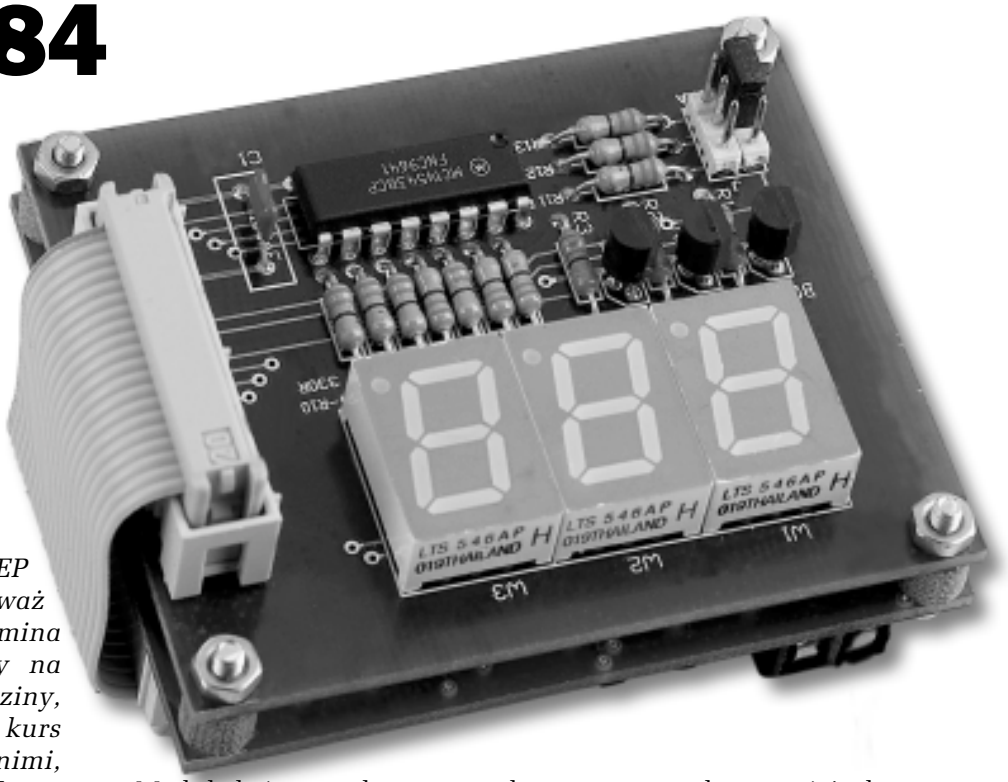
Moduł mikrokontrolera PIC16C84

AVT-873



Dawno na łamach EP nie PICowaliśmy. Ponieważ wielu Czytelników upomina się o projekty na mikrokontrolerach tej rodziny, a także o krótki kurs posługiwania się nimi, opracowaliśmy moduł, który można wykorzystać zarówno do eksperymentów, jak i do realizacji praktycznych sterowników. Zaczynamy od prezentacji modułu. Kurs PICowania od następnego numeru!

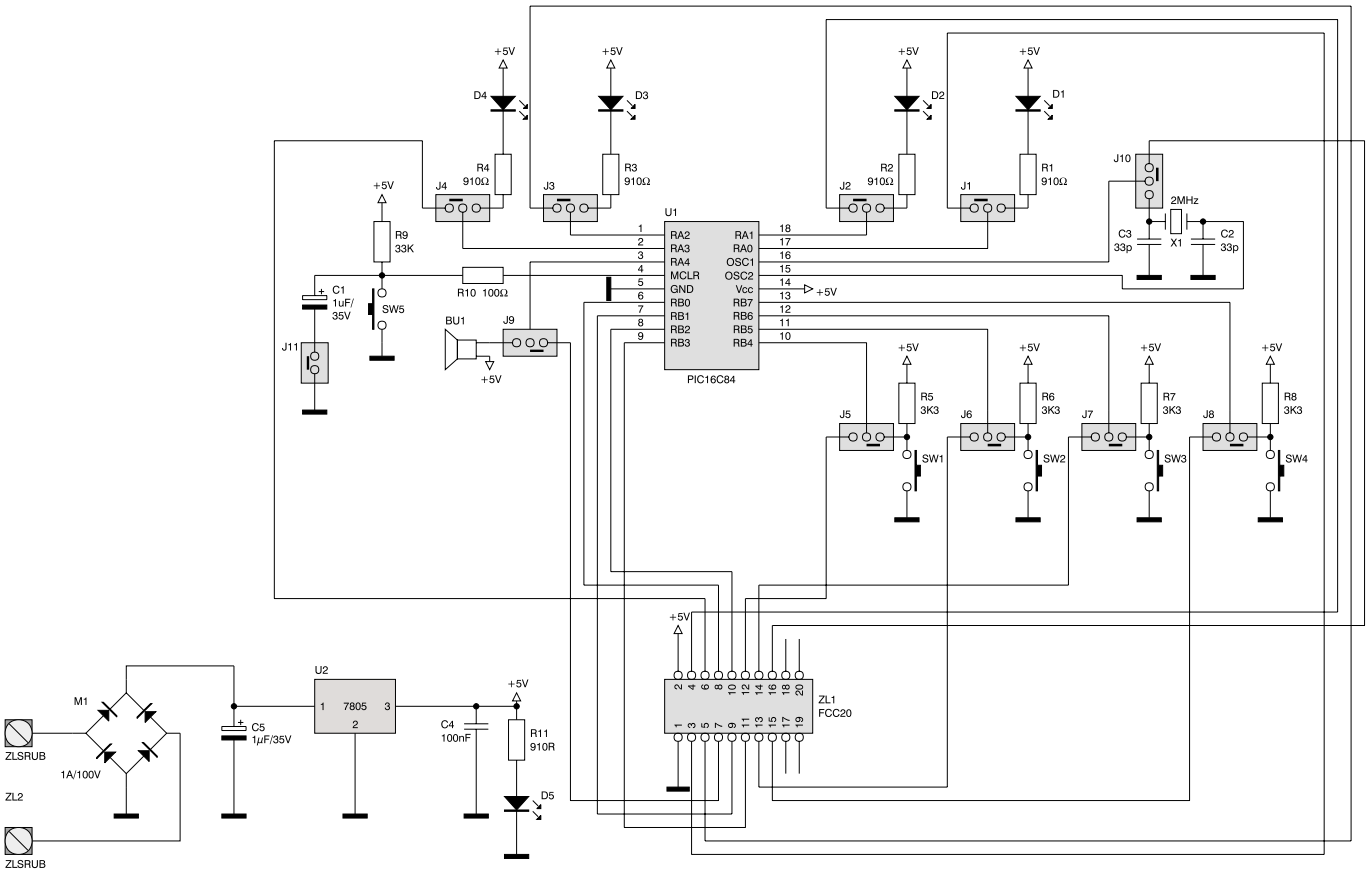
Moduł, którego schemat przedstawiamy na **rys. 1**, pozwala zarówno na przeprowadzanie prostych eksperymentów, jak i wykorzystanie go jako sterownika w bardziej zaawansowanych konstrukcjach.

Wszystkie linie portu PORTA oraz linie RB4..RB7 portu PORTB doprowadzone są do przełączników oznaczonych J1..J9. Przełącznik taki to nic innego jak trzy pozłacane piny zwierane zworkami. Rozwiązanie takie jest doskonale znane chociażby z ustawiania częstotliwości zegara w płytach głównych komputerów. Ustawienie zworek tak jak na rysunku powoduje połączenie linii portów ze złączem ZL1 (20-pinowe złącze typu IDC do zaciskania na kablach wstążkowych). Do tego złącza podłączone jest również zasilanie oraz ewentualnie pin OSC1 mikrokontrolera (J10).

Poprzez przestawienie zworek można dołączyć do linii RA0..RA3 diody świecące LED D1..D4. Rezystory R1..R4 ograniczają prąd tych diod. Linie portów mikrokontrolera można bez obawy obciążyć prądem do 20mA. Możliwe jest więc bezpośrednie sterowanie LED-ów z linii portów. Przy zastosowanych wartościach rezystorów ograniczających dobrze świecą diody czerwone o podwyższonej jasności. Nieco dalej pokazany zostanie prosty program umożliwiający zaświecenie jednej lub kilku diod. Do linii RA4 podłączony jest mały brzęczyk. Ponieważ RA4 ma wyjście typu otwarty dren, to druga końcówka brzęczyka musi być podłączona do plusa zasilania. Wystawienie na RA4 stanu niskiego powoduje podanie zasilania i zadziałanie brzęczyka.

Do linii RB4..RB7 można dołączyć cztery klawisze SW1..SW4. Przyciśnięcie klawisza powoduje pojawienie się na linii portu stanu niskiego. Rezystory R5..R8 wymuszają stan wysoki na liniach w momencie, kiedy klawisze nie są przyciśnięte. Zrezygnowano tutaj z podciągania do plusa zasilania w strukturze portu poprzez zerowanie bitu RBPU w OPTION_REGISTER.

Dołączenie klawiszy właśnie do linii RB4..RB7 nie jest przypadkowe. Wiadomo, że zmiany stanów na tych liniach mogą generować przerwanie. Można wykorzystać to w programie obsługi klawiatury wykorzystującym właśnie to przerwanie. Pozostałe linie



Rys. 1. Schemat elektryczny modułu mikrokontrolera.

portu PORTB połączone są bezpośrednio do złącza ZL1.

W skład obwodu oscylatora wchodzą elementy C2, C3 i X1. Dla częstotliwości 2MHz bity konfiguracyjne FOSC1 i FOSC0 powinny być ustawione na tryb XT. Istnieje też możliwość taktowania mikrokontrolera zewnętrznym sygnałem zegarowym. W takim przypadku należy ustawić zworkę w J10 tak jak pokazano na rys. 1. Nie należy lutować wtedy elementów obwodu oscylatora.

Zewnętrzny obwód zerowania składa się z elementów R9, R10, C1, SW5, oraz J11. Jeżeli zwora J11 jest zwarta, to po włączeniu zasilania na wejściu MCLR występuje stan 0 o czasie trwania zależnym od wartości elementów R9, C1. Takie rozwiązanie daje pewność poprawnego zerowania niezależnie od szybkości narastania napięcia zasilającego. Rozwarcie J11 powoduje połączenie MCLR z plusem zasilania przez szeregowo połączone rezystory R9 i R10. Daje to możliwość sprawdzenia poprawności generowania wewnętrznego impulsu zerującego po włączeniu zasilania (POR). Ręczne zerowanie jest możliwe po przyciśnięciu przycisku SW5.

Napięcie stałe zasilające moduł powinno być odfiltrowane i mieć wartość 8..12V.

Jak widać z tego krótkiego opisu, budowa modułu nie jest skomplikowana. Daje jednak kilka wariantów wykorzystania linii portów, taktowania i zerowania mikrokontrolera. Rozmieszczenie elementów pokazane jest na **rys. 2**.

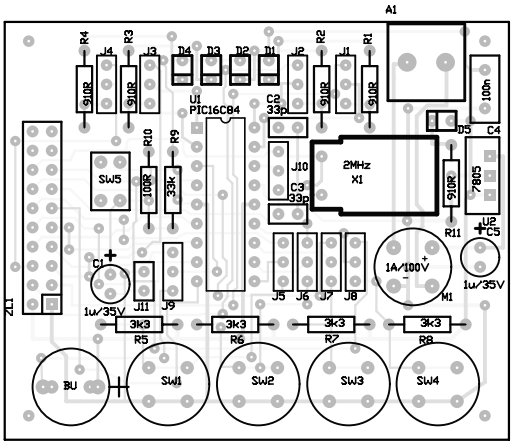
Program przykładowy - test i uruchomienie

Teraz mamy wszystko co potrzebne, aby zacząć próby. We wszystkich programach będziemy używać rejestrów obszaru SFR. Wygodnie jest posługiwać się ich symbolicznymi nazwami. Dobrze jest więc na samym początku stworzyć sobie zbiór definiujący nazwy rejestrów oraz bitów w tych rejestrach. Może on wyglądać np. tak:

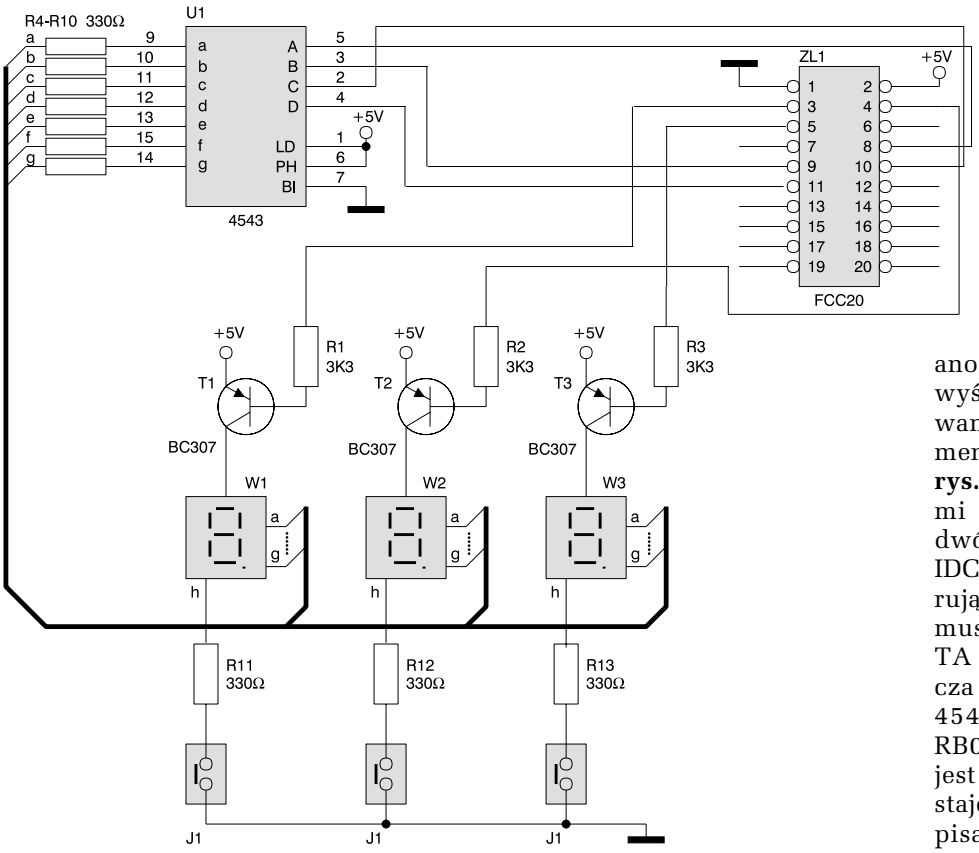
```
;definicja rejestrow SFR
INDF            equ 0
TMR0            equ 1
PCL             equ 2
  .........

;definicja zmiennych bitowych
;rejestr STATUS
#define C       0
#define DC      1
#define Z       2
  ..........
```

Zbiór ten nazwiemy np. 16c84.h i będziemy dołączać na początku pliku źródłowego za pomocą *include <16c84.h>*. Można wtedy bez obawy napisać *bcf STATUS,RP0*, a kompilator



Rys. 2. Rozmieszczenie elementów na płytce drukowanej modułu mikrokontrolera.



Rys. 3. Schemat elektryczny układu wyświetlacza dynamicznego.

mpasm przyjmie to ze stoickim spokojem i nie wygeneruje przykrego komunikatu o błędach.

Spróbujmy teraz napisać przykładowy program, który będzie zapalał diodę D1 (pozostałe będą zgaszone). Musimy zdefiniować linie RA0..RA3 jako wyjściowe. Linia RA0 powinna być w stanie zero, a pozostałe w stanie jeden.

Proponowany przeze mnie program pokazano na **list. 1**.

Trzeba teraz zaprogramować mikrokontroler, włożyć w podstawkę i włączyć zasilanie. Jeżeli zapali się dioda D1, to mamy za sobą pierwszy mały sukces. Prawidłowo działa kompilator asemblera, programator, układ zasilania, obwód oscylatora, obwód resetu i sam mikrokontroler.

Spróbujmy teraz wykorzystać nasz moduł do budowy trochę bardziej skomplikowanego urządzenia - timera kuchennego. Odmierzany czas będzie ustawiany w zakresie od 1 do 60 min. Ustawianie odbywa się za pomocą dwóch klawiszy: *plus* i *minus*. Naciskanie klawisza *plus* powoduje zwiększanie liczby odliczanych minut aż do wartości 60. Dalsze przyciskanie tego klawisza nie powoduje żadnego działania. Naciskanie klawisza *minus* powoduje zmniejszanie liczby minut aż do wartości 1. I teraz też dalsze przyciskanie tego klawisza nie zmienia ustawień liczby odliczanych minut. Po ustawieniu licznika minut trzeba nacisnąć klawisz start. Następuje wtedy wpisanie nastawionej wartości do pamięci EEPROM i uruchomienie procesu odliczania minut. Po włączeniu zasilania zapisana w pamięci EEPROM wartość jest przepisywana do licznika minut. Jest to bardzo wygodne, ponieważ nie trzeba po każdym włączeniu urządzenia ustawiać na nowo odliczanej wartości. Oczywiście dotyczy to przypadku, kiedy często odliczamy ten sam czas, np. przy gotowaniu jajek.

Rolę klawiszy pełnią SW1..SW4. SW4 to *start*, SW3 - *minus*, a SW2 - *plus*. Zworki J5..J8 trzeba ustawić tak, by dołączyć klawisze do linii RB4..RB7. Także zworką J9 dołączamy BU1 do linii RA4. J11 musi być zwarta, a J10 łączy X1 i C3 z OSC1. Zworka J4 powinna łączyć linię RA3 z diodą LED D4. Będzie ona zapalać się i gasnąć co 1s. w trakcie odliczania czasu.

Pozostaje układ wyświetlania czasu. Do tego celu został zaprojektowany układ wyświetlacza (schemat na **rys. 3**). Układ scalony U1 to konwerter kodu BCD na kod 7-segmentowego wyświetlacza LED (wspólna anoda). Tranzystory T1..T3 załączają prąd anod wyświetlaczy. Dla układu wyświetlacza została zaprojektowana płytka - rozmieszczenie elementów przedstawione jest na **rys. 4**. Połączenie między płytkami wykonane jest za pomocą dwóch złącz 20-pinowych typu IDC. Anodami wyświetlaczy sterują linie RA0..RA2. Zworki J1..J3 muszą łączyć te linie portu PORTA z odpowiednimi pinami złącza ZL1. Wejścia A..D układu 4543 połączone są z liniami RB0..RB3 portu PORTB. Teraz już jest wszystko gotowe i nie pozostaje nic innego jak rozpocząć pisanie programu.

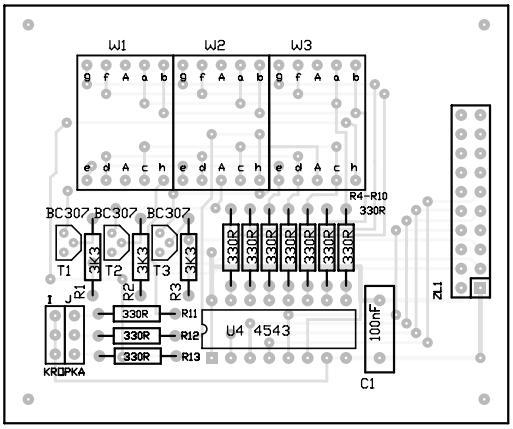
Podstawową funkcją urządzenia jest odmierzanie czasu. Będzie to robił licznik TMR0. Źródłem zliczanych impulsów jest przebieg o częstotliwości *fxtal*/4. Częstotliwość ta podzielona będzie wstępnie za pomocą preskalera przez 32. Do TMR0 wpisujemy taką wartość, aby nastąpiło przepełnienie licznika po upływie 5ms. Przepełnienie to generuje przerwanie (T0IE=1 w INTCON). W obsłudze przerwania ładowany jest ponownie licznik TMR0 oraz wywoływane są procedury *_timer*, *_czas* i *_wyświetlacz* (źródłowa wersja programu znajduje się na płycie CD-EP6/2000 oraz w Internecie pod adresem www.ep.com.pl/programy/programy.html). Procedura *_timer* jest wykonywana,

List. 1.

```
    processor 16c84
    include <16c84.h>
    org 0000
    goto start

    org 0010
start
    bsf STATUS,RP0   ;bank 1 w nim znajduje
            ;się rejestr konfiguracji TRISA
    movlw b 00010000 ;linia RA4 wejściowa,
            ;RA0-RA3 wyjściowe
    movwf TRISA      ;przepisanie z W do TRISA
    bcf STATUS,RP0   ;bank 0 w nim znajduje
            ;się rejestr portu PORTA
    movlw b 00011110 ;bit 0 ma wartość 0
            ;pozostałe jedynki
    movwf PORTA      ;wpisanie do portu

et  nop              ;pętla nieskończona
    goto et
```



Rys. 4. Rozmieszczenie elementów na płytce drukowanej modułu wyświetlacza.

kiedy bit *start* ma wartość 1. Modyfikowane są wtedy liczniki milisekund (ms), sekund (s) oraz minut (min). Przy każdej zmianie zawartości licznika minut wywoływana jest procedura *_wysczas*. Licznik minut przepisuje się wówczas do zmiennej *lhex* i następuje zamiana binarnej wartości tej zmiennej na liczbę dziesiątek (zmienna *dzies*) i liczbę jednostek (zmienna *jedn*). Zamiany tej dokonuje procedura *_konw*. Zawartość zmiennych *dzies* i *jedn* jest następnie wpisywana do bufora wyświetlacza. W momencie, kiedy licznik minut osiągnie wartość zerową, zeruje się bit start i odliczanie jest zatrzymywane.

Przy każdej zmianie licznika sekund wywoływana jest procedura *_migled*. Powoduje ona cykliczne zapalanie i gaszenie co jedną sekundę diody D4. Jest to sygnalizacja uruchomienia odliczania czasu.

Procedura *_wyswietlacz* obsługuje dynamiczne wyświetlanie trzech cyfr. Na linie RB0..RB3 wystawiana jest liczba w kodzie BCD. Na anodę wyświetlacza np. W1 podawane jest napięcie +5V w momencie, kiedy na linii RA0 jest stan niski. Ponieważ PORTA oprócz sterowania anodami wyświetlacza steruje miganiem diody D4 i brzęczykiem BU1, wszystkie operacje dotyczące tego portu są dokonywane na zmiennej *cporta*. Procedura *_wyswietlacz* przepisuje zawartość tej zmiennej do rejestru PORTA co 5ms. Wystarczy zatem np. odpowiednio ustawić bit odpowiadający za sterowanie BU1 w zmiennej *cporta*, a w obsłudze wyświetlacza zostanie on wpisany do PORTA.

Ostatnią procedurą wywoływaną w obsłudze przerwania jest *_czas*. Służy ona do odliczania opóźnień potrzebnych do obsługi klawiatury. Jeżeli bit startu ma wartość 1, to jest dekrementowana wartość licznika *msekczas*. Kiedy *msekczas* osiągnie wartość 0, to automatycznie zeruje sie bit *msekczas* i odliczanie opóźnienia jest zatrzymywane.

Zobaczmy teraz, jak nasz program będzie się zachowywał po włączeniu zasilania lub wyzerowaniu mikrokontrolera. Pierwszym rozkazem jest skok do etykiety *_inic*. Następuje tam zaprogramowanie portów. Wszystkie linie PORTA są ustawione jako wyjściowe. W PORTB linie RB0..RB3 to linie wyjściowe, natomiast RB4..RB7 to linie wejściowe. Po zaprogramowaniu portów inicjowane są liczniki milisekund, sekund i minut używane przez procedurę *_timer*, ładowany jest licznik TMR0 oraz odblokowywane jest przerwanie od przepełnienia licznika i cały system przerwań (bit GIE). Po tych czynnościach zeruje się bit *start* (zatrzymanie zliczania) i wywoływana jest procedura odczytu komórki pamięci EEPROM o adresie 00hex mikrokontrolera (*_rdeeprom*). Zapisana tam wartość licznika minut wpisywana jest do zmiennej *min* i wyświetlana (*_wysczas*).

Od etykiety *_petlagl* program wchodzi w pętlę główną. Procedura *_klawiatura* obsługuje klawiaturę. Po przyciśnięciu klawisza następuje wyjście z procedury, a kod klawisza umieszczany jest w zmiennej *pom* i rejestrze W. Działanie klawiszy *plus* i *minus* zostało już opisane wyżej. Naciśnięcie klawisza start powoduje skok do etykiety *_start*. Bit *start* jest ustawiany na 1 i timer zaczyna odliczać czas. Procedura *_wreeprom* zapisuje zawartość *min* do pamięci EEPROM pod adres 00hex. Następnie w pętli sprawdzany jest warunek skończenia odliczania - skok do etykiety *estart1* oraz warunek naciśnięcia dowolnego klawisza - skok na początek do etykiety *_pocz*. Po zakończonym odliczaniu uruchamiana jest procedura *_pipi*, która włącza i wyłącza cyklicznie co 250ms BU1. Przyciśnięcie dowolnego klawisza kończy sygnalizację dźwiękową i następuje powrót do początku.

WYKAZ ELEMENTÓW

Płytka sterownika

Rezystory
R1..R4, R11: 910Ω
R5..R8: 3,3kΩ
R9: 33kΩ
R10: 100Ω
Kondensatory
C1, C5: 1µF/35V
C2, C3: 33pF
C4: 100nF
Półprzewodniki
D1..D4: LED 3mm czerwone
D5: LED 3mm zielona
M1: mostek 1A/150V
U1: PIC16C84-04/P
U2: 7805
Różne
BU1: brzęczyk piezoelektryczny
J1..J11: listwa goldpin plus zworki
SW1..SW4: przyciski
SW5: przycisk reset
X1: rezonator kwarcowy 2MHz
ZI1: złącze IDC 20pin
ZI2: złącze śrubowe podwójne do druku
Podstawka DIL18

Płytka wyświetlacza

Rezystory
R1..R3: 3,3kΩ
R4..R13: 330Ω
Półprzewodniki
T1..T3: BC307
U1: 4543
W1..W3: wyświetlacz WA
Różne
ZI1: złącze IDC 20pin
J1: listwa goldpin plus zworka

Tomasz Jabłoński, AVT
tomasz.jablonski@ep.com.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/pcb.html*** *oraz na płycie CD-EP06/2000 w katalogu* ***PCB****.*

Źródłowa wersja programu znajduje się na płycie CD-EP6/2000 oraz w Internecie pod adresem www.ep.com.pl/programy/programy.html.