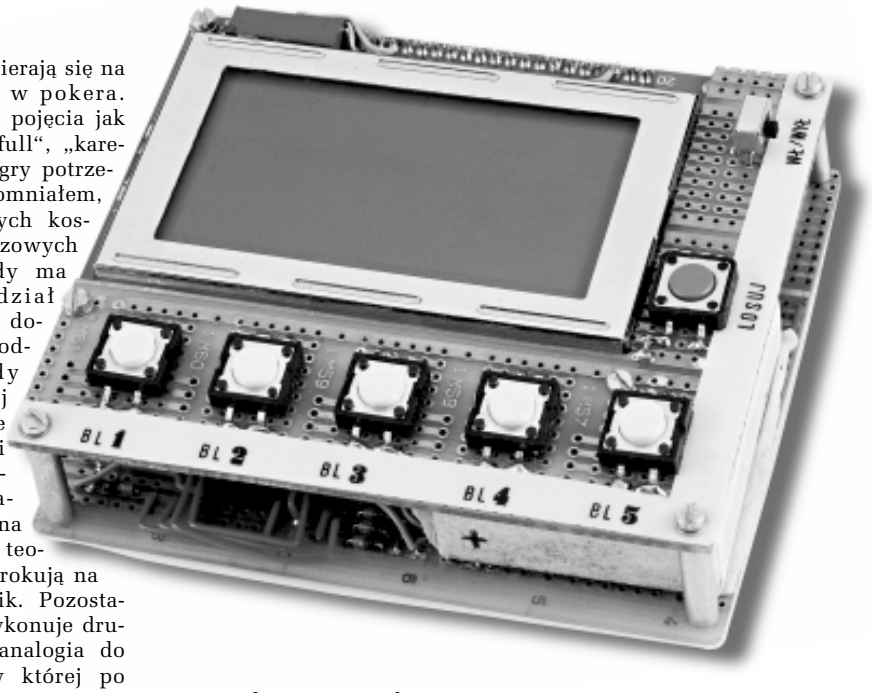
Dział "Projekty Czytelników" zawiera opisy projektów nadesłanych do redakcji EP przez Czytelników. Redakcja nie bierze odpowiedzialności za prawidłowe działanie opisywanych układów, gdyż nie testujemy ich laboratoryjnie, chociaż sprawdzamy poprawność konstrukcji.

Prosimy o nadsyłanie własnych projektów z modelami (do zwrotu). Do artykułu należy dołączyć podpisane ***oświadczenie, że artykuł jest własnym opracowaniem autora i nie był dotychczas nigdzie publikowany****. Honorarium za publikację w tym dziale wynosi 250,- zł (brutto) za 1 stronę w EP. Przysyłanych tekstów nie zwracamy. Redakcja zastrzega sobie prawo do dokonywania skrótów.*

Gra w kości



Projekt 088

Zaprojektowany i wykonany przeze mnie układ umożliwia elektroniczną symulację popularnej gry w pięć kostek. Znane są realizacje tej gry na różnych platformach komputerowych, poczynając od „Atari" a na PC-cie kończąc. W prezentowanym w artykule układzie zaimplementowano tę grę w wersji autonomicznej.

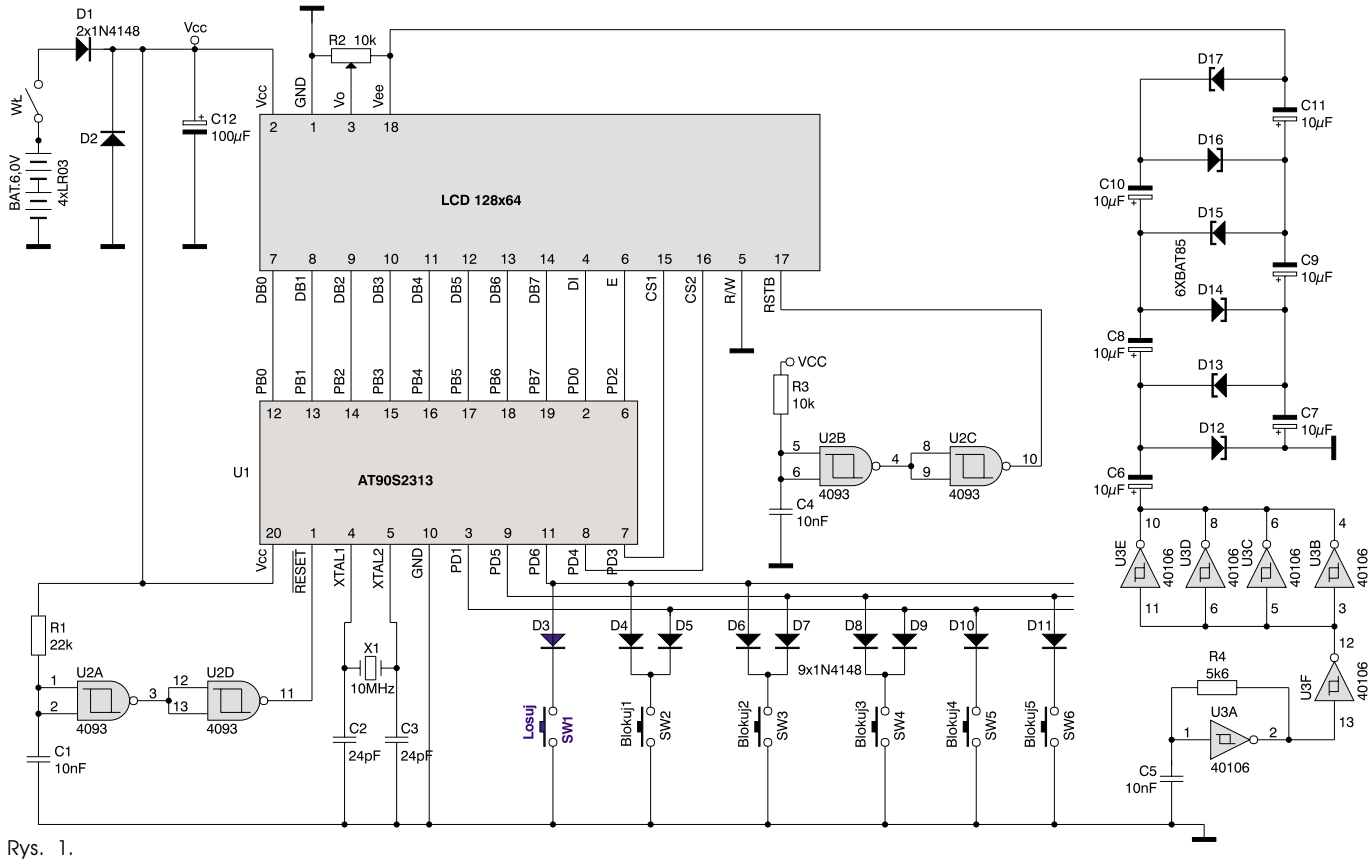
Zasady gry opierają się na karcianej grze w pokera. Wspólne są takie pojęcia jak „pary", „strit", „full", „kareta", „poker". Do gry potrzeba, jak już wspomniałem, pięć standardowych kostek do gier planszowych (w kartach każdy ma pięć kart). Udział w grze może brać dowolna liczba zawodników. Każdy z nich w swojej kolejce dysponuje dwoma rzutami kostek. Po pierwszym rzucie zawodnik zostawia na stole kostki, które teoretycznie dobrze rokują na oczekiwany wynik. Pozostałymi kostkami wykonuje drugi rzut. Jest to analogia do gry karcianej, w której po pierwszej rozgrywce zostawia się „dobre" karty, a pozostałe wymienia na inne z talii. Uzyskane wyniki rzutów kostkami notuje się w tabelkach. Wygrywa ten zawodnik, który uzyska największą liczbę punktów. Przedstawiony opis gry jest skrótowy, gdyż jest to bardzo popularna gra i jej zasady są znane, jeśli nie Czytelnikom, to ich bliskim lub znajomym. Moja wersja gry w kości jest bardzo podobna do wersji komputerowej. Do gry używamy kostek, które są wyświetlone na graficznym wyświetlaczu LCD. „Rzut kostkami" wykonujemy poprzez naciśnięcie przycisku „Losuj". Odpowiednie kostki „zostawiamy na stole" naciskając przyciski „Blokuj (1..5)". Na czynność blokowania mamy około 6 sekund, w czasie których na wyświetlaczu niewidoczny jest wykrzyknik. W tym czasie przycisk „Losuj" jest nieaktywny.

W prezentowanym projekcie zamierzano zademonstrować wykorzystanie wyświetlacza graficznego LCD w aplikacji amatorskiej. Dodatkowym celem było sprawdzenie możliwości użycia mikrokontrolera AT90S2313 do sterowania takimi wyświetlaczami. Wiele firm handlowych oferuje elektronikom wyświetlacze graficzne w różnych rozmiarach i cenach. Mnie udało się kupić wyświetlacz UG-13B-001K firmy Samsung. Jest to zgrabny panel o wymiarach 93,0x70,0x8,5mm. W oknie o wymiarach 70,7x38,8mm wyświetlanych może być 8192 ciemnoniebieskich punktów na szarym tle, zorganizowanych w 64 linie i 128 kolumn. Matrycą pikseli o rozmiarze 0,48x0,48mm każdy sterują firmowe układy scalone LSI CMOS. Układ KS0107B jest sterownikiem dla 64 rzędów pikseli. Dwa układy KS0108B sterują po 64 kolumny każdy i posiadają wewnętrzną pamięć RAM o pojemności 512 bajtów, czyli 4096 bitów w jednym układzie. Dzięki tej pamięci możliwe jest przesyłanie do wyświetlacza wcześniej przygotowanego wzoru „grafiki" obejmującego cały dostępny obszar pokryty pikselami. Wszystkie te dane zaczerpnąłem z dołączonej do wyświetlacza dokumentacji. Na stronie internetowej producenta dowiedziałem się, że zestawy układów użytych w panelu można łączyć w różne kombinacje, uzyskując wyświetlacze o rozdzielczości nawet 128x512 pikseli. Wyświetlacz użyty w „Grze w kości" zasilany jest dwoma napięciami: +5V do zasilania „logiki" oraz -12V do zasilania sterowników matrycy ciekłokrystalicznej. Do sterowania kontrastem stosuje się napięcie z odczepu potencjometru włączonego pomiędzy -12V i masę zasilania.

Aby wprowadzić do pamięci wyświetlacza jakiś obraz (grafikę), korzystamy z ośmiobitowego, dwukierunkowego portu, który jest dostępny na wyprowadzeniach 7..14. Proces sterowania wyświetlaczem wymaga operowania sygnałami: R/W - zapisz/czytaj, D/I - dane/instrukcja, E - zezwolenie na operację, CS1, CS2 - wybór pierwszej lub drugiej połowy ekranu, RSTB - sygnał zerujący. Kombinacją

Tab. 1.

D/I	R/W	Wykonywana operacja.
1	1	Czytaj dane zawarte w pamięci wyświetlacza.
1	0	Zapisz dane do pamięci wyświetlacza.
0	1	Odczytaj status zajętości wyświetlacza (trwa wykonanie instrukcji).
0	0	Wykonaj instrukcję.



Rys. 1.

stanów na wejściach D/I, R/W oraz wejściach danych można „zmusić“ wyświetlacz do wykonania jednej z siedmiu instrukcji sterujących. Zestaw instrukcji sterujących zawarto w **tab. 1**.

W tym miejscu wyjaśnienia wymaga sposób adresowania wewnętrznej pamięci wyświetlacza. Każda część ekranu jest podzielona na osiem ośmioliniowych poziomych pasków zwanych stronami. Dlatego do zaadresowania strony wystarczą nam trzy bity w instrukcji „Ustaw stronę“. Aby zaadresować kolumnę, użyjemy sześć bitów, ponieważ kolumn jest 64. Przy adresowaniu poszczególnych części ekranu należy pamiętać o wybraniu danej połowy sygnałem CS1 lub CS2. Dane dotyczące jednej kolumny w stronie mają wagi rosnące w kierunku od górnego piksela w dół. Należy wspomnieć o ułatwieniu w adresowaniu polegającym na tym, że po zaadresowaniu danej strony i kolumny oraz po przesłaniu danych pod ten adres następuje automatyczne zwiększenie adresu kolumny w wewnętrznym liczniku adresów. Dzięki temu już do końca tej strony przesyłamy dane dla poszczególnych kolumn bez konieczności wskazywania ich adresu w pamięci wyświetlacza. Podczas wykonywania operacji na wyświetlaczu musimy zdawać sobie sprawę z tego, że ich wykonanie trwa jakiś czas. Wtedy żadna inna instrukcja nie może być przyjęta do wykonania poza instrukcją „Czytaj status“. Instrukcja ta pozwala na odczytanie z wyjścia DB7 portu danych statusu zajętości sterownika wyświetlacza. Według dokumentacji wykonanie instrukcji może trwać od 4,3 do 12,9 mikrosekundy.

W **tab. 2** zamieszczono wszystkie instrukcje sterujące pracą wyświetlacza, wraz ze schematem kodu na liniach sterujących dla każdej z instrukcji. Aby „zapalić“ dowolny piksel na wyświetlaczu należy:

1. Przesłać do pamięci wyświetlacza adres strony w jakiej znajduje się dany piksel.
2. Przesłać adres kolumny, w której znajduje się piksel.
3. Przesłać dane dotyczące piksela, czyli 0, aby go wyświetlić.
4. Ustawić wyświetlacz w stan „włączony“.

Należy pamiętać, że jedną instrukcją przesyłamy dane dla ośmiu pikseli jednocześnie (słupek o wysokości 8 pikseli DB0..DB7).

Cały program realizujący grę w kości jest zawarty w pamięci Flash mikrokontrolera AT90S3213 firmy Atmel (schemat elektryczny przedstawiono na **rys. 1**). Pamięć mikrokontrolera została wykorzystana prawie do ostatniego bitu. Dlatego w grze dostępne są tylko podstawowe funkcje bez żadnych „wodotrysków“.

Oczywiście dla tych, którzy mają lepszą koncepcję realizacji gry udostępniam program w wersji źródłowej, który

Tab. 2. Lista instrukcji.

Instrukcja	Kod instrukcji										
	R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Wyświetlacz włączony/ wyłączony	0	0	0	0	1	1	1	1	1	1/0	1-wyświetlacz załączony 0-wyświetlacz wyłączony
Wyświetlaj od linii...	0	0	1	1	Początkową linią może być 0~63						Od tej linii na wyświetlaczu będą “zapalane” piksele.
Ustaw stronę (adres rzędów)	0	0	1	0	1	1	1	Strony 0~7			Ustaw adresy stron w pamięci wyświetlacza
Ustaw adres kolumn	0	0	0	1	Kolumny 0~63						Ustaw adresy kolumn w pamięci wyświetlacza
Odczytaj status.	1	0	Za-jęty	0	Wł/ Wył	Reset	0	0	0	0	Reset 1: jest w stanie reset 0:stan pracy Wł/Wył 1: wygaszony 0: wyświetla Zajęty 1: wykonuje instrukcję
Zapisz dane w pamięci	0	1	Dane do zapisania w pamięci wyświetlacza								DB0-najmłodszy bit DB7-najstarszy bit danych
Czytaj dane z pamięci.	1	1	Dane odczytane z pamięci wyświetlacza.								DB0-najmłodszy bit DB7-najstarszy bit danych

WYKAZ ELEMENTÓW

Rezystory
R1..R8: 220Ω
R9..R14: 10kΩ
R15: 4,7kΩ
R16, R17: 2,2kΩ
Kondensatory
C1, C2: 27 pF
C3: 4,7µF/16V
C4: 100µF/16V
Półprzewodniki
T1..T6: BC618
US1: ST62T65
W1..3: wyświetlacz LED 3x DB56-11GWA
Różne
X1: 8MHz
P1, P2: mikroprzełączniki

został napisany w asemblerze za pomocą programu „WAVRASM". Realizacja algorytmu działania programu sprowadza się do oczekiwania na naciśnięcie przycisku „Losuj" i odpowiedniej reakcji na fakt naciśnięcia tego przycisku. Sprowadza się to do wywoływania w odpowiednim momencie potrzebnej procedury.

Kilka słów wyjaśnienia wymaga schemat elektryczny układu do gry. Ponieważ wyświetlacz potrzebuje do działania aż 12 wyjść procesora, to aby obsłużyć 6 przycisków za pomocą 3 pozostałych wejść, musiałem zastosować matrycę diodową złożoną z diod D3..D11. Bramki A i D układu scalonego U2 powodują zerowanie mikrokontrolera po załączeniu zasilania. W tym samym czasie bramki B i C tegoż układu powodują zerowanie układów wyświetlacza. Ponieważ gra jest zasilana z czterech baterii typu „AAA" („R03" itp.), żeby wytworzyć potrzebne napięcie -12V do zasilania wyświetlacza, zdecydowałem się wykonać powielacz napięcia. Prostokątny sygnał, wytworzony przez multiwibrator zbudowany z inwertera U3A układu scalonego, jest formowany przez pozostałe inwertery tego układu. Cztery inwertery połączone są równolegle, aby zwiększyć wydajność prądową generatora. Do wyjść inwerterów dołączony jest powielacz napięcia, który pozwala osiągnąć napięcie ok. -12V pod obciążeniem. Zastosowane w powielaczu diody Schottky'ego pozwalają uzyskać napięcie wyjściowe o ok. 1,8V wyższe niż przy zastosowaniu typowych diod prostowniczych, np. 1N4148. Jest to o tyle ważne, że wyświetlacz pobiera optymalny prąd gdy napięcie Vee zawiera się w granicach 11,1..12,1V. Odchyłki napięcia - czy to na minus, czy też na plus - powodują zwiększony pobór prądu z zasilacza. Dzięki tym zabiegom cała gra pobiera mniej niż 20mA z baterii, co pozwala przez długi czas cieszyć się zabawą.

Jerzy Durka