Programowana tablica świetlna

W dyskotekach pojawiają się coraz to wymyślniejsze efekty świetlne. Chociaż, jak wielu zauważyło, nie są to jedyne atrakcje dyskotek (latające krzesła są równie widowiskowe, o ile nie podążają w naszym kierunku!), to jednak warto przyjrzeć się opisywanemu urządzeniu.

Mając do dyspozycji Bascoma i odrobinę cierpliwości, możemy wykonać bajer rodem z dyskoteki. Zastosowanie takiego urządzenia to na pewno nie tylko zabawa.

Dzięki pamięci EEPROM i interfejsowi RS232 możemy zaprogramować dowolny tekst, a potem niezależnie od zaników zasilania rozkoszować się tekstem przewijanym w dwu różnych trybach.

Jak to działa?

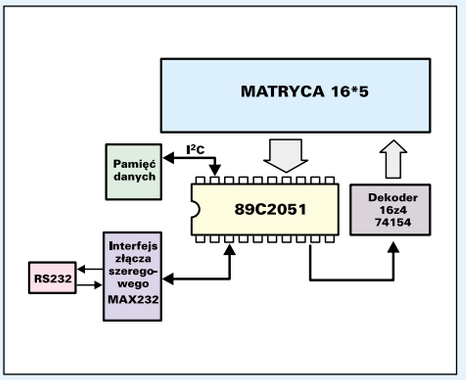
Schemat ideowy zamieszczony został na **rysunku 1**. Warto prześledzić także schemat blokowy z **rysunku 2.**

Sercem całego urządzenia jest bardzo popularny uP 89C2051, nie trzeba go chyba reklamować, a jego zalety były już wielokrotnie opisywane, tym razem z zalet wykorzystamy wbudowany układ obsługi portu szeregowego – UART i dosyć dużą wydajność prądową portu w stanie niskim.

Pomysł, według którego działa tablica, najlepiej zrozumieć analizując drogę pokonywaną przez dane.

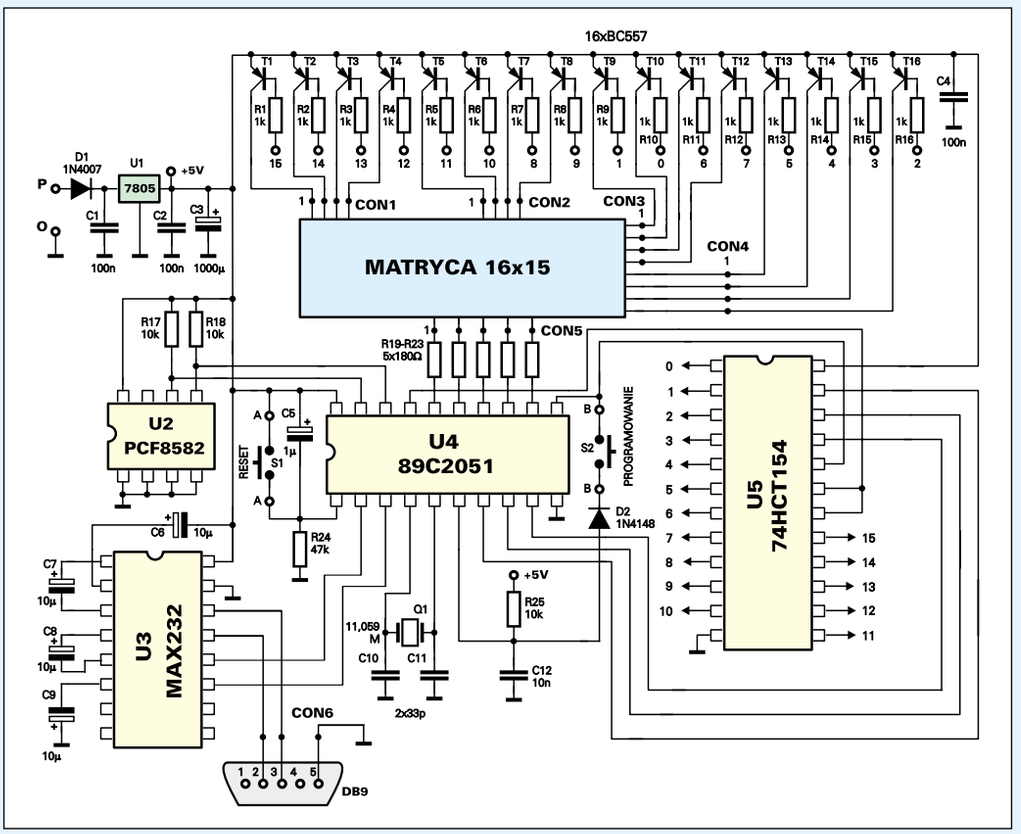
Na wejściu mamy standardową aplikację układu MAX232, dzięki niemu sygnały pojawiające się na złączu RS z łatwością zmieniają poziom na TTL, para kondensatorów towarzyszących kostce jest elementami współpracującymi z przetwornicą napięcia zabudowaną w układzie MAX232. Następnie dane są przesyłane do uP, który każdy odebrany znak umieszcza w pamięci EEPROM. Zastosowana pamięć to także dobrze znany Czytelnikom układ PCF8582. Zwarte do masy linie wyboru adresu wskazują, że układ jest widziany na magistrali I^2C pod adresem 160 do zapisu i 161 do odczytu.

Pojemność pamięci wyznacza długość tekstu, który będziemy wyświetlać i w zasadzie wynosi 256 bajtów (czytaj znaków), lecz jest pewne ale, bo niby skąd nasz uP ma wiedzieć, ile znaków zapisanych w pamięci to właściwy tekst, a ile to śmieci z poprzednich zapisów? W ten sposób tracimy jeden bajt na licznik poprawnych znaków, ale to nie wszystko, parę bajtów zżerają nam komendy specjalne. Ktoś zapyta „a to co



Rys. 2 Schemat blokowy

Rys. 1 Schemat ideowy



znowu“? - chodzi mianowicie o takie specyficzne znaczki jak „<”, „>” zastosowane do przełączania trybów wyświetlania. Będzie jeszcze o nich dalej.

Reszta elektroniki to właściwie kosmetyka elektroniczna. Duży układ w prawej części płytki to nic innego jak dekoder 16 z 4 zwiększający liczbę linii naszego uP. Aktywny stan wyjścia tego układu to stan niski, tymże stanem załączmy jeden z 16 tranzystorów PNP zasilających poszczególne kolumny matrycy.

Układ zasilania to standardowo stabilizator 7805 plus kondensatory. Tak dla ścisłości, dioda na wejściu ma tylko jeden cel - zabezpieczyć układ przed niewłaściwą polaryzacją, a to dlatego, że modelowy układ przez pewien czas pracował w samochodzie.

Program

Pora teraz na najsmaczniejszy kąsek opisu, czyli program zaszyty w uP:

Program składa się z dwóch części.

Część programowania jest aktywowana przerwaniem INT0. Naciskając guzik ***PROGRAMOWANIE*** uP, sprawiamy, że każdy znak wysłany poprzez złącze RS zapisywany jest w pamięci EEPROM, jednocześnie zwiększany jest licznik poprawnych znaków. Koniec tego trybu to naciśnięcie ESC. W ostatniej fazie tego trybu w komórce o adresie 00h zapisywana jest wartość licznika poprawnych znaków.

Procedura powyższego trybu jest prosta i wygląda tak:

```
Do

  Znak = Inkey
   Pom1 = Asc(znak)
    If Pom1 > 0 Then
      Incr Full
       Print Znak
       Call Zapis
       Waitms 10
      If Pom1 = 27 Then
       Print „end”
       Exit Do
      End If
    End If

Loop
```

Sam podprogram ZAPISZ to standardowa obsługa pamięci EEPROM opisana dokładnie na łamach „Bascom College”.

Drugi rodzaj pracy wykorzystywany na co dzień wygląda mniej więcej tak:

Timer0 pracuje w trybie 1 i wywołuje przerwanie co 1/50 sekundy na każdą kolumnę matrycy, co przy 16 kolumnach daje okres 1,25ms. W każdym przerwaniu załączany jest odpowiedni tranzystor zasilający matrycę, a jednocześnie na młodszą część portu P1 wystawiana jest wartość stosowna do odświeżanej pozycji i tak bez końca. Coś jednak musi się zmieniać w wystawianych na port wartościach, by uzyskać jakikolwiek efekt ruch. Cały urok układu polega na tym, że każda kolumna matrycy posiada przyporządkowaną sobie komórkę pamięci o nazwie *rov(1)* do *rov(16)* i gdy tylko liczba przerwań osiągnie stosowną wartość, następuje przepisanie w górę zawartości komórek. W ten sposób zmienna *rov(16)* przyjmuje zawartość zmiennej *rov(15)*, *rov(15)* z *rov(14)* i tak po kolei. Ktoś zapewne zapyta a zmienna *rov(1)*? Na pewno nie pozostaje pusta, jej zawartość jest odczytywana z tablicy opisującej wygląd znaków. W ten sposób mamy przesuwanie tekstu w lewo, ale o jeden znak. Niestety, aby tablica była uniwersalna i by można wyświetlać dowolny tekst, każdy znak zdefiniowany jest osobno. Nie widzicie jeszcze problemu? Co się stanie, gdy odczytamy wszystkie wartości opisujące naszą literkę np. A? No tak, program czytać będzie dalej, ale jakieś głupoty, w najlepszym wypadku odczyta kod literki B, ale nie o to nam chodziło. Dlatego też program po 4 odczytach ustawia bit *flaga*, a na to tylko czeka pętla główna programu. Odczyta sobie znak z pamięci EEPROM, następnie zdekoduje (polecenie *select case*) i odpowiednio ustawi wskaźnik dla polecenia *read* na kod właściwej literki. Ten kod znaku to nic innego jak wartość binarna przy wyświetlaczu WA.

Trochę to skomplikowane, ale przestraszę Was, że to nie wszystko. Mamy przecież dwa tryby przewijania:

Zwykły, czyli ciągłe rolowanie z regulowaną prędkością, oraz ten drugi, w którym najpierw przesuwana jest szybko cała zawartość tablicy, po czym tekst zatrzymuje się na chwilę sprawę, jak to działa, zostawiam jednak dla Was. Dla ułatwienia podam, że polega to na odpowiednim zwiększaniu i zmniejszaniu czasu pomiędzy kolejnym przepisywaniem wartości w komórkach *rov(x)*.

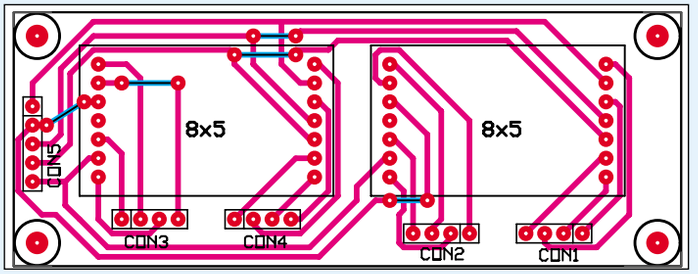
Aby wszystko działało jak należy, w programie nie mogło zabraknąć kodów znaków do wyświetlenia. Matryce mają wysokość 5 punktów świetlnych, żeby więc zachować proporcje, znaki mają szerokość 4 punktów, z czego czwarta kolumna jest zawsze pusta. Taki zabieg zwiększa czytelność. Każda litera i cyfra są zdefiniowane oddzielnie, i niestety zajmuje to sporo pamięci programu.

Montaż i uruchomienie

Tablica składa się z dwóch płytek drukowanych przedstawionych na **rysunkach 3** i **4**. Pierwsza płytka zawiera kompletny układ elektroniczny. Druga płytka to tylko wyświetlacze. Taki podział podyktowany jest budową wyświetlaczy. Do połączenia między płytkami dobrze jest użyć złączek goldpin. Montaż trzeba rozpocząć od zworek, potem należy zamontować podstawki i inne elementy, kończąc na włożeniu układów scalonych w podstawki.

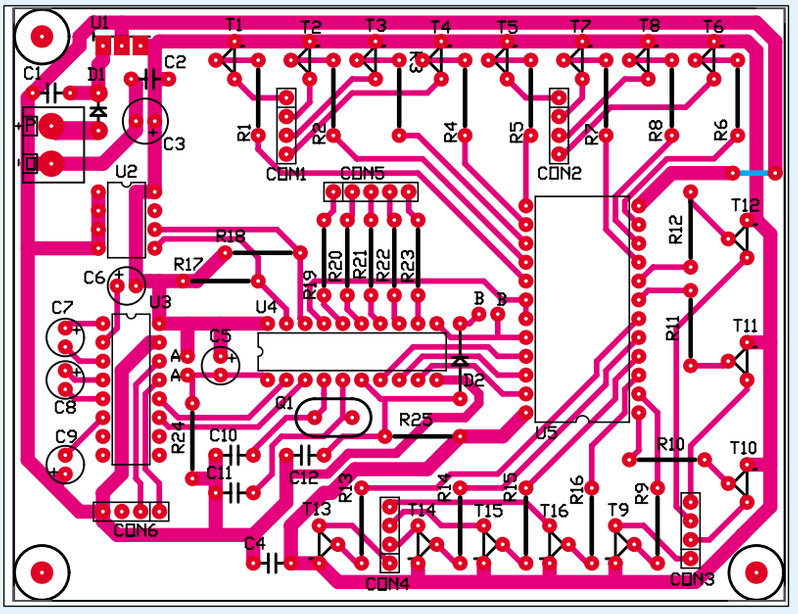
Z montażem nie powinno być żadnych problemów.

Przyciski S1,S2 należy zamontować na przewodach. Do zasilania urządzenia potrzebujemy napięcia stałego 7,5...15V. Podany zakres dzięki stabilizacji US 7805 jest na tyle szeroki, że na pewno znajdziemy jakiś zasilacz. Pobór prądu nie przekracza 150mA. Do pełni szczęścia potrzebny jest nam przewód połączeniowy do RS-a. W zależności od typu komputera możemy mieć gniazda typu

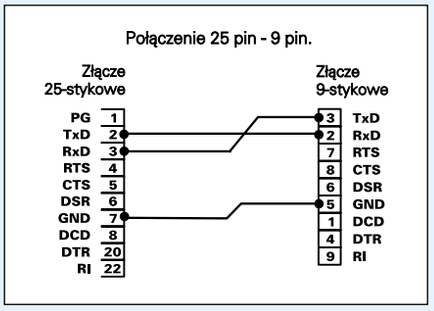


Rys. 4 Schemat montażowy wyświetlacza

Rys. 3 Schemat montażowy płyty głównej

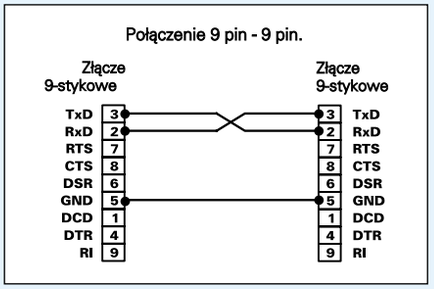


2*DB9 lub DB25 i DB9. Taki przewód można kupić w sklepie komputerowym, ale oczywiście można także zrobić samemu. Na **rysunkach 5** i **6** widać, jak powinny wyglądać takie przewody.



Rys. 5

Rys. 6



Jak łatwo zauważyć, potrzebujemy dwóch wtyków i kawałka tasiemki z trzema przewodami. Takie minimalistyczne rozwiązanie dobre jest jednak tylko na małą odległość (metr, dwa), gdyby zaszła potrzeba wykorzystania przewodu kilkumetrowego, wskazane jest użycie przewodu ekranowanego.

Wykaz elementów

Rezystory
R1-R16 ..... 1kΩ
R17,R18,R25 ..... 10kΩ
R24 ..... 47kΩ
Kondensatory
C1,C2,C4 ..... 100nF ceramiczny
C3 ..... 1000µF/9V
C5 ..... 1µF/9V
C6-C9 ..... 10µF/9V
C10,C11 ..... 33pF
C12 ..... 10nF
Półprzewodniki
D1 ..... 1N4007
D2 ..... 1N4148
T1-T16 ..... BC557
U1 ..... LM7805
U2 ..... PCF8582
U3 ..... MAX232
U4 ..... 89C2051
U5 ..... 74HCT154
Matryca 8x5 (2 szt.) np. OHT-358
Inne
Q1 ..... 11,059MHz
S1,S2 ..... µswitch
złącza goldpin
podstawki pod US
złącze DB9M

Składamy układ, podłączamy do komputera, załączamy zasilanie i - nic! O ile mamy pusty EEPROM, nie zobaczymy niczego. Musimy zaprogramować naszą tablicę.

Programowanie

Programowanie tablicy możliwe jest co najmniej na dwa sposoby.

SPOSÓB 1:

Zapuszczamy Bascoma, ustawiamy w menu Comunication, wstawiamy *baud* 2400, a *com* taki, jaki mamy aktualnie wolny. Innych parametrów nie trzeba zmieniać. W wypadku szukania wolnego coma można trochę poeksperymentować, pomocne jest czasami rozpoczęcie szukania od 1 i tak do skutku. A za sukces możemy uznać uruchomienie się *terminal emulatora* bez ostrzeżenia, że com jest zajęty.

Mamy już uruchomiony *terminal,* teraz naciskamy guzik *programuj* na tablicy i jeżeli tylko pojawi się napis „Tryb edycji", to po literce wpisujemy swój tekst. Koniec edycji to naciśnięcie ESC. Na początek nawiązanie komunikacji może wydawać się czymś trudnym, nie ma się jednak czego bać. Gdyby nawet com był wolny, a naciskanie guzika *programuj* nie dawało efektu, problem może być w zamienionych żyłach przewodu połączeniowego itp.

SPOSÓB 2:

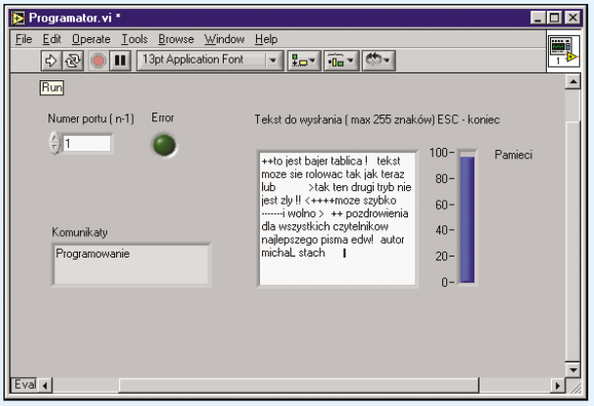
Sposób drugi jest o wiele bardziej widowiskowy. Napisałem prosty programik pracujący w środowisku LabView6.1. Użycie tego programu jest dziecinnie proste. Najpierw edytujemy własny tekst do wyświetlenia, nie wolno zapomnieć o *ESC* na koniec tekstu! Potem uruchamiamy aplikację strzałką w górnym lewym rogu i postępujemy według poleceń pojawiających się w oknie „komunikaty". Po chwili program poprosi o naciśnięcie guzika programowania i sygnalizując paskiem postępu, załaduje tekst. Narastający pasek postępu informuje o zajętości pamięci i gdyby przekroczył 100%, należy nieco okroić tekst oraz ponownie go załadować. Jeżeli wszystko przebiegło pomyślnie (strzałka z lewego rogu sama wróciła do poprzedniego koloru), wystarczy nacisnąć *Reset* na tablicy i po sprawie!

Uwaga: Zarówno przy programowaniu ręcznym, jak i z użyciem aplikacji LabView należy edytowany tekst wprowadzać **małymi** literami!!! Wynika to z faktu, że tablica posiada jeden rodzaj czcionki, od razu duże litery.

Kody sterujące

Aby uzyskać wspomniane dwa tryby przewijania tekstu, mamy 4 kody trybu rozpoznawane przez program. Kody te wprowadza się tak samo jak tekst, tyle że program po napotkaniu którejś z nich nie wyświetla tego znaku, a zmienia swoje działanie. Takie całkowicie programowe szantażowanie uP.

+ - zwiększenie prędkości przewijania
- zmniejszenie szybkości przewijania
> - zmiana trybu wyświetlania na szybkie przewijanie z chwilowym przytrzymaniem treści
< - przewijanie ciągłe



Rys. 7 - Przykładowy „screenshot" z aplikacji LabView

Możliwości zmian

Program z trudem mieści się w 2kB 89C2051, dlatego potrzebne były sztuczki.

Pierwsza z nich, raczej dobrze znana wszystkim, to podział programu na procedury. Opłaci się wydzielić z programu fragmenty wywoływane więcej niż raz i umieścić je np. na końcu. W miejscach gdzie były oryginalnie, wystarczy się do nich odwołać. Oszczędzamy w ten sposób miejsce zajęte przez ciągłe przepisywanie kodu.

Następny trik to fakt, że stosując zapis:

```
If a = 1 then
    .....
end if

if a = 2 then
    .....
end if
```

oszczędzamy za każdym poleceniu warunkowym 3 bajty w porównaniu z użyciem

```
if a = 1 then
    .....
elseif a  = 2 then
    .....
    .....
end if
```

, czy też *select case*.

Taka optymalizacja kodu sprawiła, że w ostatecznej wersji programu udało się jeszcze upchać obsługę i wyświetlanie cyfr !!!

Przyjemnego „rolowania" tekstów życzy

Michał Stach