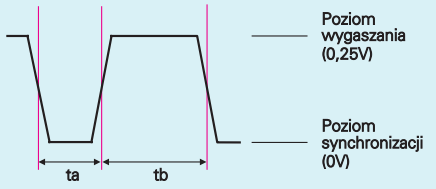
Gra telewizyjna „SQUASH”

Do czego to służy?

Squash to gra sportowa mająca wiele wspólnego z tenisem. Nazywana „tenisem w klatce”, ponieważ jest rozgrywana w pomieszczeniu zamkniętym. Do rozegrania partii squasha potrzebna jest piłka, dwie rakiety i zamknięte pomieszczenie z czterema ścianami. W artykule przedstawiam grę telewizyjną bardzo podobną do gry w squasha. Różni się od prawdziwego squasha tym, że występuje jeden gracz (jego „przeciwnikiem” jest wzrastająca prędkość piłeczki). Przypomina ona pierwsze gry wideo, jakie pojawiły się na świecie. Jest to bardzo prosta konstrukcja i może ją zmontować prawie każdy. Choć nie jest równie atrakcyjna, jak współcześnie produkowane gry TV, to jej własnoręczne wykonanie może dać mnóstwo satysfakcji.

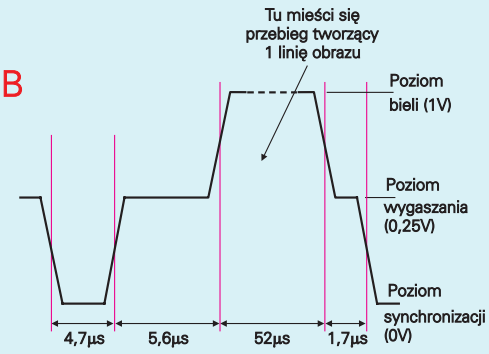
Rys. 1 Podstawowe wiadomości o sygnale VIDEO

A



Rodzaj impulsu	ta (µs)	tb (µs)
Przedni wyrównawczy (PW)	2,35	29,65
Synchronizacji pionowej (SP)	27,3	4,7
Tylny wyrównawczy (TW)	2,35	29,65
Linia wygaszania pionowego (LWP)	4,7	59,3

B



C

Ilość powtórzeń	5	5	4	16	289,5	5	5	5	16	289
Rodzaj impulsu	PW	SP	TW	LWP	LP	PW	SP	TW	LWP	LP
	Obraz nieparzysty					Obraz parzysty				

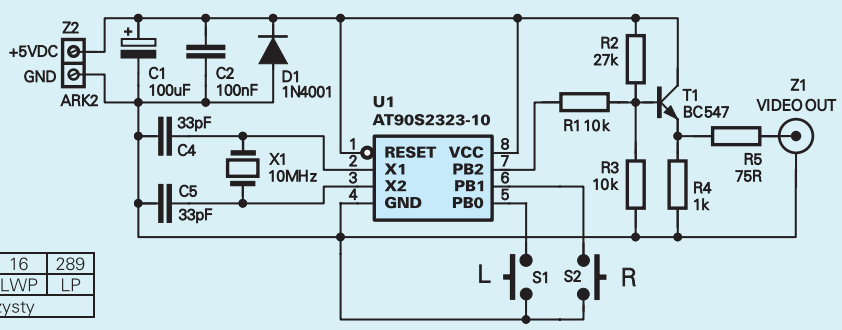
Jak to działa?

Gra zbudowana jest w oparciu o mikrokontroler AVR AT90S2323. Układ ten oprócz tworzenia świata gry widocznego na ekranie telewizora zajmuje się również generowaniem sygnału WIDEO. To właśnie konieczność jednoczesnego wytwarzania sygnału WIDEO i prowadzenia gry wymusiła zastosowanie szybkiego procesora z rodziny AVR. Do zrozumienia działania części elektronicznej układu niezbędna jest podstawowa wiedza o sygnale WIDEO. Na **rysunku 1** zamieściłem podstawowe informacje dla osób nieznających tego sygnału.

Krótko mówiąc, obraz widoczny na ekranie TV tworzą pojedyncze obrazy, których na ekranie w ciągu sekundy jest 25. Jeżeli podajemy do OTV sygnał z zewnątrz (np. magnetowid, DVD...), to każdy z tych obrazów jest efektem podania ciągu impulsów na wejście WIDEO telewizora. Ten ciąg impulsów nazywa się ramką, której przebieg przedstawia rys. 1C. Ramka pojedynczego obrazu dzieli się na dwa półobrazy: nieparzysty i parzysty, które razem tworzą jeden obraz. Każdy z półobrazów tworzą impulsy synchronizujące oraz impulsy, w których zawarta jest informacja o obrazie przekazywanym na ekran OTV. Na rysunku przedstawiłem również czas trwania poszczególnych impulsów oraz poziomy napięcia charakteryzujące każdy z nich. Muszę tu zaznaczyć, że opis dotyczy sygnału monochromatycznego, znacznie łatwiejszego w realizacji niż sygnał przedstawiający obraz kolorowy.

Żeby zatem stworzyć obraz monochromatyczny, wystarczy, że będziemy dysponować trzema napięciami: 0V, 0,25V i 1V. Można je wytworzyć bez problemu, wykorzystując 1 wyjście mikroprocesora i dodatkowo budując z kilku rezystorów prosty przetwornik A/C. Szczegóły realizacji widoczne są na schemacie ideowym na **rysunku 2**. Rezystory R1-R3 tworzą właśnie taki przetwornik. Dodatkowo, po ustawieniu portu PB2 jako wejście z wewnętrznym podciąganiem do plusa zasilania (pull-up) na wyjściu przetwornika uzyskujemy czwartą wartość napięcia, wynoszącą ok. 0,6 V. Do czego można ją wykorzystać? Np. do wyświetlenia koloru szarego. Można zatem bez problemu wyświetlić trzy kolory: czarny, szary i biały, wykorzystując do tego zaledwie

Rys. 2 Schemat ideowy



jedną końcówkę procesora. Wszystko to uzyskamy dopiero, pisząc odpowiedni program, który w odpowiednim momencie właściwie ją ustawi. O nim za chwilę, pozostało omówienie roli pozostałych elementów elektronicznych.

Tranzystor wraz z rezystorami R4 i R5 tworzy wtórnik emiterowy, który ma za zadanie dopasować sygnał do wejścia WIDEO telewizora.

Kondensatory C1 i C2 odsprzęgają zasilanie, zaś dioda D1 zabezpiecza przed uszkodzeniem procesora na skutek błędnego podłączenia zasilania.

Dlaczego to działa?

Cały „rozum" urządzenia zawarty jest w programie załadowanym do pamięci mikroprocesora. Program napisałem w asemblerze procesorów AVR, używając do tego takich narzędzi jak AVRStudio 3.56 oraz PonyProg2000. Dlaczego asembler? Napisanie programu do tej gry w języku wysokiego poziomu, takim jak C czy BASCOM, byłoby praktycznie niemożliwe. Procesor bez ustanku musi czuwać nad precyzyjnym skonstruowaniem całej ramki obrazu. Każda mikrosekunda mniej lub więcej oznacza znaczne zniekształcenie obrazu. Asembler w pełni umożliwia zapanowanie nad czasem wykonania każdej procedury programu z dokładnością co do jednego cyklu zegarowego. Niestety, okupione to zostało znacznie większym skomplikowaniem programu i mniejszą jego czytelnością.

Procesor przez większość czasu swojej pracy generuje sygnał WIDEO. Dzięki temu, że jest on bardzo szybki, możliwe stało się zrealizowanie kodu odpowiedzialnego za świat gry w ostatnim impulsie linii wygaszania pionowego obrazu nieparzystego. Na porcie PB2 panuje wtedy poziom wygaszania przez 59,3 mikrosekundy i procesor może w tym czasie wykonywać wszelkie operacje niezmieniające stanu portu PB2. W tym czasie procesor sprawdza, który przycisk jest wciśnięty, czy piłeczka została odbita, czy nastąpiła strata piłeczki przez gracza, itp. Podprogram zajmujący się światem gry, czyli dynamiką piłeczki i paletki, przyznawaniem punktów oraz kontrolą odbicia piłeczki nosi nazwę „BRAIN". Podczas wyświetlania piłeczki czy paletki program korzysta z ich gotowych pozycji wyliczonych właśnie w podprogramie BRAIN. Taka struktura programu umożliwiła szybkie rysowanie na ekranie TV piłeczki oraz paletki. Na **listingu 1** można zobaczyć fragment kodu, który realizuje impulsy wygaszania pionowego, natomiast na **listingu 2** fragment podprogramu BRAIN, który zajmuje się odczytywaniem stanu przycisków. Kompletny kod programu (w kilku plikach) wraz z plikiem eeprom_s.hex, w którym jest zawartość pamięci EEPROM, można ściągnąć ze strony internetowej EdW z działu FTP.

Piłeczka może przyjąć jedną z 95 pozycji w poziomie oraz jedną ze 183 pozycji w pionie.

Listing 1 Sposób generowania przez program impulsu wygaszania pionowego.

```
KEYS:
   sbis  $16,0x00   ;czy wciśnięty lewy?
   rjmp  L_ON       ;jeśli tak to skocz do L_ON
   sbis  $16,1      ;czy wciśnięty prawy?
   rjmp  R_ON             ;jeśli tak to skocz do R_ON
LR_DEL_RST:
   nop                    ;opóźnienie
   nop                    ; -||-
   nop                    ; -||-
   clr   r16
   ldi   r30,$65    ;zapisuje w RAM informacje o stanie licznika
   clr   r31
   st    Z+,r16
   st    Z,r16
   rjmp  KONIEC
L_ON:
   nop              ;opóźnienie
   nop            ; -||-
   ldi   r17,delay  ;delay - stała
   lds   r16,$65    ;pod adresem $65 licznik czasu wciśnięcia przycisku
   cpse  r16,r17    ;jeśli delay!=*$65 to inkrementuj licznik
   inc   r16
   ldi   r30,$65
   clr   r31
   st    Z,r16      ;zapisz licznik
   rjmp  KONIEC
R_ON:
   ldi   r17,delay
   lds   r16,$66
   cpse  r16,r17
   inc   r16
   ldi   r30,$66
   clr   r31
   st    Z,r16
   rjmp  KONIEC
KONIEC:
```

Listing 2 Obsługa przycisków

```
;-----------------------------------
;OBRAZ NIEPARZYSTY
;-----------------------------------
   ldi   imp,5            ;5 impulsów przednich wyrównawczych
SHPW_N:
   rcall SHPW_RUN         ;procedura generująca 1 impuls PW
   dec   imp
   brne  SHPW_N
;-----------------------------------
   ldi   imp,5            ;5 impulsów synchronizacji pionowej
SHSP_N:
   rcall SHSP_RUN         ;procedura generująca 1 impuls SP
   dec   imp
   brne  SHSP_N
;-----------------------------------
   ldi   imp,4            ;4 impulsy tylne wyrównawcze
SHTW_N:
   rcall SHTW_RUN         ;procedura generująca 1 impuls TW
   dec   imp
   brne  SHTW_N
;-----------------------------------
.
.
;-----------------------------------
;IMPULSY WYGASZANIA PIONOWEGO
;-----------------------------------
SHWP_RUN:                 ;procedura generująca 1 impuls wygaszania pionowego
   SYNC                   ;ustalenie na porcie PB2 stanu niskiego
   ldi   loop,12          ;ustalam opóźnienia
   rcall loop_run         ;wywołuję procedurę opóźniającą
   nop                            ;opóźnienie o 1 cykl zegarowy (100 ns)
   nop                            ; -||-
   BLACK                  ;ustawiam port PB2 jako wejście "wiszące w powietrzu"
   ldi   loop,191         ;ustalam opóźnienie
   rcall loop_run         ;wywołuję procedurę opóźniającą
   nop                            ;1 cykl zegarowy
   ret                            ;powrót z podprogramu
.
.
;-----------------------------------
;PROCEDURKA OPÓŹNIAJĄCA - PĘTELKA
;-----------------------------------
loop_run:                 ;pętla opóźniająca
   dec   loop                     ;odejmuje 1 od wartości inicjującej pętle
   brne  loop_run         ;jeśli zmienna loop==0 to koniec pętli
   ret                            ;powrót z podprogramu
```

Żeby gra nie była monotonna, paletka po włączeniu jest umiejscawiana w jednym z pięćdziesięciu losowo wybranych miejsc, stwarzając tym samym losowe warunki początkowe gry. Po zdobyciu wielokrotności 10 punktów zmieniają się wektory prędkości poziomej i pionowej piłki, w wyniku czego zwiększa się jej prędkość oraz kąt, pod jakim się porusza względem paletki.

Niestety, procesor 90S2323 nie posiada w swoim zestawie rozkazu mnożącego daną liczbę przez dowolną liczbę. Dlatego też wartości wektorów prędkości piłeczki oraz losowe położenia paletki mają z góry ustaloną wartość i pobierane są z pamięci EEPROM. W pamięci zapisanych jest 34 wartości wektorów prędkości poziomej oraz tyle samo dla prędkości pionowej. Gra posiada zatem 34 poziomy. Każdy może ustalić własne poziomy trudności, odpowiednio programując pamięć EEPROM. Odpowiednie wartości wektorów mieszczą się w komórkach pamięci EEPROM o adresach odpowiednio dla: wektorów poziomych: 3Ch..5Dh oraz dla wektorów pionowych: 5Eh..7Fh. Należy mieć na uwadze, że wektory prędkości pionowej muszą być dzielnikami liczby 180 oraz muszą być większe od 2, czyli np. 5,9,10,18,20, itp. Położenia paletki znajdują się w komórkach o adresach 0Ah-3Bh i muszą mieć wartość z przedziału od 1 do 83.

Montaż i uruchomienie

Układ można zmontować na płytce drukowanej pokazanej na **rysunku 3**. Montaż jest prosty i nie ma specjalnych wymagań co do jego sposobu realizacji. Oczywiście pod mikroprocesor stosujemy podstawkę. Po zmontowaniu układ powinien „ruszyć" bez żadnych dodatkowych regulacji.

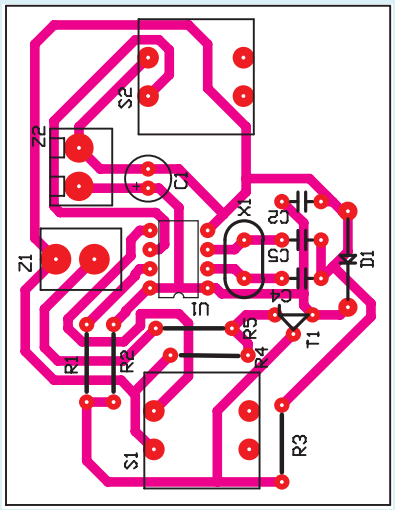
Grę do telewizora podłączamy za pomocą kabla WIDEO. Może okazać się konieczne zastosowanie dłuższego kabla, co jest nawet wskazane, ponieważ zbyt bliskie przesiadywanie przed telewizorem nie jest zbyt zdrowe. Wtedy jesteśmy zmuszeni do własnoręcznego wykonania kabla. W przypadku, gdy telewizor posiada wejście wideo w postaci gniazda CHINCH, wykonanie takiego kabla jest bardzo proste – wystarczą dwa wtyki CHINCH i parę metrów kabla. Jeżeli posiada złącze EURO, to podłączamy przewód sygnałowy do końcówki numer 20, natomiast masę do końcówki nr 17 wtyku EURO.

Do zasilania gry używamy zasilacza dającego napięcie 5VDC o wydajności prądowej kilkuset miliamperów. W przypadku użycia zasilacza o wydajności prądowej większej niż 1A należy szczególnie uważać na prawidłową polaryzację, gdyż istnieje ryzyko uszkodzenia diody zabezpieczającej D1 i w konsekwencji również mikroprocesora. Do zasilenia gry z powodzeniem można zastosować zestaw trzech baterii 1,5 V połączonych szeregowo.

Jeżeli ktoś zdecyduje się na obudowanie układu, musi się zastanowić nad wyprowadzeniem przycisków ponad obudowę. Jeżeli gra ma być używana bez obudowy, wtedy należy zabezpieczyć ścieżki płytki przed dotykaniem palcami ścieżek, stosując np. podstawkę z tworzywa sztucznego od strony ścieżek.

Po uruchomieniu gry na ekranie TV ukazuje się obraz, w którym liczba po lewej stronie oznacza ilość pozostałych „żyć", środkowe liczby to ilość zdobytych w grze punktów, a liczba po prawej to rekord, czyli maksymalna liczba punktów uzyskana od momentu włączenia gry. Za każde odbicie piłeczki otrzymujemy 1 punkt, a każda strata piłeczki oznacza stratę życia. „Ilość żyć" jest ustalona na 3. Po stracie ostatniego życia punkty są zerowane i rozpoczyna się nowa gra.

Rys. 3 Schemat montażowy



Piotr Wójtowicz
piotr.wojtowicz@edw.com.pl

Wykaz elementów

Rezystory
R1 ..... 10kΩ
R2 ..... 27kΩ
R3 ..... 10kΩ
R4 ..... 1kΩ
R5 ..... 75Ω

Kondensatory
C1 ..... 100µF/16V
C2 ..... 100nF
C3,C4 ..... 33pF

Półprzewodniki
D1 ..... 1N4001
T1 ..... BC547
U1 ..... AT90S2323-10 (zaprogramowany)

Różne
X1 ..... kwarc niski 10MHz
S1,S2 ..... przyciski duże
Z1 ..... gniazdo chinch do druku
Z2 ..... listwa ARK2
podstawka precyzyjna 8PIN

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2739