Gigantyczny zegar

Elektroniczny zegar to niewątpliwie jeden z „żelaznych punktów" każdego elektronika. Prawie każdy hobbysta stawia sobie za punkt honoru wykonanie zegara cyfrowego.

W EdW zaprezentowano już kilka zegarów. Teraz przyszła kolej na zegar-gigant z ogromnymi wyświetlaczami. Jak poświadcza fotografia okładkowa, sześciocyfrowy wyświetlacz ma 124cm długości i 27cm wysokości. Wskazanie jest czytelne nawet z odległości kilkuset metrów. Prezentowany imponujący wyświetlacz zawiera 770 diod LED.

Co bardzo ważne, każda cyfra tego gigantycznego wyświetlacza zmontowana jest z kilku wąskich pasków płytki drukowanej, a to radykalnie zmniejsza koszty wyświetlaczy, które przecież decydują o całkowitym koszcie zegara. Ponieważ wąskie paski płytki i popularne 5-milimetrowe diody są dziś naprawdę niedrogie, budowy tego imponującego zegara mogą się także podjąć osoby z mniej zasobnym portfelem. Z modułem mogą też współpracować klasyczne 20- i 14-milimetrowe wyświetlacze LED, co udowadnia **fotografia wstępna**.

Specjalnie dla tego zegara wspólnie ze Zbyszkiem Orłowskim zaprojektowaliśmy aż pięć różnej wielkości wyświetlaczy, zbudowanych z pojedynczych diod LED. Więcej szczegółów można znaleźć w krótkim artykule *Gigantyczne wyświetlacze LED* w tym numerze EdW. Sterownik zegara zrealizowałem na mikroprocesorze, co oczywiście genialnie uprościło konstrukcję. Wiem, że u wielu czytelników słowo mikroprocesor natychmiast budzi nieprzepartą odrazę i niechęć do takiego rozwiązania. Nie będę się jednak wdawał w dywagacje o przyczynach, słuszności i sile takich odczuć. Jeśli i Ty masz opory, przyjmij że sterownik zegara to specjalizowany, 20-nóżkowy układ scalony o symbolu *Gigant2002*, którego budowy wewnętrznej i działania wcale nie musisz rozumieć. I tym prostym sposobem pozbędziesz się kłopotu!

Zachęcam wszystkich do przeanalizowania prezentowanego rozwiązania. Pożytek odniosą z tego nie tylko „procesorowcy", ale i ci, którzy realizują swoje układy tradycyjnymi metodami. Zastosowane rozwiązania mogą być wykorzystane w innych tego rodzaju konstrukcjach.

Zaletą zegara jest intuicyjna obsługa za pomocą dwóch przycisków. Wykorzystany prosty sposób został przetestowany w praktyce, gdy poprosiłem kilka przypadkowych osób o ustawienie czasu na zegarze. Wszystkie szybciutko poradziły sobie z tym zadaniem, mimo że nie otrzymały żadnych wskazówek w tym zakresie (zegar cyfrowy nie ma wskazówek).

Miłośnicy mikroprocesorów zapewne z uwagą przeanalizują program. Ponieważ zgodnie z przyjętymi zasadami, program zostaje udostępniony na naszej stronie internetowej, można go zmodyfikować, wzbogacić lub uprościć, by jeszcze bardziej dostosować układ do własnych potrzeb. Można nawet zastąpić kostkę 89C2051 procesorem AVR 90S2313, mającym identyczny układ wyprowadzeń.

Dodatkową mobilizacją do własnej aktywności jest konkurs, ogłoszony na końcu artykułu.

Obsługa

Normalnie układ zlicza czas, pokazując jednocześnie godziny, minuty i sekundy. Do ustawiania służą przyciski S1 i S2. Podczas normalnej pracy przycisk S2 jest nieczynny. Naciśnięcie przycisku S1 spowoduje, że na wyświetlaczu zacznie migać pierwsza cyfra – i wtedy można ustawić dziesiątki godzin za pomocą S2. Kolejne naciśnięcie S1 spowoduje miganie drugiej cyfry i z pomocą S2 można ustawić jednostki godzin. Kolejne dwa naciśnięcia S1 pozwolą ustawić minuty. W czasie ustawiania godzin i minut ustawiana jest zawsze tylko jedna cyfra, bez wpływu na pozostałe. Piąte i szóste naciśnięcie S1 spowoduje miganie wyświetlaczy sekund. Naciśniecie S2 spowoduje wtedy wyzerowanie sekund. Jeśli licznik sekund pokazuje liczbę 0...29, nastąpi po prostu wyzerowanie, jeśli natomiast w chwili naciśnięcia S2 wskazanie sekund wynosi 30...59, oprócz wyzerowania nastąpi też zwiększenie licznika minut, ewentualnie godzin. Ma to duże znaczenie praktyczne, ponieważ często zegar jest korygowany na sygnał z radioodbiornika, nadawany o pełnej godzinie.

Kolejne siódme naciśnięcie spowoduje powrót do normalnej pracy zegara.

Dokładność wskazań zależy od stabilności zastosowanego rezonatora kwarcowego. W układzie przewidziano trymer, który nawet z popularnym kwarcem pozwoli uzyskać dużą precyzję, zwłaszcza gdy zegar będzie pracował w mieszkaniu, gdzie wahania temperatury są niewielkie.

Opis układu

Schemat ideowy zegara pokazany jest na **rysunku 1**. Do punktów pokazanych z prawej strony schematu dołączony jest sześciocyfrowy wyświetlacz LED. Schemat podstawowej wersji wyświetlacza pokazany jest na **rysunku 2**.

Jak widać, jest to układ multipleksowy, wykorzystujący wyświetlacze ze wspólną anodą.

Podczas pracy w danej chwili czasu świeci tylko jeden wyświetlacz. Oznacza to, że tylko na jednej z linii A1...A6 występuje dodatnie napięcie zasilające. O tym, które segmenty tego wyświetlacza będą zaświecone, decyduje stan linii A...G. Punkty te są zwierane do masy.

Sercem układu jest popularny mikrokontroler AT89C2051, pracujący z kwarcem o częstotliwości 12MHz. Trymer C6 pozwala precyzyjnie ustawić częstotliwość oscylatora i tym samym uzyskać dużą dokładność zegara.

Normalnie układ zasilany jest z zasilacza sieciowego (niekoniecznie stabilizowanego) o napięciu 6,5...18V, dołączonego do punktów P, O. Stabilizator U1 (7805) zapewnia odpowiednie napięcie pracy mikroprocesora. Wtedy zwora ZW1 musi być przerwana. Diody Schottky'ego D1, D2 zapewniają bezprzerwowe zasilanie. Przy braku napięcia sie-

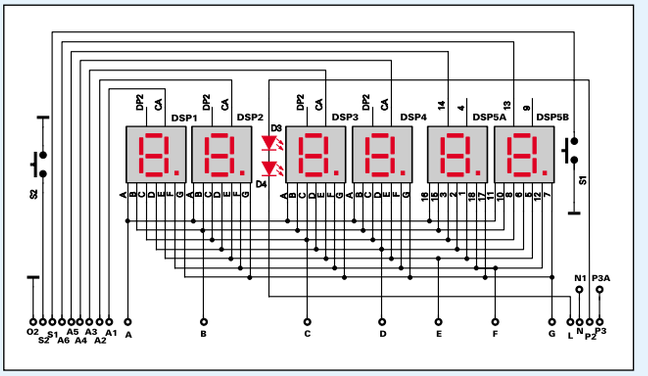
ci, mikroprocesor zasilany jest z baterii rezerwowej 3...4,5V, dołączonej do punktów P1, O1.

Napięcie VCC przy zasilaniu z sieci wynosi około 4,7V (5V minus spadek napięcia na diodzie D1). Źródłem zasilania może być zasilacz stabilizowany o napięciu 5V (4...6V). Wtedy układ U1 jest zbędny, a konieczna jest zwora ZW1.

Obwód z tranzystorem T1 to detektor braku napięcia sieci. Gdy zabraknie napięcia sieci, procesor jest zasilany z baterii rezerwowej, a na wejściu P3.1 pojawia się stan wysoki, co powoduje zmianę trybu pracy procesora i zmniejszenie poboru prądu.

Ze względu na niewielką liczbę wyprowadzeń procesora 'C2051, w układzie zastosowano dodatkowo dwa dekodery. Kostka CMOS 4543 jest dekoderem z kodu BCD na kod wskaźnika 7-segmentowego. Układ 4028 jest dekoderem typu 1 z 10 i pomaga zaświecać kolejne cyfry na wyświetlaczu. Dzięki zastosowaniu tych dwóch układów, do sterowania sześciocyfrowym wyświetlaczem wystarczy siedem linii wyjściowych procesora.

Pozostałe linie mogą być wykorzystane w rozmaity sposób. Dwie współpracują z przyciskami umożliwiającymi ustawianie zegara, jedna (P3.0) może sterować dodatkowymi diodami LED, a kolejna (P3.1) pełni bardzo ważną rolę, stanowiąc wejście dla informacji o zaniku napięcia sieci. Punkty J1...J4 dołączone do wolnych wyjść dekodera U5 oraz niewykorzystane linie portu 3 (P3.4, P3.5, P3.7) umożliwiają niemal nieograniczoną rozbudowę układu, choćby podłączenie dodatkowych układów przez łącze I^2C, oraz przekaźników i brzęczyka, niezbędnych przy pracy w roli budzika (co wymaga rozbudowy programu). Punkty L, N można wykorzystać dowolnie. Do punktu L można na przykład dołączyć diody LED oddzielające wyświetlacze godzin i minut, jak pokazuje to rysunek 2, albo inne znaczne obciążenie, choćby przekaźnik.



Rys. 2

W układzie nie przewidziano typowego obwodu resetu z kondensatorem dołączonym do nóżki 1. Zamiast tego włączony jest przycisk pozwalający na zresetowanie układu w dowolnej chwili. Takie rozwiązanie jest tu potrzebne z uwagi na fakt, że napięcie zasilające przy zaniku i powrocie napięcia sieci będzie się znacznie zmieniać, co przy obecności kondensatora mogłoby doprowadzić do niezamierzonego zresetowania zegara po powrocie napięcia sieci.

Aby układ prawidłowo pracował, także w stanie IDLE, potrzebne są rezystory R14, R15. Związane to jest z brakiem wewnętrznych rezystorów podciągających na końcówkach P1.0 i P1.1 procesora, jako że opcjonalnie są to wejścia analogowego komparatora.

Zdekodowane sygnały z układu U3 (4543) podane są na bufor-inwerter U4 typu ULN2803, zaświecający poszczególne segmenty wyświetlaczy. Kostka ULN2803 zawiera osiem jednakowych tranzystorów Darlingtona. Maksymalny prąd wyjściowy wynosi 0,5A, dzięki czemu moduł może sterować nawet wielkimi wyświetlaczami.

Rezystory R7...R13, a także R5, R6, wyznaczają prąd segmentów oraz zmniejszają zależność jasności świecenia wyświetlaczy od zmian napięcia zasilania.

Impulsy zaświecające poszczególne cyfry z wyjść Q1...Q6 dekodera U5 podawane są na układ U6 - mało znaną kostkę CMOS 4504.

Układ scalony 4504 zawiera sześć buforów. Nie są to jednak zwyczajne bufory, ponieważ umożliwiają translację poziomów logicznych. Układ U6 zasilany jest dwoma napięciami dodatnimi o różnej wartości. Na wejścia AI...FI są podawane sygnały o poziomach 0 i 4,7V, bo dekoder U5 jest zasilany napięciem VCC (ok. 4,7V). Tym samym napięciem zasilane są obwody wejściowe kostki U6. Natomiast obwody wyjściowe tego układu (AO...FO) są zasilane napięciem VPP. Napięcie VPP jest wyższe niż napięcie VCC, przynajmniej o spadek napięcia na diodzie D1, lub jeszcze więcej przy wykorzystaniu stabilizatora U1. I właśnie tym wyższym napięciem zasilana jest nie tylko część kostki U6, ale też kolektory tranzystorów T2...T7 (napięcie VPP podane jest także na nóżkę 10 układu U4, czyli na katody diod ochronnych, ale w tym zastosowaniu jest to bez znaczenia). Dzięki takiemu rozwiązaniu wyświetlacze zawsze zasilane są wyższym napięciem, a prąd wyświetlaczy nie płynie przez stabilizator U1.

Jest to bardzo istotne, ponieważ umożliwia zasilanie wyświetlaczy napięciem dochodzącym do 18V. Tak wysokie napięcie zasilania otwiera z kolei drogę do wykorzystania wielkich

Rys. 1

wyświetlaczy, których segmenty z oczywistych względów zawierają kilka świecących struktur LED połączonych w szereg. Przykładowo każdy segment gigantycznego wyświetlacza pokazanego na fotografii okładkowej zawiera cztery grupy po pięć diod połączonych w szereg. A pięć zielonych diod wymaga napięcia zasilania ponad 11V.

Opis programu

Program w postaci źródłowej jest dostępny na stronie internetowej EdW (www.edw.com.pl). Nabywcy zestawu AVT-2632 otrzymają zaprogramowany procesor. Zapoznanie się z programem nie jest więc w żadnym wypadku niezbędne. Ponieważ jednak wielu Czytelników zechce zmodyfikować program zegara, a jest to program dość rozbudowany, warto omówić najważniejsze fragmenty.

Wzorcem czasu jest oczywiście rezonator kwarcowy 12MHz. Jego częstotliwość jest dzielona sprzętowo przez 12, a potem przez 250 w liczniku-timerze T0, który pracując w trybie 2 jest automatycznie przeładowywany. Każdy cykl Timera0 co 250µs generuje przerwanie, które jest obsługiwane przez procesor, niezależnie od wcześniej wykonywanej czynności.

Program obsługi przerwania pokazany jest na **listingu 1**. Każde przerwanie od Timera0 zwiększa zmienną *Co2ms*. Zmienna ta jest w istocie licznikiem do 8 - po każdych ośmiu przerwaniach, czyli co 2ms, ustawiany jest znacznik *Flaga* i zwiększana zmienna *Co4ms*. Zmienna *Co4ms* i zmienna *Co1sek* to kolejne liczniki, dzielące w sumie przez 500.

Listing 1

```
Obsl_timera:            '  co 250us

Incr Co2ms
    If Co2ms = 8 Then
       Co2ms = 0
       Set Flaga  ' obsluga wyświetlacza
'po zliczeniu 8 "przerwań" czyli co 2ms
       Incr Co4ms
       If Co4ms = 2 Then
       Co4ms = 0
       Incr Co1sek       'to wykonywane co
       If Co1sek = 250 Then
          Co1sek = 0
          Incr Jedsek       ' to następuje
          If Jedsek = 10 Then          'powrac
            Jedsek = 0        'wykonuje gdy
            Incr Dziesek
            If Dziesek = 6 Then
               Dziesek = 0
               Incr Jedmin
               If Jedmin = 10 Then
                  Jedmin = 0
                  Incr Dziemin
                  If Dziemin = 6 Then
                     Dziemin = 0
                     Incr Jedgodz
                     If Jedgodz = 10 Then
                        Jedgodz = 0
                        Incr Dziegodz
                     End If
                  End If
               End If
            End If
          End If
       End If
       End If
       If Dziegodz = 2 Then       ' to nie
          If Jedgodz = 4 Then       'podcza
            Jedgodz = 0
            Dziegodz = 0
           ' Incr Dzietyg  ' do wykorzystan
             '    If Dzietyg = 7 Then
             '       Dzietyg = 0
             '    End If
          End If
       End If
    End If

Return
```

Do podziału przez 500 potrzebne są dwie zmienne, bo zwykły, ośmiobitowy licznik nie poradzi sobie z takim zadaniem.

Trwający jedną sekundę cykl licznika *Co1sek* powoduje zliczanie sekund, minut i godzin w kolejnych zmiennych-licznikach. Jak widać, zegar pracuje w trybie 24-godzinnym, a próba skrócenia cyklu do 24 godzin następuje co 2ms. Takie na pierwszy rzut oka dziwne rozwiązanie jest potrzebne, by liczniki prawidłowo zliczały także podczas ustawiania czasu.

Jak się łatwo zorientować, warunkiem poprawnej pracy zegara jest wykonanie tej części programu pomiędzy kolejnymi przerwaniami, czyli w czasie krótszym, niż 250µs, co jest tu zapewnione.

Listing 3

```
Obsluga_wysw:       'następuje co 2ms  po ustawieniu Flagi

Flaga = 0 ' trzeba od razu wyzerować flagę,
'       żeby tylko raz przeprowadzić cykl obslugi wyświetlacza
Incr Mux  'co 2ms zaswiecany jest kolejny wyswietlacz
If Mux = 7 Then Mux = 0       'zlicza do 8,
'    więc cały cykl trwa 2*8=16ms, co daje częstotliwość 62,5Hz
'    przy czestotliwści 62,5Hz nie będzie efektu migania
'  Mux=0 - nie wykorzystany w tej wersji
'        przy rozbudowie można tu np. obslugiwać klawiaturę
'  Mux=1 wyświetlenie dziegodz
'  Mux=2 wyświetlenie jedgodz
'  Mux=3 wyświetlenie dziemin
'  Mux=4 wyświetlenie jedmin
'  Mux=5 wyświetlenie dziesek
'  Mux=6 wyświetlenie jedsek
'  Mux=7 dzietyg - nie wykorzystany w tej wersji

Select Case Mux
   Case 0 : Wysw = 255       'wyświetlacz jest wygaszony
      'tu może być np. obsluga klawiatury
   Case 1 :
         Wysw = Dziegodz
         If Dziegodz = 0 Then Wysw = 15
         ' wygaszanie pierwszego zera
   Case 2 : Wysw = Jedgodz
   Case 3 : Wysw = Dziemin
   Case 4 : Wysw = Jedmin
   Case 5 : Wysw = Dziesek
   Case 6 : Wysw = Jedsek
'  Case 7 : Wysw = Dzietyg 'gdyby byl wyświetlacz dnia tygodnia
End Select
' do zmiennej wysw zostala wpisana liczba z zakresu 0...9,
' która zajmuje co najwyżej cztery mlodsze bity
Clr c      ' to jest potrzebne, by prawidlowo przesunąć
'bity 0...3 na starsze pozycje 4...7, skąd trafią do ukladu 4543
Rotate Wysw , Left , 4       'przesuwanie bitów w lewo o 4 miejsca
Wysw = Wysw Or Mux       ' sumowanie logiczne z Muxem
' Stan Licznika Mux(liczba trzybitowa 0...7) zostaje wpisana do
' trzech mlodszych bitów zmiennej Wysw
P1 = Wysw 'przepisanie do portu i dekoderów oraz wyświetlenie
Return
```

W czasie, gdy nie jest realizowana procedura obsługi przerwania od Timera0, procesor „kręci się w kółko" w nieskończonej pętli DO...LOOP, pokazanej na **listingu 2**, i co 2 milisekundy obsługuje wyświetlacz. Jest to rozwiązanie standardowe w tego typu układach. Podczas takiej działalności procesor pobiera około dziesięciu miliamperów prądu. Przy braku napięcia sieci (stan wysoki na wejściu P3.1), po obsłużeniu przerwania od Timera0 procesor nie pracuje w pętli i nie obsługuje wyświetlacza, tylko go wygasza i przechodzi w stan uśpienia IDLE. Budzi go kolejne przerwanie Timera0, po którym znów „zasypia" i tak dalej. Przy braku napięcia sieci Timer0 stale zalicza i generuje przerwania, które są obsługiwane a czas jest zliczany na bieżąco, przy czym wyświetlacz nie jest obsługiwany, bo wszystkie linie portu P1 są w stanie wysokim (P1=255). Dzięki temu przy braku napięcia sieci cały zegar pobiera z 3-woltowej baterii rezerwowej tylko około 1,3mA, co jest naprawdę bardzo dobrym wynikiem.

Listing 2

```
' glowny program
Do
  If P3.1 = 1 Then       'gdy brak napięcia sieci
     P3.0 = 0       ' wygaszenie ewentulnej diod LED
     P1 = 255       'wygaszenie wyświetlacza
     Idle       'przejście do stanu zamrożenia
  Else    'normalna praca
     If Flaga = 1 Then Gosub Obsluga_wysw
  End If
Loop

End       ' koniec glownego programu
```

Większą oszczędność można byłoby uzyskać stosując tylko zewnętrzny układ scalony zegara RTC, np. z serii PCF85x3.

Należy zauważyć, że główny program „kręci się w kółko" i czeka na ustawienie znacznika *Flaga*. Następuje to co 2ms i właśnie co 2ms wykonywana jest procedura *Obsluga_wysw*. Kluczowe elementy procedury obsługi wyświetlacza pokazane są na **listingu 3**.

Na początek zerowana jest flaga, co gwarantuje, że procedura obsługi wyświetlacza zostanie wykonana tylko raz. Następnie zwiększana jest zawartość zmiennej *Mux*, decydującej o tym, który segment ma zostać wyświetlony. Zwróć uwagę, że cykl obsługi wyświetlacza składa się z siedmiu odcinków czasu po 2ms każdy. Cykl trwa więc 14ms, co daje znaczną częstotliwość odświeżania wyświetlacza powyżej 70Hz, gwarantującą, że nie wystąpi efekt migotania wskaźników. W czasie tych 14ms każdy z sześciu segmentów świeci tylko przez 2ms. Celowo nie skróciłem cyklu do sześciu, tylko do siedmiu stanów (0...6). Gdy *Mux*=0 żaden z wyświetlaczy nie świeci, ale taki stan jest niezbędny dla łatwej realizacji procedur ustawiania, o czym się za chwilę przekonasz. W razie potrzeby stan ten można też wykorzystać na przykład do obsługi dodatkowych przycisków, co umożliwi zastosowanie klawiatury z dużą liczbą klawiszy.

Stany *Mux* od 1...6 są wykorzystane do zaświecania poszczególnych wskaźników (cyfr) wyświetlacza. Niewykorzystany stan *Mux*=7 może posłużyć do wyświetlacza dni tygodnia. Ja zrezygnowałem z tej opcji, bo pokazywanie dnia tygodnia w postaci cyfry moim zdaniem nie ma sensu, a sensowniejsze wykorzystanie siedmiu diod LED wymagałoby rozbudowy układu.

Zależnie od stanu zmiennej *Mux*, do zmiennej *Wysw* zostaje wpisana wartość z odpowiedniego licznika czasu, za co odpowiada instrukcja *Select Case*. Wpisana wartość to liczba z zakresu 0...9, zajmująca cztery młodsze bity. Kolejne instrukcje przesuwają te młodsze bity wewnątrz zmiennej

Wysw o cztery pozycje w lewo. Przy okazji trzeba wyzerować znacznik przeniesienia c, bo BASCOM-owa instrukcja *Rotate* najwidoczniej wykorzystuje asemblerowy rozkaz RLC A (rotate left through carry). Kolejne wersje programu BASCOM, demo i komercyjne, różnią się tu szczegółami. Opisywany program został ostatecznie skompilowany za pomocą wersji demo 2.0.6.0 z roku 2001, gdzie jak widać wykorzystałem instrukcję *Rotate* z wcześniejszym zerowaniem znacznika c(arry), a nie instrukcję *Shift*, która działa różnie w różnych wersjach kompilatora.

Po przesunięciu bitów „w górę", do młodszych czterech, a właściwie trzech bitów zmiennej *Wysw* zostaje za pomocą instrukcji OR dopisana zawartość zmiennej Mux. W ten sposób w zmiennej *Wysw*, a potem na końcówkach portu P1 i na wejściach dekoderów U3, U5 pojawia się jednocześnie informacja, który wskaźnik zaświecić (trzy młodsze bity 0...2), jak i cyfra do wyświetlenia (cztery starsze bity 4...7). Jeden bit zmiennej *Wysw* (bit numer 3) ma zawsze wartość zero i praktycznie nie jest wykorzystany. W razie potrzeby można go dowolnie spożytkować, co umożliwia punkt K na płytce (należy wtedy przeciąć ścieżkę K-K1 i wykonać zworę K1-K2). Połączenie punktu K z K1 umożliwia z kolei wykorzystanie wyjść Q8...Q9 dekodera U5, a przy wykorzystaniu dodatkowego dekodera zwiększenie liczby wyświetlaczy nawet do 15.

Podstawy działania programu są więc proste. Procesor przez cały czas „kręci się w kółko" w pętli głównej. Ta bezproduktywna działalność jest przerywana co 250µs przez przerwanie od Timera0 zwiększające zawartość liczników czasu, oraz co 2ms, gdy zostanie ustawiony znacznik *Flaga*. Nie ma przy tym żadnej sprzeczności interesów. Co bardzo ważne, przerwanie od timera wykonywane jest zawsze, niezależnie od tego, co program akurat robi. Przerwanie od timera zwiększa stany liczników liczących czas i właśnie to jest dla programu zadanie najważniejsze (wykonywane jest także przy braku napięcia sieci). Jeśliby przypadkiem zdarzyło się, że przerwanie od licznika przyjdzie w trakcie wykonywania obsługi wyświetlacza, procesor przerwie tę mniej ważną czynność i obsłuży przerwanie, które decyduje o zliczaniu czasu. Dzięki temu czas zawsze jest liczony poprawnie, a ewentualne zakłócenie (zawieszenie) obsługi wyświetlacza nie ma żadnych złych konsekwencji, najwyżej jedna z cyfr będzie świecić o ułamek milisekundy dłużej, lub wyświetlacz pozostanie wygaszony o ten ułamek milisekundy dłużej. Oczywiście człowiek tego nie zauważy.

Jak wspomniałem, warunkiem poprawnego działania jest to, żeby procedura obsługi przerwania od Timera0 w żadnym przypadku nie trwała dłużej niż 250µs (jej czas trwania nie jest jednakowy, zależy odstanu liczników – o północy zmieniane są stany wszystkich liczników i wtedy trwa ona najdłużej).

Opisane fragmenty programu pokazują ogólną zasadę pracy, natomiast nie dają możliwości ustawiania zegara i nie informują o dodatkowych właściwościach.

W rzeczywistości podprogram *Obsluga_wysw* jest dużo bardziej rozbudowany, co wzbogaciło zegar o kilka dodatkowych funkcji i rozwiązało kilka istotnych problemów.

Uwaga! Wszystkie omawiane dalej procedury zawarte są w podprogramie *Obsluga_wysw*, czyli są wykonywane co 2ms.

Po pierwsze trzeba dodać procedury umożliwiające ustawianie. W tym celu wprowadziłem dodatkową zmienną *Ustawianie*, która ma ścisły związek ze zmienną *Mux*. Zmienna *Ustawianie* to też licznik zliczający od 0 do 6. Przy stanie *Ustawianie*=0 zegar pracuje normalnie, przy stanach 1...6 – migają i mogą być ustawione kolejne cyfry na wyświetlaczu.

O ile *Mux* zmienia zawartość automatycznie co 2ms, o tyle w czasie normalnej pracy zmienna *Ustawianie* ma stale wartość 0. Można to zmienić, naciskając przycisk S1, dołączony do nóżki 6 procesora (P3.2). Naciskanie S1 powoduje zwiększanie zawartości zmiennej *Ustawianie*, a to z kolei powoduje miganie kolejnych cyfr i umożliwia ich ustawienie za pomocą przycisku S2. Ogólna zasada jest znów bardzo prosta: jeśli zmienna *Ustawianie* ma taką samą wartość jak *Mux*, to miga cyfra wyznaczona przez *Mux*. Gdy *Ustawianie*=0, nic nie miga, bo przy *Mux*=0 wyświetlacz nie świeci. Ustawiana cyfra wyświetlacza ma migać w stosunkowo wolnym rytmie (okres rzędu kilkuset milisekund), a tymczasem opisywane procedury są wykonywane co 2ms. Trzeba było wprowadzić kolejną zmienną-licznik *Miganie*, wyznaczającą rytm migania. Jak pokazuje listing 3, co 2ms zwiększa się stan tej zmiennej, a cykl trwa 248ms (124*2ms). Co 248ms zmienia się stan zmiennej bitowej *Wygasz*. Zmienna ta w wersji podstawowej układu steruje pracą dodatkowych diod świecących – patrz rysunek 2.

Listing 4 pokazuje w ogólnym zarysie sposób realizacji migania ustawianej cyfry. W tym uproszczonym listingu pominąłem sprawę zwiększania liczników czasu. Najpierw muszę wyjaśnić sprawę przycisków. W programie z listingu 4 już króciutkie zakłócenie lub drgania styków powodowałyby przypadkowe zliczenie nawet kilku impulsów. Trzeba uodpornić zegar na takie sytuacje. Jest to szczególnie ważne w zegarze-gigancie, gdzie przewody do przycisków ustawiania mogą być długie i podatne na zakłócenia. Ja odkłóciłem styki za pomocą procedur pokazanych na **listingu 5**.

Przeanalizuj dokładnie dwa ostatnie listingi, a jeśli masz wątpliwości co do szczegółów, spróbuj sobie rozrysować na kartce, co dzieje się w poszczególnych cyklach obsługi wyświetlacza, które występują co 2ms. Zwróć uwagę, jak zrealizowałem tu dużo dłuższe czasy (migania i odkłócenia styków). Pamiętaj, że cała procedura obsługi wyświetlacza

Listing 4

```
'to jest część podporogramu Obsluga_wysw

Incr Miganie          ' co 2ms
If Miganie = 124 Then
   Miganie = 0
   Wygasz = Not Wygasz       'szybkość migania prawie 2Hz
'co 248ms zmienia się stan zmiennej bitowej Wygasz
End If

P3.0 = Wygasz        ' zaświeca ew. diodę kontrolną, dolaczoną do P3.0

If P3.2 = 0 Then Incr Ustawianie
'czyli naciśnięcie S1 powoduje przejscie w tryb ustawiania
' w rzeczywistości procedura jest bardziej skomplikowana
'a zmienna Ustawianie może przyjmować wartości 0...7, podobnie jak Mux

Select Case Mux
' tu jest procedura omówiona wcześniej, ale wzbogacona,
' pozwalająca zwiększać stan aktualnie wybranej cyfry za pomocą S2
End Select

' to jest procedura powodujaca miganie ustawianej cyfry
' ustawiana cyfra jest wygaszana w rytm zmian stanu zmiennej Wygasz
' stan Wygasz zmienia się  co 248ms
If Ustawianie = Mux Then
'dotyczy tylko jednej cyfry, aktualnie ustawianej
   If Wygasz = 1 Then Wysw = 15
'przy wartości 15 dekoder 4543 nic nie wyswietli, wygasi wyświetlacz
End If

' tu są dalsze omówione wcześniej procedury
```

Listing 5

```
'to też jest część podporogramu Obsluga_wysw

If P3.2 = 0 Then        ' gdy nacisnięty  przycisk S1
' żeby uniezależnić się od drgań styków i innych "śmieci"
' nie zwiększaj od razu zmiennej Ustawianie,
' tylko odczekaj 60ms
   Incr Przyc1
   If Przyc1 = 30 Then        'reaguje dopiero po 30x2ms naciskania,
' a potem także co 256x2ms czyli co 0,512s jeśli S1 naciskany ciągle
      Incr Ustawianie
      If Ustawianie = 7 Then Ustawianie = 0
   End If
Else
'jeśli były to tylko krótkie zaklócenia
'oraz jeśli S1 zwolniony natychmiast zeruje, zmienną Przyc1,
' żeby nie było kumulacji przy kolejnych naciśnięciach S1
   Przyc1 = 0
End If

'gdy Ustawianie=0 - normalna praca zegara,
' jesli Ustawianie= 1...6 - ustawianie kolejnych liczników
If Ustawianie > 0 Then
'tylko w takim stanie czynny jest przycisk S2
 If P3.3 = 0 Then       'naciśnięcie S2
' naciśniecie S2 nie powoduje natychmiastowej reakcji
' ustawia flagę_zwiększ dopiero po 30x2ms=60ms naciskania
' a potem także co 256x2ms czyli co 0,512s jeśli S2 naciskany ciągle
   Incr Przycisk_plus         ' aby uniezależnić się od drgań
   If Przycisk_plus = 30 Then Flaga_zwieksz = 1
 Else        ' - gdy przycisk S1 zwolniony lub gdy krótkie zaklócenia
    Przycisk_plus = 0
    Flaga_zwieksz = 0
 End If
End If
```

powtarzana jest co 2ms, ale sama trwa o wiele krócej, wiec procesor przez większość czasu „kręci się w kółko” w pętli głównej.

To jeszcze nie wszystko. Przycisk S2 powoduje ustawienie zmiennej bitowej *Flaga_zwieksz* i wtedy powinno być zwiększone wskazanie migającego właśnie wyświetlacza. Wymaga to dodatkowych zabiegów, a podana wcześniej procedura wyświetlania musi być znacznie rozbudowana. W rzeczywistości wygląda ona jak na **listingu 6**, a nie jak na listingu 3.

Listing 6

```
If Ustawianie = Mux Then
   Aktywna_pozycja = 1        ' dodatkowa zmienna bitowa,
   If Flaga_zwieksz = 1 Then Zwieksz = 1
   Flaga_zwieksz = 0
End If

Select Case Mux
   Case 0 : Wysw = 255        'tu może być np. obsluga klawiatury
   Case 1 :
         If Ustawianie = 0 Then        'podczas normalnej pracy zegara
             Wysw = Dziegodz
             If Dziegodz = 0 Then Wysw = 15  'wygaszanie pierwszego zera
         Else        'podczas ustawiania
           If Zwieksz = 1 Then Incr Dziegodz
           If Dziegodz = 3 Then Dziegodz = 0
                ' pozwala ustawić tylko 0, 1, 2
           If Jedgodz > 3 Then        ' nie można ustawić godziny 24...30
              If Dziegodz = 2 Then Jedgodz = 0
           End If
           Wysw = Dziegodz
         End If
   Case 2 : Wysw = Jedgodz
          If Zwieksz = 1 Then Incr Jedgodz
          If Jedgodz = 10 Then Jedgodz = 0
          If Dziegodz = 2 And Jedgodz > 3 Then Jedgodz = 0
   Case 3 : Wysw = Dziemin
          If Zwieksz = 1 Then Incr Dziemin
          If Dziemin = 6 Then Dziemin = 0
   Case 4 : Wysw = Jedmin
          If Zwieksz = 1 Then Incr Jedmin
          If Jedmin = 10 Then Jedmin = 0
   Case 5 : Wysw = Dziesek
          If Zwieksz = 1 Then Gosub Zerowanie_sekund
'     wprostszym przypadku wystarczyloby:
'           Colsek = 0
'           Jedsek = 0
'           Dziesek = 0
' ale wtedy jesli zegar się trochę późni, trzeba zwiększać minuty
   Case 6 : Wysw = Jedsek
          If Zwieksz = 1 Then Gosub Zerowanie_sekund        'patrz wyżej
End Select

If Aktywna_pozycja = 1 Then        'miganie ustawianej cyfry
   Aktywna_pozycja = 0
   If Wygasz = 1 Then Wysw = 15        '4543 nic nie wyswietli
End If

Zwieksz = 0
' tu dalej przesuwanie logiczne i wyświetlanie
```

Zmienna bitowa *Aktywna_pozycja* została wprowadzona, by nie badać powtórnie warunku *Ustawianie=Mux* po instrukcji *Select Case*, gdy realizowane jest miganie, stosownie do zawartości zmiennej *Wygasz*.

Zwróć uwagę, że zmianę zawartości wybranego wyświetlacza powoduje zmienna bitowa *Zwieksz*, a nie *Flaga_zwieksz*. Czy rozumiesz, dlaczego tak jest?

Zmienna bitowa *Flaga_zwieksz* zostaje ustawiona w jakimś dowolnym momencie, po odpowiednio długim naciskaniu S2 i pozostaje ustawiona, ale nieaktywna aż do czasu, gdy *Ustawianie=Mux*. Dopiero wtedy zmienna *Zwieksz* zostanie ustawiona, w ramach instrukcji *Select Case* zwiększy stan odpowiedniego licznika i potem zostanie wyzerowana.

Czy tej roli nie mogłaby pełnić po prostu *Flaga_zwieksz*? Co należałoby wtedy zmienić?

Zastanów się jeszcze, dlaczego przy ustawianiu trzeba skracać cykl liczników czasu? Czy nie zapewni tego omówiona na początku procedura obsługi przerwania od Timera0, powtarzana co 250µs? Dlaczego nie zapewni?

Uznałem, że pierwsza cyfra, dziesiątki godzin powinna być wygaszana, jeśli wyświetlana godzina jest liczbą jednocyfrową. Dzięki jednej linijce programu zegar wyświetli nie 01, tylko 1; nie 08, tylko 8. Ale podczas ustawiania powinny być widoczne wszystkie cyfry, stąd użycie klauzuli *Else*, gdy *Mux*=1.

Podczas ustawiania występuje dodatkowy kłopot. Choć zegar normalnie zlicza czas, poszczególne licznik, ustawiane są pojedynczo. Można sobie wyobrazić sytuację, że ktoś ustawi dziesiątki godzin na 2 i jednostki godzin na 5 czy więcej. Ponieważ nie ma godziny 25, należy uzależnić ustawianie jednostek godzin od stanu dziesiątek godzin. Stąd kolejne linie kodu.

W programie ostatecznie wykorzystałem rozbudowaną procedurę zerowania sekund. Chodziło o to, by zerowanie sekund, gdy zegar nieco się późni, spowodowało dodatkowo zwiększenie licznika minut i ewentualnie godzin. To bardzo ważne w praktyce, bo zwykle koryguje się zegar o pełnej godzinie na podstawie sygnału z radioodbiornika. Jeśli np. zegar pokazuje 14:59:46, naciskamy zerowanie sekund i... zamiast 14:59:00 powinno być 15:00:00. Zapewnia to procedura *Zerowanie_sekund*, pokazana na **listingu 7**.

Listing 7

```
 Zerowanie_sekund:

 If Dziesek > 2 Then Incr Jedmin
 ' jeżeli do następnej pełnej minuty  brakuje
 ' mniej niż 30 sekund, zwiększ licznik minut
             Colsek = 0
             Jedsek = 0
             Dziesek = 0
 ' ewentualnie zwiększa kolejne liczniki,
 ' żeby zegar "nie zgubił" się przy ustawianiu
 If Jedmin = 10 Then
    Jedmin = 0
    Incr Dziemin
    If Dziemin = 6 Then
       Dziemin = 0
       Incr Jedgodz
       If Jedgodz = 10 Then
          Jedgodz = 0
          Incr Dziegodz
       End If
    End If
 End If
 Return
```

Listing 8

```
Mux1 = Mux Xor &B11111111       'nie udalo sie z rozkazem Not, stąd XOR
'operacja "odwrócenia" bitów Mux jest potrzebna, bo w wersji podstawowej
' z wyświetlaczami 20mm i 14mm adresy są ściśle określone
' np. adres 1 jest przypisany jednostkom sekund i trzeba to odwrócić
Mux1 = Mux1 And &B00000111        ' wyższe bity muszą być zerami

Clr c
Rotate Wysw , Left , 4       'przesuwanie Wysw do bitów 4...7
Wysw = Wysw Or Mux1       ' sumowanie logiczne z "odwóconym" Muxem
Wysw = Wysw And &B11110111
P1 = Wysw 'przepisanie do portu
```

Na koniec muszę Ci się przyznać do istotnej zmiany, jaką wprowadziłem po testach pierwszego modelu. Jak wynika z rysunków 1 i 2, podanie na wejścia dekodera U5 liczby 1 powoduje zaświecenie jednostek sekund. Liczba 6 powoduje zaświecenie dziesiątek godzin. Nie można tego zmienić przy zastosowaniu płytki małego wyświetlacza, pokazanego na fotografii tytułowej. Czy zwróciłeś uwagę na tę niekonsekwencję przy analizie listingu 3? Jeśli tak, szczerze gratuluję!

W pierwszej wersji jedno naciśnięcie przycisku S1 powodowało miganie jednostek sekund, a szóste – dziesiątek godzin. Zegar trzeba było ustawiać, począwszy od sekund, potem minuty i na koniec godziny. Mnie jako twórcy programu, taka kolejność wydawała się naturalna.

Jak już wspomniałem, dałem zegar do testowania (do ustawienia) kilku osobom. Choć obsługa nie okazała się problemem, większość z nich najpierw chciała ustawiać godziny, potem minuty i sekundy. W związku z takim wynikiem testu zdecydowałem się zmienić kolejność ustawiania na bardziej intuicyjny. W związku ze „sztywnym” przyporządkowaniem cyfr do wyjść A1...A6 płytki, musiałem niejako „odwrócić” stan zmiennej *Mux*. Wprowadziłem zmienną pomocniczą *Mux1* i dwie dodatkowe linie kodu, pokazane na **listingu 8**. Przy okazji okazało się, że używana wersja kompilatora nie radzi sobie z zanegowaniem bajtu z pomocą operatora logicznego NOT, stąd negowanie za pomocą XOR.

Przy zastosowaniu gigantycznych wyświetlaczy zamiast takiej operacji można po prostu zmienić kolejność przewodów sterujących anody (A1...A6).

Na stronie internetowej EdW oprócz ostatecznej wersji programu (*GigantNew.BAS*), znajdziesz też zmodyfikowane listingi pokazane w artykule oraz pierwotną wersję programu z ustawianiem „od końca” (*GigantOld.BAS*).

Montaż i uruchomienie

Sterownik można zmontować na płytce drukowanej, pokazanej na **rysunku 3**. Montaż sterownika jest klasyczny i nie powinien sprawić trudności nawet mniej zaawansowanym. W pierwszej kolejności należy wykonać zwory, zaznaczone na płytce kółeczkami i liniami. Jest ich sporo, ale nie bez przyczyny płytka drukowana jest jednostronna – radykalnie obniża to jej cenę, co na pewno i dla Ciebie jest istotne.

W wersji podstawowej punkty K, K1, oraz M, M1, są zwarte odcinkami ścieżek i nie ma potrzeby ingerencji w te obwody.

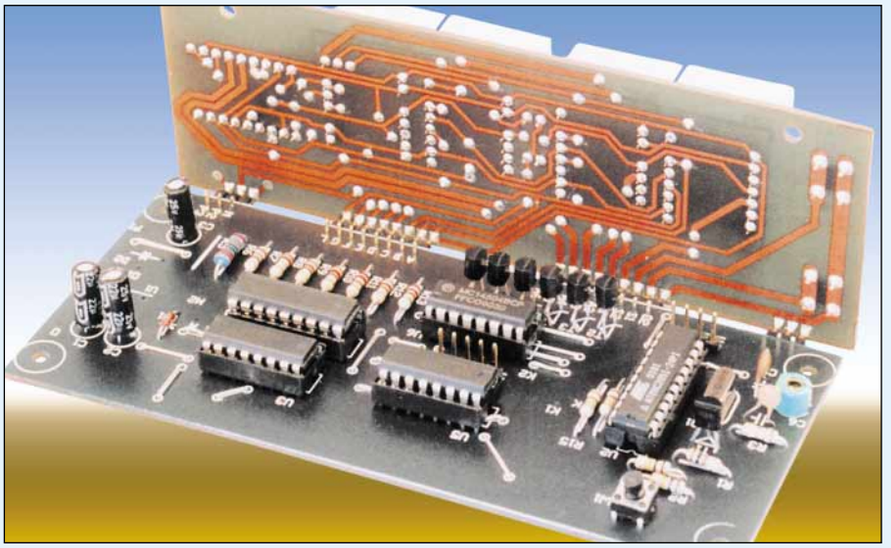
W zestawie AVT-2632 dostarczony jest zaprogramowany procesor, więc układ bezbłędnie zmontowany ze sprawnych elementów od razu będzie pracować poprawnie.

W roli baterii rezerwowej można zastosować jedno ogniwo litowe albo 2...3 ogniwa 1,5V.

Wykonanie i dołączenie wyświetlacza na płytce według **rysunku 4** nie będzie problemem. Najwięcej czasu pochłonie wlutowanie w tę jednostronną płytkę wszystkich zaznaczonych zwór. Wyświetlacze warto umieścić w podstawkach. Przy takich niewielkich wyświetlaczach zegar najprościej będzie zasilić zapięciem 5V z zewnętrznego stabilizowanego zasilacza. Napięcie to należy podać na punkty P, O, a w sterowniku nie montować stabilizatora U1, tylko wlutować zworę ZW1. W modelu pokazanym na fotografii wstępnej rezystory R7...R13 mają po 22Ω, natomiast R6 - 220Ω. Przy napięciu 5V układ pobiera nie więcej niż 140mA prądu, a jasność wskaźników jest absolutnie wystarczająca. Układ można umieścić w obudowie, np. KM-60, jednak godna rozważenia jest jeszcze inna wersja. Kilka osób pytanych w tej kwestii stwierdziło, iż tak ładnego układu... nie trzeba umieszczać w obudowie, ewentualnie zastosować obudowę przezroczystą. Jeden z pytanych stwierdził, że chętnie umieści taki zegar w swoim pokoju w regale, gdzie za szkłem nie będzie się kurzył, a zaprezentuje swe walory w całej okazałości.

Przy zastosowaniu gigantycznych wyświetlaczy obudowę i sposób mocowania trzeba dobrać we własnym zakresie. Płytkę sterownika należy podłączyć do wyświetlacza za pomocą przewodów, podobnie przewodami należy wykonać połączenia między segmentami wyświetlaczy.

Ciąg dalszy na stronie 27.



Wykaz elementów

Sterownik AVT-2632/1

R1	10kΩ
R2,R4,R14,R15	100kΩ
R3	20kΩ
R6	220Ω
R7-R13	22Ω (* patrz tekst)
C1	22µF/25V
C2 C3	22µF/10V
C4	33pF
C5	10pF
C6	trymer 10...40pF
D1,D2	dioda Schottky'ego, np. BAT85
T1	BC548
T2-T7	BC517
U1	LM7805
U2	AT89C2051
U3	CMOS 4543
U4	ULN2803
U5	CMOS 4028
U6	CMOS 4504
SW1	uswitch
Q1	kwarc 12MHz

Wyświetlacz – AVT-2632/2

DSP1-DSP4	SA08-11EWA
DSP5	DA56-11EWA
D3,D4	LED 3mm czerw.
S1,S2	uswitch

goldpiny kątowe – 24szt

Uwaga! W skład kitu AVT-2632/A wchodzi tylko płyta główna zegara, a zestawu AVT-2632/B płytka główna i komplet elementów, ale bez wyświetlacza. Wyświetlacz należy zamówić oddzielnie. Dostępne są płytki wyświetlacza według rysunku 4 (AVT-2632/2) oraz zestawy opisane w artykule Gigantyczne wyświetlacze LED w tym numerze EdW.

W skład kitu AVT-2632/A wchodzi płytka główna zegara wraz z zaprogramowanym procesorem, a komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2632.

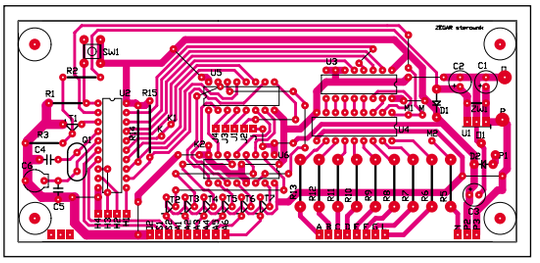
Ciąg dalszy na stronie 27.

Wtedy stabilizator U1 jest konieczny, a napięcie zasilające, podawane na punkty P, O (niekoniecznie stabilizowane) nie powinno być niższe niż 6,5V. Na płytce sterownika zamiast rezystorów R7...R13 należy wlutować zwory, natomiast rezystory ograniczające przewidziane są na płytkach wyświetlaczy (patrz artykuł *Gigantyczne wyświetlacze LED*). Ich wartość trzeba dobrać indywidualnie, zależnie od parametrów wyświetlaczy i wartości napięcia zasilania. W dużym modelu, zasilanym napięciem niestabilizowanym około 15V, przy pięciu zielonych diodach LED połączonych w szereg rezystory w wyświetlaczach mają po 15Ω. Pobór prądu nie przekracza 0,55A.

Uwaga! Ponieważ w układzie nie ma kondensatora w obwodzie resetu, zawsze po włączeniu zasilania należy nacisnąć przycisk SW1 umieszczony obok procesora.

Po pewnym czasie użytkowania można skorygować częstotliwość oscylatora kwarcowego za pomocą trymera C6. Gdyby nawet przy minimalnej pojemności C6 zegar nadal się późnił, można wylutować C5. Przy starannej korekcji można uzyskać dokładność zegara rzędu kilku sekund na miesiąc.

Piotr Górecki



Rys. 3 Skala 1:2

Rys. 4 Skala 1:2