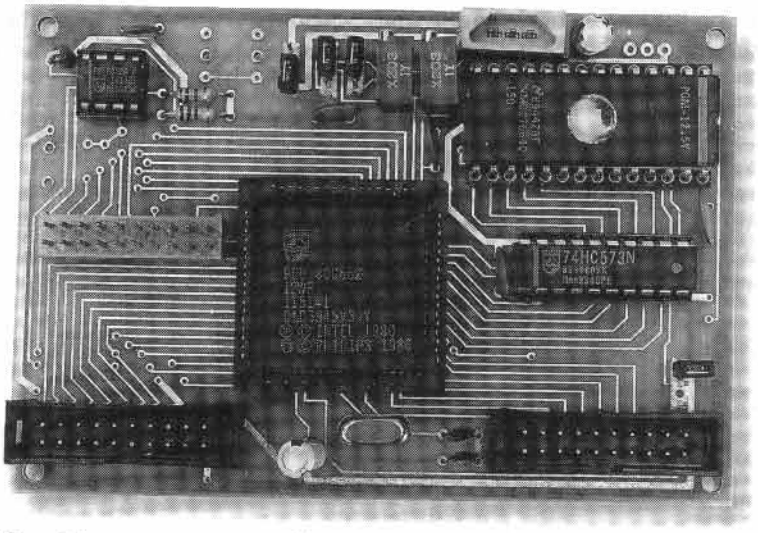
Płytka bazowa mikrokontrolera 80C552, część 2

kit AVT-280

W drugiej części artykułu opisującego płytkę bazową skupimy się na przedstawieniu kilku ciekawych, a przy tym prostych w wykonaniu, aplikacji. Z ich wykonaniem poradzą sobie nawet mało wprawni konstruktorzy, a zamieszczone w artykule listingi ułatwią analizę sposobu działania przykładowych programów.



Możliwość sprzętowej reakcji na programowo ustawiany warunek jest cenną zaletą procesora 80552. Umożliwia np. precyzyjne generowanie impulsów i tworzenie zależności czasowych między zdarzeniami, bez angażowania w te procesy programu, co jest bardzo ważne w układach automatyki. Przykładem będą dwa opisane później programy wykorzystujące m.in. opisane mechanizmy.

Przedtem należy wspomnieć o drugiej grupie rejestrów współpracujących z TIMEREM 2, czyli o rejestrach wyłapujących CT0-CT3 i związanej z nimi licznej gromadce nowych przerwań.

Rejestry wyłapujące składają się z dwóch bajtów, a ich adresy są następujące:
- CT0 -CTH0 (0CCh), CTL0 (0ACh)
- CT1 -CTH1 (0CDh), CTL1 (0ADh)
- CT2 -CTH2 (0CEh), CTL2 (0AEh)
- CT3 -CTH3 (0CFh), CTL3 (0AFh)

Rejestry wyłapujące mogą być tylko odczytywane.

Działanie rejestru wyłapującego polega na zapamiętaniu wartości TIMERA 2 w chwili, gdy na wejściu sterującym działaniem rejestru pojawi się odpowiedni sygnał. Działaniem rejestrów sterują piny p1. Wejście p1.0 steruje CT0, p1.1 -CT1, p1.2 -CT2, p1.3 -CT3. Można ustalić które zbocze sygnału podanego na p1 uaktywni rejestr wyłapujący, może to być zbocze narastające, opadające lub reakcja nastąpi na każde z nich. Dokonuje się tego przez ustawienie bitów w rejestrze CTCON (0EBh). Jego struktura jest następująca:
- b0- CT0 aktywny przy zboczu narastajacym b1- CT0 aktywny przy zboczu opadającym,
- b2- CT1 aktywny przy zboczu narastajacym b3- CT1 aktywny przy zboczu opadającym,
- b4- CT2 aktywny przy zboczu narastajacym b5- CT2 aktywny przy zboczu opadającym,
- b6- CT3 aktywny przy zboczu narastającym b7- CT3 aktywny przy zboczu opadającym.

Określenie w jakiej sytuacji p1.0-3 jest aktywny pozwala nie tylko zapamiętywać w rejestrach CT wartość TIMERA 2. Może także wywołać przerwanie oddzielne dla każdego wejścia.

Całą tą „baterią" nowych przerwań związanych z TIMEREM 2 steruje rejestr IEN1 (0E8h), którego indywidualnie adresowane bity mogą każde z przerwań dopuszczać lub wyłączać jeśli są wyzerowane. Adresy poszczególnych bitów i ich funkcje są następujące:
- ECT0 (0E8h)- zezwolenie na przerwanie od CT0,
- ECT1 (0E9h)- zezwolenie na przerwanie od CT1,

- ECT2 (0EAh)- zezwolenie na przerwanie od CT2,
- ECT3 (0EBh)- zezwolenie na przerwanie od CT3,
- ECM0 (0ECh)- zezwolenie na przerwanie od CM0,
- ECM1 (0ED0)- zezwolenie na przerwanie od CM1,
- ECM2 (0EEh)- zezwolenie na przerwanie od CM2,
- ET2 (0EFh)- zezwolenie na przerwanie przeładowaniem TIMERA 2.

Wszystkie przerwania można oczywiście globalnie wyłączyć wyzerowaniem bitu EA w rejestrze IEN0 tak, jak to jest w 8051. Priorytet przerwań związanych z TIMEREM 2 można ustawić w rejestrze IP1 (0F8h), którego najmłodszy indywidualnie adresowany bit przyznaje wysoki priorytet przerwaniu CT0, a najstarszy odpowiednio przerwaniu przeładowaniem TIMERA 2.

Każde z przerwań w momencie jego zaistnienia ustawia swoją flagę, czyli indywidualnie adresowany bit w rejestrze TM2IR (0C8h). Ich adresy i powiązanie z przyczynami przerwań są następujące:

- CTI0 (0C8h)- przerwanie od CT0,
  CTI1 (0C9h)- przerwanie od CT1,
- CTI2 (0CAh)- przerwanie od CT2,
  CTI3 (0CBh)- przerwanie od CT3,
- CMI0 (0CCh)- przerwanie od CM0, CMI1 (0CDh)- przerwanie od CM1,
- CMI2 (0CEh)- przerwanie od CM2, T2OV (0CFh)- przerwanie przeładowaniem TIMERA 2.

W odróżnieniu od innych przerwań wszystkie te flagi nie są automatycznie kasowane przy powrocie z podprogramu przerwania rozkazem RTI, lecz muszą być kasowane programowo.

Tak rozbudowany system przerwań jest kolejną ważną cechą procesora. Wielu początkujących programistów nie lubi używać przerwań uważając, że trudno nad nimi zapanować. W istocie jednak przerwania bardzo ułatwiają życie. Wyobraźmy sobie np. sytuację, gdy procesor ma zapalić diodę LED po stwierdzeniu naciśnięcia przycisku. Podczas wykonywania tego zadania w sposób czysto programowy, procesor musiał by poświęcić cały czas na kontrolę przycisku, w przeciwnym wypadku zajęty realizacją innej części programu mógłby przegapić jego naciśnięcie lub zareagować na nie z dużym opóźnieniem. Jeżeli naciśnięcie przycisku może wywołać przerwanie sytuacja jest dużo prostsza. Używając przerwań trzeba pamiętać o kilku ważnych sprawach:

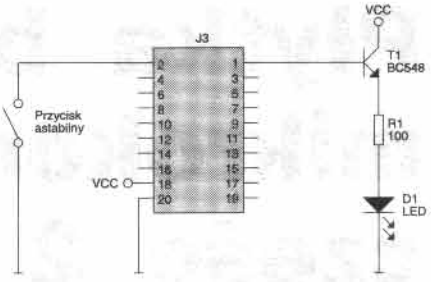
a/ przerwanie zawiesza wykonywanie bieżącego programu. Przerwanie o niższym lub takim samym priorytecie nie będzie obsłużone dopóki nie zostanie zakończony podprogram przerwania trwającego. Przerwanie o wyższym priorytecie przerywa obsługę innych przerwań realizując własny podprogram;

b/ realizacja podprogramu przerwania może zmienić zawartość rejestrów ogólnego przeznaczenia używanych przez program główny w momencie zaistnienia przerwania. Dlatego też należy zabezpieczyć zawartość takich rejestrów jak akumulator, rejestry R0-7, flaga carry itd. a przed powrotem z przerwania należy odtworzyć ich pierwotną wartość;

c/ flagi niektórych przerwań nie są kasowane automatycznie. W takim przypadku przed zakończeniem programu przerwania należy wyzerować flagę danego przerwania.

```
;*************************************************************
;* LISTING 3                                                 *
;* Fragment programu miernika refleksu                       *
;*************************************************************
org 0
jmp start
org 0bh
jmp wyswietlacz         ;wektor przerwania timera0
org 033h
jmp przycisk            ;wektor przerwania przycisku
org 05bh
jmp pomiar_start        ;wektor przerwania CM0
org 073h
jmp t2overflow          ;wektor przerwania po przeladowaniu
                        ;timera2
start:
mov wys1,#9     ;ustawienie parametrow poczatkowych
                ;wyswietlacza
mov wys2,#9
mov wys3,#9
mov wys4,#9

mov tm2con,#10000001b  ;zaprogramowanie rejestru kontroli
;timera2
                       ;zezwolenie na przerwanie
                       ;przeladowaniem timera2
                       ;timer2 taktowany f.osc/12
mov ctcon,#00000010b   ;przerwanie przycisku bedzie wyzwalane
                       ;opadajacym zboczem (SW1 zewrze p1.0
;do masy)
setb et2        ;zezwolenie na przerwanie przeladowaniem timera2
mov cml0,#0     ;ladowanie rejestrow porownan CM0
mov cmh0,#0
setb f0


t2overflow:     ;przerwanie przeladowania timera2
push acc
inc los         ;inkrementacja rejestru pseudolosowego
jb f0,t2o2      ;pomiar jeszcze sie nie rozpoczal
mov a,t2htime
inc a
mov t2htime,a
cjne a,#15,t2o2
                ;czas pomiaru zbliza sie do 1s, slaby refleks
clr ecm0        ;wylaczenie przerwania porownania timera2 i
                ;CM0
clr ect0        ;przerwanie przycisku wylaczone
clr t20v        ;skasuj flage przerwania
setb f0         ;ustaw znacznik zakonczenia pomiaru

pop acc
reti            ;i wroc z przerwania
t2o2:
clr t20v
pop acc
reti

pomiar_start:   ;przerwanie rejestrow porownan CM0
setb ect0       ;wlaczenie przerwania przycisku
clr cmi0        ;skasowanie flagi przerwania
reti

przycisk:       ;przerwanie po nacisnieciu przycisku refleksu
setb f0         ;ustaw znacznik zakonczenia pomiaru
clr ecm0        ;wylaczenie przerwania rownosci timera2 i CM0
clr ect0        ;przerwania przycisku wylaczone
clr cti0        ;skasuj flage przerwania
reti

end
```

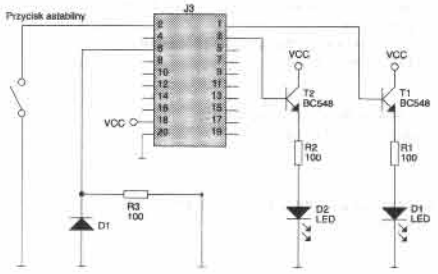
Listing 3.



Rys. 4. Schemat interfejsu miernika refleksu.

Miernik refleksu

Dla ilustracji działania TIMERA 2 i związanych z nim przerwań posłuży zbudowanie prostego miernika refleksu. Na **rys.4** pokazany jest schemat układu, który trzeba dołączyć do płytki prototypowej. Ponieważ program miernika jest już znacznie dłuższy od poprzednich, niemożliwa jest prezentacja jego pełnego wydruku. Znajdzie się on razem z innymi omawianymi programami na



Rys. 5. Schemat elektryczny interfejsu częstościomierza.

dyskietce dołączanej do płytki prototypowej. Jego najważniejsze fragmenty pokazano na **listingu 3.** Po naciśnięciu i puszczeniu przycisku SW1 gaśnie dioda D1. Po upływie czasu, którego długość zmienia się losowo od 0.5s do ok. 4s dioda zapala się. Powtórne naciśnięcie SW1 jest odpowiedzią na ten sygnał. Miernik mierzy czas między zaświeceniem się diody a naciśnięciem przycisku.

Program najpierw deklaruje szereg rejestrów niezbędnych do jego działania. Pod adresami 0Bh, 033h, 05Bh, 073h znajdują się skoki do podprogramów obsługujących odpowiednie przerwania. Następnie inicjowane są niektóre rejestry, ustawiane tryby pracy TIMERA 0 i TIMERA 2, rejestru CM0. W pominiętej na wydruku części programu oczekuje on na naciśnięcie przycisku, eliminując potem jego drgania przez filtr programowy. Po pauzie 0.5s program znowu wstrzymuje swoje działanie na czas zależny od wartości zapisywanej do rejestru los. Zawartość tego rejestru jest zwiększana przy każdym przeładowaniu TIMERA 2. W momencie, gdy wpisana do rejestru liczba ma 6 najmłodszych bitów równych 0 program kontynuuje swoje działanie. Włączone zostaje zezwolenie na przerwanie rejestru CM0. Przerwanie nastąpi wtedy gdy TIMER 2 naliczy identyczną liczbę jak ta która zapisana jest w rejestrach CM0, w tym przypadku 00h. Program przerwania pomiar_start włącza m.in. zezwolenie na przerwanie CT0, które będzie wywołane przez kolejne naciśnięcie SW1. Gdy to nastąpi w rejestrach CTH0 i CTL0 znajdzie się wartość, którą przyjął w tej chwili TIMER 2. Wartość ta zsumowana z ilością przeładowań TIMERA 2, która jest zapamiętana w rejestrze *t2htime* pozwala obliczyć czas jaki minął od zapalenia D1 do naciśnięcia przycisku. Licznik potrafi zmierzyć czas reakcji w zakresie 0,1ms do 999,9ms. Można jeszcze dodać, że autorowi z trudem udawało się uzyskiwać wynik krótszy od 200ms, co dla większości czytelników nie będzie pewnie wielkim problemem. Miłej zabawy!

Częstościomierz - okresomierz

Drugim przykładowym urządzeniem wykorzystującym możliwości jakie daje TIMER 2 jest układ częstościomierza-okresomierza. Jego schemat zamieszczony jest na **rys.5**, a program w wersji źródłowej znajdzie się na dyskietce. W tej wersji przyrząd potrafi mierzyć sygnały prostokątne o poziomach TTL. Świecenie D1 sygnalizuje, że przyrząd mierzy częstotliwość a zapalenie D2 oznacza pomiar okresu. Przełączania funkcji dokonuje się przyciskiem SW1. Zakresy pomiarowe: 1Hz-9999Hz i 0.1ms-999,9ms. W pewnym sensie program częstościomierza stanowi rozwinięcie programu miernika refleksu. Tak jak w przypadku pozostałych programów, autor zachęca czytelników do jego przeróbek, rozszerzania zakresu pomiarowego i dokładności. Dla zaczynających programować mogą to być pożyteczne wprawki i spora satysfakcja, gdy uda się stworzyć „ambitny" program.

Interfejs I²C

Procesor 80552 wyposażony został w dodatkowy szeregowy port wejścia/wyjścia, który można podłączyć do magistrali I2C. Ten format wymiany informacji z otoczeniem stosowany jest przez wiele specjalizowanych układów scalonych. Zastosowanie ich do współpracy z procesorem nie wyposażonym w taki interfejs wymaga pisania specjalnych procedur obsługi I2C. W przypadku mikrokontrolera 80552 problem ten jest znacznie uproszczony. Zanim przejdziemy do omawiania procedur obsługi magistrali najpierw kilka słów o jej elementach charakterystycznych. Magistrala oprócz wspólnej dla wszystkich układów masy, składa się jedynie z dwóch linii sygnałowych: zegarowej SCL i danych SDA. Układy przyłączone do magistrali mogą mieć albo status nadrzędny (master) albo podporządkowany (slave). Każdy z układów ma swój unikalny adres wywołania. Tylko master może zainicjować i kontrolować transmisję wybierając układ do którego chce przesłać dane lub z którego chce je otrzymać. Transmisja rozpoczyna się sekwencją startu, po której następuje 7-bitowy adres urządzenia wywoływanego i bit kierunku transmisji. Urządzenie wywoływane powinno odpowiedzieć potwierdzeniem ACK, ustawiając na linii SDA poziom niski na czas trwania jednego taktu zegara. Po nawiązaniu transmisji następuje przesłanie żądanej liczby bajtów. Po każdym odebranym bajcie urządzenie odbiorcze obowiązane jest wysłać sygnał ACK. Transmisję kończy master sekwencją stopu.

Procesor 80552 może być zaprogramowany niezwykle elastycznie. Może pełnić rolę nadrzędną lub podporządkowaną, może wysyłać i odbierać dane, można mu nadawać własny adres wywołania.

Procesor współpracuje z magistralą I2C przy pomocy 4 rejestrów umieszczonych w obszarze SFR. Ich funkcje oraz znaczenie poszczególnych bitów są następujące:

- S1ADR (0DBh) -bity b7-1 zawierają siedmiobitowy adres wywołania procesora na który będzie odpowiadał w trybach slave. Ustawienie b0 spowoduje, że procesor odpowie także na adres wywołania ogólnego 00h. Zawartość tego rejestru jest nieistotna w trybach nadrzędnych;
- S1DAT (0DAh) -rejestr danych. Podczas odbioru odczytywany jest z niego ostatni odebrany bajt. W czasie nadawania, program wpisuje tu bajt do wysłania;
- S1CON (0D8h) -rejestr kontrolny, którego bity mogą być adresowane indywidualnie:

b6 -ENSI (0DEh) ustawienie bitu powoduje dołączenie procesora poprzez porty p1.6 i p1.7 do magistrali,

b5 -STA (0DDh) flaga startu. Program ustawia ją gdy chce rozpocząć transmisję,

b4 -STO (0DCh) flaga stopu. Program ustawia ją gdy chce zakończyć transmisję. Gdy na magis-

```
      ;*****************************************************
      ;* LISTING 5                                         *
      ;* Procedury I/O I2C procesora 80552 dla trybu master *
      ;* wykorzystujace 4 strone pamieci programu          *
      ;*****************************************************

extrn  data (mrd,mtd,numbyt,sla,hadd,backup) ;zmienne
                                             ;zdefiniowane
extrn  bit  (i2cerr)    ;w programie glownym
public i2cint,i2cinit   ;nazwa tych adresow bedzie dostepna w
                        ;programie glownym
using 1                 ;uzywany zestaw rejestrow nr.1

org 400h
baza:
org baza+00h            ;obsluga bledu magistrali
mov s1con,#0d4h
setb i2cerr             ;ustaw flage bledu!
pop psw
reti
org baza+08h            ;start transmisji
mov s1dat,sla
mov s1con,#0c4h
ajmp initbase
org baza+010h           ;start powtorzony
mov s1dat,sla
mov s1con,#0c4h
ajmp initbase
org baza+018h           ;po wyslaniu adresu urzadzenia
                        ;podporzadkowanego
mov psw,#08h            ;odebrano potwierdzenie
mov s1dat,@r1
ajmp con
org baza+020h           ;po wyslaniu adresu urzadzenia
                        ;podporzadkowanego
mov s1con,#0d4h         ;brak potwierdzenia
setb i2cerr             ;ustaw znacznik bledu!
pop psw
reti
org baza+028h           ;wyslany kolejny bajt danych odebrano
                        ;potwierdzenie
djnz numbyt,notldat1
mov s1con,#0d4h
ajmp retmt
org baza+030h           ;po wyslaniu bajtu danych nie odebrano
                        ;potwierdzenia
mov s1con,#0d4h
setb i2cerr             ;ustaw flage bledu!
pop psw
reti
org baza+038h           ;utracony arbitraz magistrali
mov s1con,#0e4h
mov numbyt,backup
ajmp retmt
org baza+040h           ;zapoczatkowanie odbioru bajtow
                        ;trybie master
mov s1con,#0c4h
pop psw
reti
org baza+048h           ;brak potwierdzenia po wyslaniu adresu
                        ;urzadzenia
stop: mov s1con,#0d4h   ;podporzadkowanego majacego nadawac
setb i2cerr             ;ustaw flage bledu!
pop psw
reti
org baza+050h           ;odebrany bajt z urzadzenia
                        ;podporzadkowanego
mov psw,#08h
mov @r0,s1dat
ajmp rec1
org baza+058h           ;bajt zostal odebrany potwierdzenia
                        ;nie bedzie
mov psw,#08h
mov @r0,s1dat
ajmp stop

i2cinit:                ;procedura inicjalizacji zlacza I2C
mov s1adr,#031h
setb p1.6
setb p1.7
orl ien0,#0a0h
clr ps1
mov s1con,#0c4h
ret

i2cint:                 ;procedury przerwania zlacza I2C
push psw
push s1sta
push hadd
ret

initbase:
mov psw,#08h
mov ar1,mtd
mov ar0,mrd
mov backup,numbyt
pop psw
reti

notldat1:
mov psw,#08h
mov s1dat,@r1
con: mov s1con,#0c4h
inc ar1
retmt: pop psw
reti

rec1:
djnz numbyt,notldat2
mov s1con,#0c0h
ajmp retmr
notldat2: mov s1con,#0c4h
retmr: inc ar0
pop psw
reti

end
```

Listing 4.

trali rozpoznana zostanie sekwencja stop, flaga kasowana jest sprzętowo,

b3 -SI (0DBh) flaga przerwania magistrali I2C. Musi być kasowana przez program po obsłużeniu przyczyny wywołującej przerwanie,

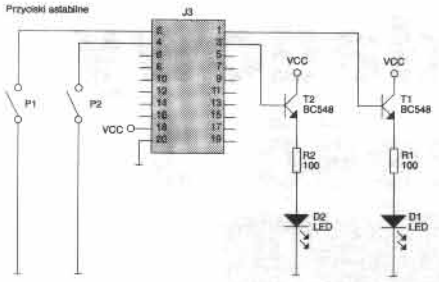
b2 -AA (0DAh) flaga potwierdzenia. Ustawienie tej flagi spowoduje wysłanie sygnału potwierdzenia ACK gdy procesor rozpozna swój adres wywołania lub gdy potwierdza odbiór kolejnego bajtu danych,

b7,b1,b0 -CR2, CR1, CR0 (0DFh), (0D9h), (0D8h) bity szybkości transmisji. Magistrala I2C dopuszcza maksymalną szybkość transmisji do 100kHz. Przy długich ścieżkach łączących układy i większych pojemnościach związanych z liczbą dołączonych do magistrali układów, szybkość powinna być niższa,

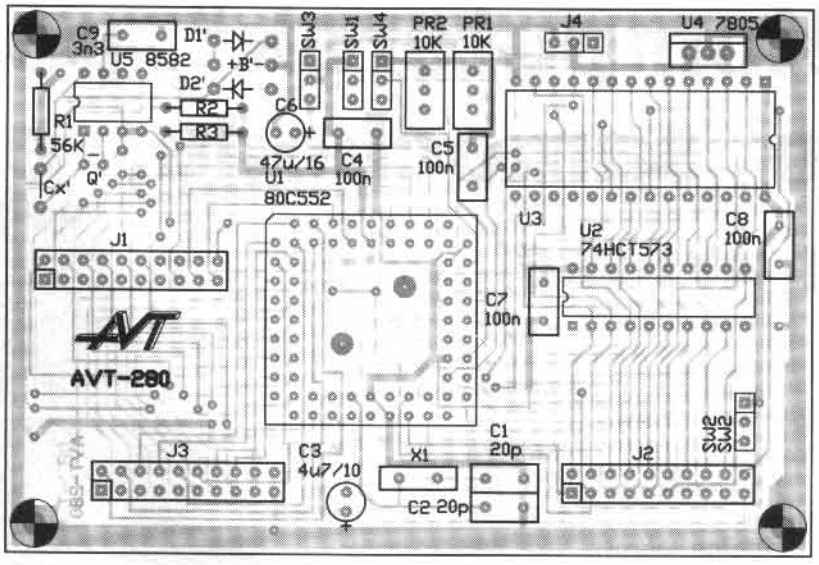
- S1STA (0D9h) -rejestr statusowy. Jego trzy najmłodsze bity są zawsze wyzerowane. Pozostałe zawierają kod statusu zależnie od sytuacji jaka ma aktualnie miejsce podczas dostępu do magistrali. Możliwych jest 26 kodów na które program obsługi przerwania musi zareagować.

W dokumentacji technicznej producent procesora podaje przykład efektywnego i ciekawego programu obsługi interfejsu I2C procesora. Zajmuje on jedną, wybraną, stronę pamięci programu i zaczyna się od samego jej początku. Program ten z małymi modyfikacjami przedstawiony jest na **list. 4.**

Obsługa przerwania rozpoczyna się skokiem pod adres *i2cint*. Najpierw odkładana jest na stos bieżąca zawartość rejestru statusowego procesora PSW, który w trakcie obsługi przerwania będzie zmieniany. Potem na stos odkładany jest kod statusu przerwania odczytany z rejestru S1STA i numer strony pamięci na której znajduje się program obsługujący przerwanie zapisany w rejestrze *hadd*. Krótki programik kończy się rozkazem RET. Zgodnie z zasadami działania procesora traktuje on dwie ostatnie dane odłożone na stos jako adres powrotny pod który ma skoczyć. I rzeczywiście, znajdzie pod tym adresem dalszy ciąg programu obsługi przerwania odpowiedni dla zgłaszanego statusu. Łatwo to sprawdzić. Jeżeli pojawia się przerwanie magistrali I2C ze statusem o numerze 018h (oznacza on, że po wysłaniu adresu urządzenia podporządkowanego odebrano z tego urządzenia potwierdzenie), to program skoczy pod adres 0418h, i znajdzie tam dalszy ciąg obsługi tego zdarzenia. To właśnie jest przyczyną, że podprogramy obsługu-



Rys. 6. Schemat interfejsu wykorzystywanego przez zegar



Rys. 7. Rozmieszczenie elementów na płytce drukowanej.

jące przerwanie magistrali zajmują jedną stronę pamięci i zaczynają się na jej początku. Najmniejszy kod statusu wynosi bowiem 00h. Na zakończenie obsługi przerwania odtwarzany jest pierwotny PSW i wykonywany rozkaz RTI - powrót z przerwania.

Korzystanie z magistrali I2C powinno zaczynać się od uaktywnienia procedur obsługi. Jeśli korzysta się z programu przedstawionego na list. 4, oznacza to wykonanie na początku programu głównego rozkazu CALL *i2cinit*. Od tego momentu procesor za pomocą magistrali może komunikować się z układami podrzędnymi. Sposób zapisu bajtów do urządzenia podporządkowanego i ich odczyt przedstawione zostały na **list. 5**. Są to sekwencje, które powinny znaleźć się w programie głównym korzystającym z magistrali.

Przykładem wykorzystania magistrali I2C jest program prostego zegara, który znajdzie się na dyskietce. Konstrukcja płytki prototypowej przewiduje umieszczenie w podstawce U5 pamięci EEPROM 8582 lub podobnej, albo zegara czasu rzeczywistego 8583. Zależnie od wybranego układu montaż tej części płytki będzie przebiegał odmiennie. Najpierw bez względu na rodzaj układu należy wlutować rezystory podciągające R2 i R3. Dla układu 8582 należy także wlutować elementy R1 i C9.

W przypadku zastosowania układu 8583 montaż trzeba rozpocząć od przecięcia ścieżek na górnej stronie płytki zwierających wyprowadzenia 1, 2 i 3 U5 oraz ścieżki zwierającej D2'. Następnie w miejscu oznaczonym jako Cx' należy wlutować miniaturowy trymer lub kondensator o pojemności ok. 10pF, w miejscu Q' rezonator zegarkowy 32,768kHz, oraz dwie diody D1' i D2' zgodnie z oznaczeniami. Do miejsc oznaczonych jako + i - B' należy podłączyć ogniwo o napięciu 3-5V podtrzymujące działanie zegara gdy wyłączone jest zasilanie płytki. Elementy dodatkowe dołączane jeszcze do płytki pokazane są na **rys.6**. Działanie zegara jest bardzo proste i nie wyczerpuje możliwości jakie posiada układ 8583. Może on wyświetlać godziny i minuty, a po naciśnięciu SW1 miesiąc i dzień. Ustawianie zegara odbywa się po naciśnięciu SW2. Zaczyna wtedy migotać wyświetlacz zegara, a ustawianie jest możliwe dzięki SW1. Jeżeli nie migocze żaden wyświetlacz oznacza to, że zegar jest ustawiony i pracuje normalnie. Świecenie odpowiedniej diody D1 lub D2 wskazuje czy wyświetlany jest czas czy data. Program co minutę odczytuje z zegara dane i aktualizuje swoje rejestry. Przy wychodzeniu z opcji ustawiania program zapisuje do 8583 nowe ustawienie czasu i daty.

Montaż układu należy przeprowadzić na dwustronnej płytce drukowanej, której widok przedstawiono w EP4/96. Rozmieszczenie elementów na płytce przedstawia **rys.7**. Układ jest prosty w montażu i zastosowanie podstawowych zasad montażu elektronicznego zapewnia poprawne działanie układu od razu po zmontowaniu.

Na zakończenie życzę Czytelnikom dobrej i pożytecznej zabawy z procesorem 80552.

Ryszard Szymaniak, AVT

```
;***********************************************
;* LISTING 6                                   *
;* Procedury odczytu i zapisu danych do 8583   *
;* poprzez magistrale I2C                      *
;***********************************************

odczyt:             ;procedura odczytu danych z zegara
 mov 30h,#3         ;wyslanie subadresu
 mov mtd,#30h       ;adres bufora danych transmitowanych
 mov numbyt,#1      ;ilosc transmitowanych danych
 mov sla,#0a0h      ;adres ukladu podrzednego
 clr i2cerr         ;zerowanie flagi bledu magistrali
 setb sta           ;inicjacja odczytu
odczyt1: jnb i2cerr,odczyt11
 mov a,#0ffh
 ret
odczyt11: mov a,numbyt
 jnz odczyt1
 mov mrd,#30h       ;adres burfora danych odbieranych
 mov numbyt,#4      ;odczyt 4 bajtow
 mov sla,#0a1h      ;adres ukladu podrzednego
 clr i2cerr
 setb sta
odczyt2: jnb i2cerr,odczyt21
 mov a,#0ffh
 ret
odczyt21: mov a,numbyt

                    ;procedura zapisu danych do zegara
 mov 30h,#3         ;zapis do bufora subadresu
 mov mtd,#30h       ;adres bufora danych transmitowanych
 mov numbyt,#5      ;wyslanie subadresu i 4 bajtow
 mov sla,#0a0h      ;adres ukladu podrzednego
 clr i2cerr
 setb sta           ;inicjacja zapisu
zapis1: jnb i2cerr,zapis11
 ret
zapis11: mov a,numbyt
 jnz zapis1
 ret
```

Listing 5.