Widmo, albo magiczna różdżka

kit 3017 AVT

Do czego to służy?

Jest to młodszy brat opisanej w EdW „Widmowej wstążki" (numer 7/2000). Układ trudno już nazwać wstążką, dlatego postanowiłem pozostawić samo *„Widmo"*. Tak jak i jego poprzedniczka, młodszy brat wykorzystuje efekt stroboskopowy po to, aby troszkę oszukać nasz wzrok. Tym razem jednak diodami nie steruje już prościutki generator lecz całkiem mądry procesor. Efekt?

Ta niezwykła zabawka mieszcząca się w rurce o średnicy 2cm i długości 35cm podczas machania wyświetla w powietrzu rozmaite serduszka, kółeczka, domki... Efekt jest niesamowity. Odnosi się wrażenie, że obrazki te wiszą w powietrzu!

Do tego, oprócz standardowych obrazków, każdy, kto ma komputer, może stworzyć własną kompozycję i umieścić ją w pamięci układu za pomocą kilku kliknięć! Program obsługujący *„Widmo"* został napisany w taki sposób, że akceptuje zwykłe dwukolorowe bitmapy które można przygotować w dowolnym edytorze graficznym.

Do wykonania układu zachęcam wszystkich którzy mają jeszcze w sobie coś z dziecka, mianowicie - lubią się bawić!

Jak to działa?

Schemat elektryczny układu widoczny jest na **rysunku 1.** Tak prosty elektrycznie układ realizujący dość złożone zadania udało się zrealizować dzięki zastosowaniu mikrokontrolera AVR 90S2313. Procesor ten świetnie sprawdził się w układzie ze względu na swoje możliwości sprzętowe.

Ale procesor to nie wszystko. Sam w sobie jest ślepy, głuchy i nie robi właściwie nic ciekawego. Aby „coś ciekawego" mogło się objawić został on wsparty przez 12 diod LED, przycisk i układ dopasowania poziomów RS232 – CMOS. Podczas czytania dokumentacji układu U1 konieczne okazało się ograniczenie prądu płynącego przez diody do około 10mA, aby nie przekroczyć maksymalnego prądu płynącego przez wyprowadzenia zasilania. Jest to realizowane przez rezystory R1-R12.

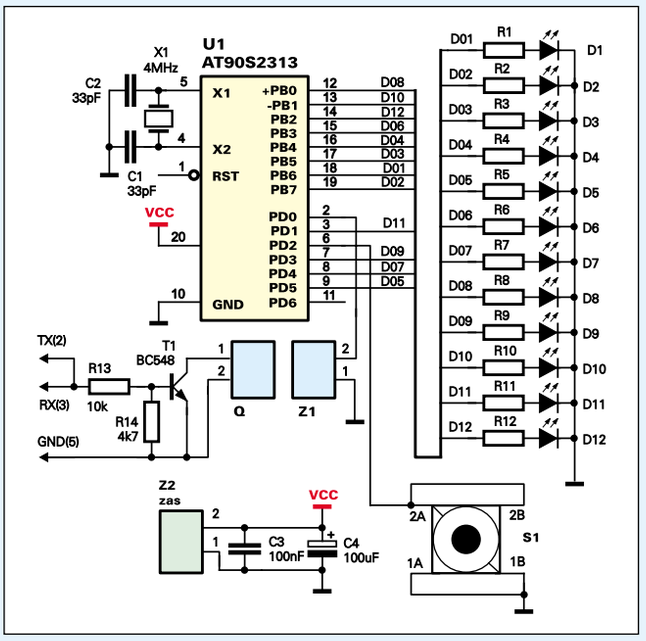
Układ pośredniczący między portem szeregowym komputera jest wykonany w trochę nietypowy sposób. Dzięki temu udało się osiągnąć oszczędności zarówno kosztów jak i miejsca na płytce. Zdziwienie może budzić zwarcie zacisków TX i RX w porcie. Bez obaw, komputerowi to nie zaszkodzi. Po prostu odbierze on znaki, które sam wysyła a dzięki takiemu rozwiązaniu z tej samej płytki możemy zrobić interfejs podłączany zarówno bezpośrednio do portu jak i przez zwykły kabel (w kablu linie RX oraz TX są skrzyżowane).

Tu właściwie kończy się sprzęt a zaczyna się oprogramowanie. Aby jednak dobrze zrozumieć działanie programu konieczna jest wiedza na temat tego, w jaki sposób dane o obrazkach są przechowywane w pamięci: Obrazki są jakby cięte na kawałki tak jak na **rysunku 2.** Każda kolumna która ma po 12 bitów (po jednym na każdą diodę) jest zapisywana jako bajt i jego połówka. Para sąsiadujących ze sobą linii jest zapisywana jako 3 bajty. W programie przyjęto następującą konwencję: najpierw bajt zawierający połówki, następnie dwa bajty „całe". Taki zapis, choć komplikuje nieco procedurę odczytywania danych w stosunku do zapisu, w którym każda kolumna byłaby zapisywana w oddzielnych dwóch bajtach, pozwala jednak skompresować obrazki w stopniu 3 do 4.

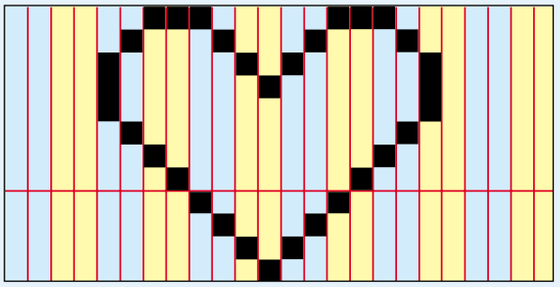
Tak naprawdę rysunek 2 jest tylko rysunkiem poglądowym. Obrazki są zapisywane w taki sposób, że całe bajty odpowiadają bezpośrednio diodom na porcie B, natomiast połówki danym na pinach portu D. Kolejnym odciążeniem procesora jest założenie, że obrazki muszą mieć parzystą liczbę kolumn. Dzięki temu procesor ma trochę mniej roboty a program zajmuje trochę mniej miejsca.

Takie zabiegi były konieczne. Po skompilowaniu końcowej wersji programu okazało się, że w pamięci ROM układu U1 zostały już tylko 2B!

Rys. 1 Schemat ideowy



Pełny ciąg danych zawiera jeszcze dodatkowo informacje o rozmiarach kolejnych obrazków. Dzięki temu program wie, gdzie dany obrazek się kończy, a zaczyna następny. Całość wygląda jak w **tabeli 1.** Ostatnia dana równa 0 oznacza, że rozmiar następnego obrazka wynosi właśnie 0 czyli, że to już koniec definicji obrazków. Dane w pamięci użytkownika (EEPROM) są przechowywane w identyczny sposób. Dzięki temu procedury odwołujące się do nich różnią się tylko jedną instrukcją: różne są sposoby dostępu do pamięci.



Rys. 2

Ponieważ wiemy już, w jaki sposób obrazki „siedzą" w pamięci, pora dowiedzieć się co układ dokładnie z nimi robi. Linie są po prostu wyświetlane kolejno z dużą częstotliwością. Podczas przesuwania diodami dzięki bezwładności ludzkiego oka widzimy całe obrazki. Nie ma tu żadnej synchronizacji z ruchami ręki. Wbrew pozorom nawet bez synchronizacji obrazki są świetnie widoczne, a jej brak ma ten plus, że nie jesteśmy zmuszani do jednostajnego machania układem tam i z powrotem. Możemy wywijać *„Widmem"* na wszystkie strony, obracać... i co tylko przyjdzie nam do głowy.

Ilość danych obrazka 1
Dane pierwszego obrazka
Ilość danych obrazka 2
Dane drugiego obrazka
...
Ilość danych obrazka n
Dane n-tego obrazka
0

Tab. 1

Kolejne linie obrazka są wyświetlane i ładowane w obsłudze przerwania *Timer0*. Kod jego obsługi widzimy na **listingu 1.** Naraz obsługiwane są dwie kolumny obrazka. Informacja o tym która kolumna była ostatnio wyświetlona znajduje się w zmiennej *Bit_przet_polowka.* Procedura przerwania uzależnia podejmowane działania od wartości tej zmiennej. Jeżeli jest ona wyzerowana, jedyne co jest wykonywane to wyświetlenie pierwszej kolumny. W przeciwnym wypadku po wyświetleniu drugiej kolumny wywoływana jest procedura *Laduj_linie.* Procedura ta pobiera z odpowiedniej pamięci dane następnych dwóch kolumn. Kod ten jest powtarzany z eksperymentalnie dobraną częstotliwością. Przy przerwaniu do timera nie jest wpisywana żadna wartość – następne przerwanie wystąpi po wykonaniu pełnego cyklu zliczania.

Listing 1

```
On_timer:
   If Bit_przet_polowka = 0 Then
      Gosub Wyswietl_1
      Bit_przet_polowka = 1
   Else
      Gosub Wyswietl_2
      Gosub Laduj_linie
      Bit_przet_polowka = 0
   End If
Return
```

Lecz co siedzi w tajemniczych, wymienionych wyżej procedurach? Na **listingu 2** widzimy prościutkie procedury wyświetlania obydwu kolumn. Zmienne *B_bait1* i *B_bait2* zawierają całe bajty wyświetlanego obrazka. Zmienna *B_polowki* zawiera połówki bajtów z kolumn. Zmienne te są inicjowane w podprogramie *Laduj_linie,* który widzimy na **Listingu 3.** To, skąd dane są ładowane, jest uzależnione od stanu bitu *Bit_bank_eeprom*. Zmienna *W_przet_linia* zawiera adres aktualnie przetwarzanej linii. Gdy jej wartość będzie wskazywała na ostatnią linię aktualnie wybranego obrazka zostanie ona zainicjowana wartością ze zmiennej *W_adres_obrazka.* Spowoduje to wyświetlanie obrazka od początku.

Obsługą wyświetlania w kodzie programu zajmują się jeszcze trzy podprogramy, są to: *Zmien_bank, Ustaw_pierwszy_obrazek, Zmien_obrazek.* Opiszę tylko ogólnie ich działanie.

Listing 2

```
Wyswietl_1:
   Portb = B_bait1
   D11 = B_polowki.0
   D09 = B_polowki.1
   D07 = B_polowki.2
   D05 = B_polowki.3
Return
Wyswietl_2:
   Portb = B_bait2
   D11 = B_polowki.4
   D09 = B_polowki.5
   D07 = B_polowki.6
   D05 = B_polowki.7
Return
```

Listing 3

```
Laduj_linie:
   If Bit_bank_eeprom = 0 Then
      B_polowki = Lookup(w_przet_linia , Dane_obrazki)
      Incr W_przet_linia
      B_bait1 = Lookup(w_przet_linia , Dane_obrazki)
      Incr W_przet_linia
      B_bait2 = Lookup(w_przet_linia , Dane_obrazki)
   Else
      Readeeprom B_polowki , W_przet_linia
      Incr W_przet_linia
      Readeeprom B_bait1 , W_przet_linia
      Incr W_przet_linia
      Readeeprom B_bait2 , W_przet_linia
   End If

   Incr W_przet_linia
   W_akumulator = W_adres_obrazka + B_wielkosc_obrazka
   If W_przet_linia >= W_akumulator Then
      W_przet_linia = W_adres_obrazka
   End If
Return
```

Pierwsza z nich umożliwia zmianę banku obrazków między standardowym a zawartym w EEPROM'ie. Gdy próbujemy zmienić bank na EEPROM sprawdza ona, czy pamięć użytkownika zawiera jakieś dane, jeśli nie to nie realizuje ona przełączenia. Druga procedura służy do inicjacji wszystkich zmiennych tak, aby rozpoczęło się wyświetlanie pierwszego obrazka z wybranego banku. *Zmień_obrazek* inicjuje wszystkie zmienne na następny obrazek chyba, że aktualnie wyświetlany jest ostatnim. Wtedy wywołuje ona podprogram *Ustaw_pierwszy_obrazek.*

Serce naszego programu, czyli pętla główna, jest malutkie, ale zupełnie wystarczające. Widzimy je na **listingu 4.** Niewielki rozmiar tego elementu programu udało się osiągnąć dzięki przerzuceniu większości zadań na przerwania. Procedura *On_przycisk* rozróżnia długie i krótkie naciśnięcie przycisku. W przypadku pierwszego zmienia bank pamięci, w przypadku drugiego zmienia aktualnie wyświetlany obrazek.

Listing 4

```
Petla_glowna:
Do
   Debounce Przycisk , 0 ,
On_przycisk
   Reset Watchdog
Loop
On_przycisk:
   B_akumulator = 0
   Do
      If B_akumulator = 255
Then
         Gosub Zmien_bank
         Gosub Mignij
         Goto Petla_glowna
      Else
         Incr B_akumulator
      End If
      Waitms 3
      Reset Watchdog
   Loop Until Przycisk = 1

   Gosub Zmien_obrazek
```

Po pętli głównej od razu widać, że zdecydowałem się na użycie watchdog'a. Czasami podczas wyłączania procesor zawieszał się, gasił diody i pobierał w tym stanie tak mało prądu, że kondensatory blokujące zasilanie wystarczyły mu na utrzymanie tego nieprzyjemnego stanu przez dość długi czas. Watchdog ustawiony na około 0,5 sekundy skutecznie usunął problem i dodatkowo przyczynił się do ułatwienia procedury odbioru danych z komputera.

Właśnie ostatnim fragmentem programu jest moduł odpowiedzialny za transmisję. Aby ułatwić zrozumienie algorytmu, warto najpierw poznać przyjęty format transmitowanych danych. Do układu muszą być przesłane następujące informacje: 1 bajt – ilość przesyłanych danych; 5 bajtów – identyfikator, kolejne bajty muszą tworzyć napis „WIDMO" z dużych liter; *n* Bajtów – dane w ilości określonej w pierwszym bajcie, te bajty są wpisywane bezpośrednio do pamięci eeprom.

Transmisja jest oparta na przerwaniu *URXC* – odbioru znaku z portu szeregowego. Pierwsza wersja układu korzystała z poleceń INPUTBIN. Powodowało to, że przed przesłaniem danych należało układ wyłączyć a następnie uruchomić go z przytrzymanym przyciskiem. Uznałem, że w obecnej modzie urządzeń PLUG&PLAY jest to rozwiązanie nieeleganckie. Dzięki wykorzystaniu przerwania wystarczy podpiąć działający układ do komputera, nacisnąć „Wyślij" i jeżeli wszystko jest w porządku to diody zgasną na chwilę a następnie układ rozpocznie wyświetlanie pierwszego z odebranych obrazków.

Jak się okazało BASCOM nie inicjuje portu szeregowego, gdy w programie nie ma poleceń typu INPUT, OUTPUT... etc. Konieczne

okazało się więc samodzielne wykonanie tej czynności. Sposób inicjacji przedstawia **listing 5.** Wszystko jest łatwe do zrozumienia za wyjątkiem może ustawienia prędkości transmisji. Układ UART procesora AT90S2313 wyposażony jest we własny generator transmisji. Potrzebną nam prędkość ustalamy wpisując do rejestru *UBRR* wartość wyliczoną ze wzoru:
Prędkość transmisji = częstotliwość zegara / 16(1 + UBRR)
Po przekształceniu otrzymujemy:
*UBRR = częstotliwość zegara / (16*prędkość transmisji) – 1*

Listing 5

```
Petla_glowna:
Petla_glowna:
Do
  Debounce Przycisk , 0 , On_przycisk
Reset Ucr.2'8bit
Ubrr = 207  '1200bps / 4MHz
Set Ucr.4   'Aktywacja odbiornika
Enable Urxc
```

Co dla częstotliwości transmisji 1200bps i zegarze 4MHz daje wartość 207.

Niska szybkość transmisji została wybrana głównie dlatego, że zapis do pamięci eeprom zajmuje trochę czasu i przy zbyt dużej szybkości układ „gubiłby" niektóre dane.

Całej procedury transmisji nie prezentuję ze względu na jej znaczną objętość. Zawiera ona sporo odgałęzień *IF... ELSEIF... ELSE* a to ze względu na obsługę ewentualnych błędów. W przypadku błędu zapala się odpowiadająca mu dioda, a następnie program przechodzi do miejsca pokazanego na **listingu 6.** Możliwe błędy transmisji i odpowiadające im diody przedstawia **tabela 2.**

Listing 6

```
Blad_transmisji:
  'Czekamy na reset z WDT
  Disable Interrupts
  Stop
```

Zapalona dioda	Opis błędu
D1	Zbyt duża liczba danych do przesłania
D2	Przesyłana dana powinna być ostatnia a nie jest = 0
D3	Zły format - identyfikator
D4	Błąd ramki - brak bitu stopu

Tab. 2

Cała obsługa błędów została stworzona w celu zapobiegnięcia przypadkowemu wymazaniu pamięci użytkownika.

Tak właśnie to działa. W razie jakiś niejasności polecam analizę kodu źródłowego, szczególnie fragment odpowiedzialny za odbiór danych.

Montaż i uruchomienie

Cały układ składa się z trzech płytek drukowanych zaprezentowanych na **rysunku 2**. Osobno montujemy płytkę główną, osobno płytkę baterii i osobno płytkę konwertera. Omawianie montażu rozpocznę od ostatniej płytki.

Konwerter jest na tyle prosty, że można go było co prawda zrealizować „na pająka" ale ja nie jestem zwolennikiem tej techniki. Zaprojektowałem maleńką płytkę która znakomicie mieści się w obudowie wtyczki DB9. Montaż elementu rozpoczynamy od przylutowania gniazda. To, jakiego ono będzie typu, (męskie / żeńskie) zależy od tego czy chcemy konwerter podpiąć bezpośrednio do portu, czy wykorzystamy w tym celu kabelek. Następnie montujemy elementy R13, R14 i T1. Do złącza Q należy przylutować za pomocą kabelków wtyk cinch. Całość umieszczamy w obudowie. Aby obudowa dała się zamknąć prawdopodobnie konieczne okaże się położenie tranzystora T1 na rezystorach. Konwerter jest gotowy. Odkładamy go i przygotowujemy się na zabawę wymagającą już większego zaangażowania.

W tym momencie dobrze jest mieć przygotowaną już rurkę której użyjemy na obudowę. Można zastosować rurkę PCV do zimnej wody o średnicy 21,2mm i grubości ścianki 1,7mm. Rurki tego typu powinny być bez problemu dostępne w sklepach z artykułami hydraulicznymi lub sklepach z armaturą.

Najpierw montujemy płytkę zasilacza. Jest on przystosowany do użycia w nim typowych baterii AAA. Kładziemy baterie na płytkę i przymierzamy, czy całość mieści się w rurce. Z obydwu stron miejsca na baterie na końcach należy wyciąć podłużne otwory. Miejsca te zaznaczone są grubszymi kreskami po stronie opisu. Dobrze jest najpierw wszystko przymierzyć, a potem robić otwory. W otwór po stronie plusa baterii należy wpasować sztywną blaszkę. Lutujemy ją od spodu dużą ilością cyny. Całość musi wytrzymać nacisk sprężyny z drugiej strony płytki. Po drugiej stronie lutujemy właśnie sprężynę. Możemy wymontować ją z oprawki na baterię. Powinna to być raczej obudowa z baterii węższej niż AA. Sprężyna zyska większą wytrzymałość jeżeli umieścimy ją tak, że jeden z jej zwojów będzie znajdował się w zrobionym przez nas podłużnym otworze.

Aby utrzymać baterie na wyznaczonym miejscu konieczne jest wykonanie obejm z blaszek. Umieszczamy je w bocznych wgłębieniach płytki. Dobrym rozwiązaniem jest zastosowanie po jednej blaszce na każdą baterię. Obejmy możemy przylutować do płytki dużą ilością cyny.

Na końcu płytki (tym, po stronie plusa) lutujemy zwyczajny przełącznik przykręcany do płyty czołowej. Lutujemy go za najniższą końcówkę a wyprowadzenie środkowe podłączamy za pomocą srebrzanki do punktu oznaczonego na płytce kółkiem.

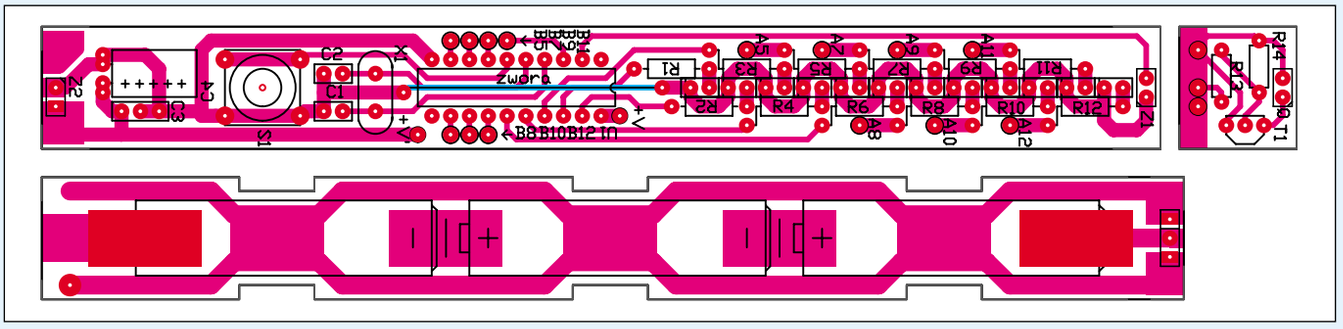
Sprawdzamy raz jeszcze czy wszystko pasuje do rurki. W razie jakiś problemów dokonujemy odpowiednich korekcji. Proszę zwrócić uwagę, czy obejmy baterii nie blokują płytki.

Ostatnią płytką jest płytka główna. Niestety ze względu na wąskość płytki i dużą ilość koniecznych połączeń nie udało mi się uniknąć kilku zworek, a nawet kilku połączeń, które będziemy musieli poprowadzić wąskimi kabelkami. Zaczynamy od zworki pod układem U1. Następnie łączymy punkty A z odpowiadającymi im punktami B. Dobrze jest użyć do tego cieniutkich kabelków. Oznaczenia punktów B na płytce mogą być trochę niejasne. Najwyżej umieszczony napis odpowiada najwyższemu punktowi. Na przykład po lewej stronie najwyższy punkt to B1. Należy zwrócić uwagę na prawidłowość przeprowadzenia tych połączeń. Łatwo na tym etapie popełnić błąd, który spowoduje błędne działanie układu. Ostatnim takiego typu połączeniem jest +V z +V'.

Dalszy montaż przeprowadzamy typowo. Diody LED nie zostały opisane ze względu na brak miejsca na płytce. Nie przeszkadza to jednak w ich montażu. Najlepiej użyć ultrajasnych lub niskoprądowych prostokątnych diod 5x3mm. Diody należy zamontować jak najbliżej płytki. Jeśli ktoś chce użyć diod okrągłych zalecałbym montaż rezystorów R1-R12 po stronie druku.

Przycisk S1 powinien mieć zdejmowany klawisz dzięki czemu całość uda się wsadzić

Rys. 2 Schemat montażowy



do rurki. W innym przypadku klawisz będzie się blokował przy próbie włożenia.

Na końcu płytki znajduje się miejsce do podłączenia konwertera. Sposób jego podłączania zależy od konstruktora. Ja przylutowałem do płytki gniazdo cinch za pomocą odcinków srebrzanki.

Płytkę główną próbujemy włożyć do rurki. Jeśli się nie mieści postępujemy z nią tak samo jak z płytką zasilania. Jeśli diody są ewidentnie za wysokie, może zajść konieczność ich zmiany na inne.

Płytki główną i zasilania łączymy ze sobą za pomocą odcinków srebrzanki. Dodatkowo od spodu połączenie można wzmocnić za pomocą cyny.

Elektrycznie urządzenie jest już sprawne. Pozostaje nam tylko wykonać obudowę. Z rurki obcinamy kawałek trochę dłuższy niż połączone płytki zasilacza i główna. Przykładamy go obok widma i zaznaczamy miejsce na wyświetlacz i przycisk. Należy zwrócić uwagę na to, żeby nie zostały żadne zadziory po wewnętrznej stronie rurki. Po stronie włącznika możemy wykonać zaślepkę z pokrywki z pudełka po filmie fotograficznym. W tym celu wycinamy jego środkową część – utworzy to coś na kształt kapsla. Rurka jest trochę za szeroka, aby brzegi naszej zaślepki mogły ją objąć. Należy jej koniec piłować naokoło tak długo, aż zaślepka da się luźno nałożyć. W zaślepce należy wyciąć otwór na przełącznik.

Po złożeniu całości mamy już pełnowartościową zabawkę! Otwór na diody można upiększyć za pomocą prześwietlonej błony fotograficznej. Osobiście nie przewidziałem zaślepki na górę obudowy. Uważam, że nie jest ona konieczna.

Obsługa układu

Chociaż już o tym wspominałem, chcę te informacje zebrać osobno, ponieważ mogły się one wydać za bardzo rozrzucone po treści artykułu.

Obsługa jest banalna: krótkie naciśnięcie przycisku powoduje zmianę wyświetlanego obrazka. Długie, aż do zamigania wyświetlacza, powoduje zmianę banku pamięci.

W celu programowania układu nie trzeba nic robić oprócz podłączenia układu przez konwerter do wolnego portu szeregowego PC-ta. Układ jest cały czas gotowy na przyjmowanie danych.

Dodatki

Oprócz samego układu przygotowałem dla Was dwa programy.

Pierwszy o nazwie *„BitmapToWidmo”* będzie wsparciem dla programistów. Jest to program, którego sam używałem. Nie posiada on żadnych wodotrysków, ponieważ po pierwsze pisałem go na przysłowiowym „kolanie”, a po drugie wiem, jak różne bajery denerwują, gdy ciężko pracujemy główką. Jego zadanie jest proste: po położeniu na okienku obrazka, jeżeli jego format jest właściwy umieszcza on w schowku tekst, który możemy wkleić do programu w BASCOM'ie. Umieszcza on najpierw ilość danych obrazka a następnie po znaku [ENTER] dane. Musimy jedynie dopisać słówko ***Data*** przed obydwoma liniami i wykasować przecinek z końca ostatniej linii. Ten nadmiarowy przecinek wynika ze sposobu przekształcania obrazka na dane i braku czasu na umieszczenie procedur korekcji.

Mimo wymienionych niedogodności narzędzie to naprawdę jest wygodne i bardzo pomaga. Możecie mi wierzyć – obrazek pierwszego serca przekształcałem na kartce! Okropna robota.

Drugi program wyposażony już w wiele upiększeń nazywa się „ProgramatorWidma”. Jest to aplikacja dokonująca transferu obrazków do pamięci użytkownika EEPROM *„Widma”*. Jest on wyposażony w plik pomocy uruchamiany za pomocą klawisza F1. Plik pomocy starałem się napisać prosto i treściwie tak, że zapoznanie się z programem nie powinno zająć więcej niż 5 minut. Z tego względu tutaj ograniczę się do stwierdzenia, że obsługa *„Widma”* z pomocą *„ProgramatoraWidma”* jest bardzo prosta i przyjemna. Sprowadza się właściwie do przeniesienia obrazków na okno programu i przyciśnięcia przycisku wyślij. Polecam zapoznanie się z tym programem, ponieważ dzięki niemu można dowiedzieć się paru dodatkowych rzeczy o układzie.

Obydwa programy akceptują monochromatyczne bitmapy o wysokości 12 pikseli, przy czym ich szerokość musi być liczbą parzystą.

Radosław Koppel

Wykaz elementów

Rezystory
R1-R12 ..... 300Ω
R13 ..... 10kΩ
R14 ..... 4,7kΩ
Kondensatory
C1,C2 ..... 33pF
C3 ..... 100nF ceramiczny
C4 ..... 100µF/16V
Półprzewodniki
D1-D12 ..... Diody LED 3x5mm
T1 ..... BC548
U1 ..... AT90S2313 (zaprogramowany)
Różne
S1 ..... Przycisk ze zdejmowanym klawiszem
X1 ..... Rezonator kwarcowy 4MHz
2-pozycyjny, przykręcany na płytę czołową, przełącznik zasilania
Sprężynka z pudełka na baterie
Blaszki (np. z dużego przekaźnika)
Gniazdo cinch
Wtyk cinch
Wtyk DB9 w wersji uzależnionej od typu układu (patrz tekst)
Obudowa na wtyk DB9

Płytka drukowana jest dostępna w sieci handlowej AVT jako kit szkolny AVT-3017