie sygnału potwierdzającego z magistrali I2C (ACK) po wysłaniu danych przez master.	Użycie drugiej pamięci EEPROM znajdującej się na LPC1768; migotanie diodą jeśli powyższe nie zlikwiduje awarii.	0.5
Awaria Joysticka	0.05	10 (Krytyczne)	1 (łatwa), wykrycie poprzez błędne wartości w rejestrach FIODIR0 i FIODIR2.	Użycie przycisków SW3 i SW4 jako sterowania (pojedyncze wciśnięcia odpowiadające lewemu i prawemu stanowi joysticka, przytrzymanie jednego przycisku i szybkie wciśnięcie drugiego - dla stanów góra i dół).	0.5
Awaria OLED	0.2	10 (Krytyczne)	2 (średnia), wykrycie dzięki funkcji ug_initialize zapewnionej przez producenta w sterownikach, po wykonaniu instrukcji sprawdzających połączenie z spi może zwrócic NULL przy jakiejkolwiek awarii	Podłączenie do I2C zamiast SPI; użycie innego interfejsu I2C/SPI; użycie innych pinów jako wyjście I2C/SPI; migotanie diodą i zatrzymanie programu jeśli powyższe nie naprawią awarii.	4
Awaria SPI	0				0
Awaria I2C	0				0

8. BIBLIOGRAFIA

- UM10360 LPC176x/5x User manual
  http://www.nxp.com/documents/user_manual/UM10360.pdf
- schematy
  http://skl.it.p.lodz.pl/~morawski/SCR&ES/Guides&Schematics/LPCXpresso_Base_Board_revB.pdf
  http://skl.it.p.lodz.pl/~morawski/SCR&ES/Guides&Schematics/LPCXpressoLPC1768revA.pdf
- EEPROM
  http://skl.it.p.lodz.pl/~morawski/SCR&ES/NotyKatalogowe/24LC08.pdf
- Głośnik
  http://www.alldatasheet.com/datasheet-pdf/pdf/8935/NSC/LM4811MM.html
- wyświetlacz OLED
  http://skl.it.p.lodz.pl/~morawski/SCR&ES/NotyKatalogowe/oled_UG-9664HSWAG01%20EVK%20user%20guide.pdf
- akcelerometr
  http://skl.it.p.lodz.pl/~morawski/SCR&ES/NotyKatalogowe/MMA7455L.pdf
  strona źródłowa z w/w:
- http://skl.it.p.lodz.pl/~morawski/SCR&ES/Devices.html