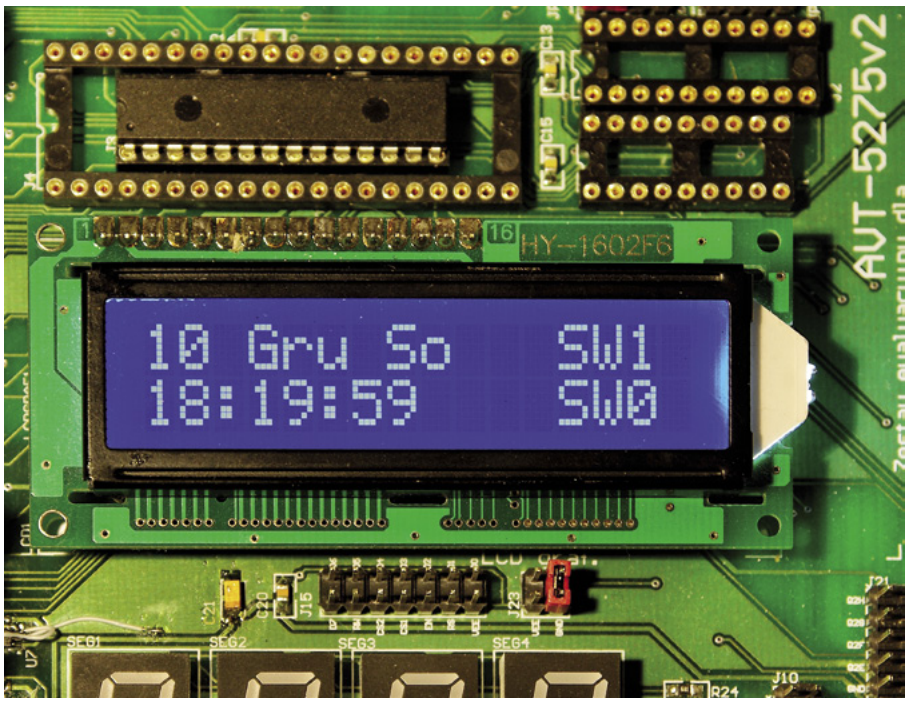
Kurs programowania mikrokontrolerów PIC (9)

Obsługa układu zegarowego M41T00

Interfejs I²C został zaprojektowany przez firmę Philips i jest przeznaczony do lokalnej wymiany danych pomiędzy układami. W najczęściej stosowanej konfiguracji do magistrali jest dołączony pojedynczy układ master (sterownik nadrzędny, najczęściej mikrokontroler) i jeden lub kilka układów slave (układów peryferyjnych). Dzięki swoim zaletom ten standard komunikacji zyskał ogromną popularność i współcześnie, pomimo rosnących wymagań odnośnie do szybkości komunikacji pomiędzy układami w systemie, nadal jest oferowana ogromna rzesza różnorodnych układów. W artykule opisano sposób obsługi nowoczesnego, popularnego układu zegarowego.



Na płytce ewaluacyjnej AVT-5275 jest zamontowany ciekawy, nowoczesny układ zegara czasu rzeczywistego typu M41T00 produkowany przez firmę STM.

Podstawowe informacje na temat M41T00

Układ ma wbudowany oscylator stabilizowany za pomocą zewnętrznego rezonatora kwarcowego o częstotliwości 32768 Hz. Liczniki czasu z danymi w kodzie BCD są umieszczone w 8 komórkach pamięci RAM (rejestrach). Układ nadzoru napięcia zasilającego automatycznie przełącza zasilanie na podtrzymywanie bateryjne w wypadku zaniku głównego napięcia zasilania. Bateria zasilająca jest dołączana do dodatkowej nóżki układu i zapewnia energię do podtrzymania zawartości rejestrów liczników i działania generatora kwarcowego. Przy napięciu +3 V układ pobiera prąd o natężeniu ok. 0,8 µA.

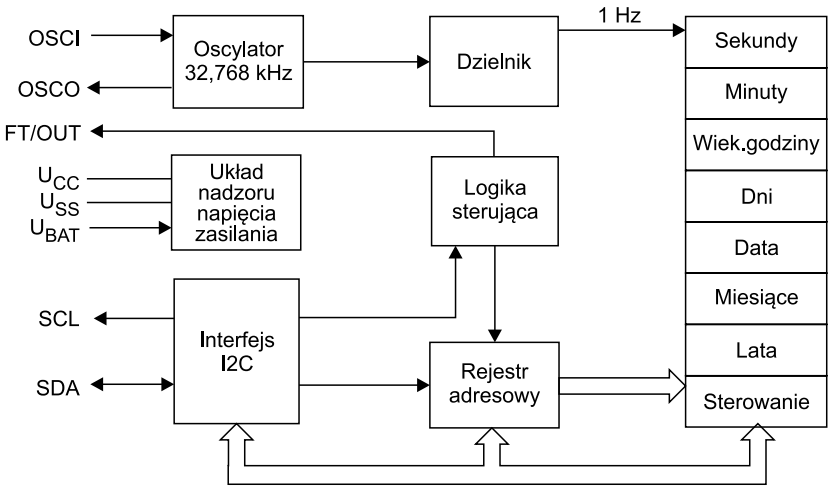
Rejestry zegara zliczają czas w kodzie BCD. Cztery młodsze bity rejestru zawierają jednostki, natomiast cztery starsze dziesiątki liczby zapisanej w kodzie BCD. Na przykład liczba 18 jest reprezentowana w kodzie BCD przez zapis w rejestrze dziesiątek wartości 0001, a w rejestrze jednostek wartości 1000 (bajt szesnastkowo=0x18). Wartości zapisane na połówkach bajtu tworzących liczbę dziesiętną nie mogą być (co oczywiste) większe od cyfry 9. Na **rysunku 1** pokazano schemat blokowy układu zegarowego M41T00 oraz mapę jego rejestrów.

Rejestry układu zapisuje się i odczytuje za pomocą interfejsu I²C. Adres układu zegarowego ma stałą wartość równą 0xD0 z bitem R/W=0 i 0xD1 z bitem R/W=1 (**rysunek 2**). Odczytywanie danych zaczyna się od wysłania sekwencji *Start*, adresu układu z bitem R/W=0 i adresu odpowiedniego rejestru układu – **rysunek 3**. Ta część transmisji wpisuje do zegara adres, od którego będzie się zaczynało odczytywanie danych. Właściwy odczyt danych rozpoczyna się od ponownego wysłania sekwencji *Start* (*Repeated Start*) i adresu układu zegarowego z bitem R/W=1. Jako pierwsza zostanie odczytana zawartość rejestru o już ustalonym adresie. Każde kolejne odczytanie danych zwiększa o jeden licznik adresowy i nie ma potrzeby modyfikowania go za każdym razem, gdy potrzeba odczytać kolejny rejestr. Sekwencja odczytu kończy się wysłaniem na magistralę bitu braku potwierdzenia NAK po odczytaniu bajtu i sekwencji *Stop*.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 18453, pass: 5eyp1854
• poprzednie części kursu

Zapisywanie danych do układu zaczyna się od wysłania sekwencji *Start* i adresu układu zegarowego z bitem R/W=0. Następnie jest wysyłany adres rejestru, od którego będzie zapisany pierwszy bajt danej. Zapisanie danej powoduje automatyczne zwiększenie o 1 licznika adresowego (**rysunek 4**). Możliwe jest zapisywanie kolejnych danych bez wysyłania sekwencji: adres slave, adres rejestru. Sekwencja zapisywania kończy się wysłaniem przez mikrokontroler na magistralę sekwencji STOP.

Układ M41T00 jest dostępny w 8-nóżkowej obudowie do montażu powierzchniowego. Do wyprowadzeń oscylatora OSCI i OSCO jest dołączany standardowy kwarc zegarkowy o częstotliwości 32768 Hz. Linie danych SDA i zegarowa SCL są liniami interfejsu I²C. Wejście VBAT jest wejściem dla dodatniego bieguna baterii zasilania pomocniczego. W naszym układzie do wejścia



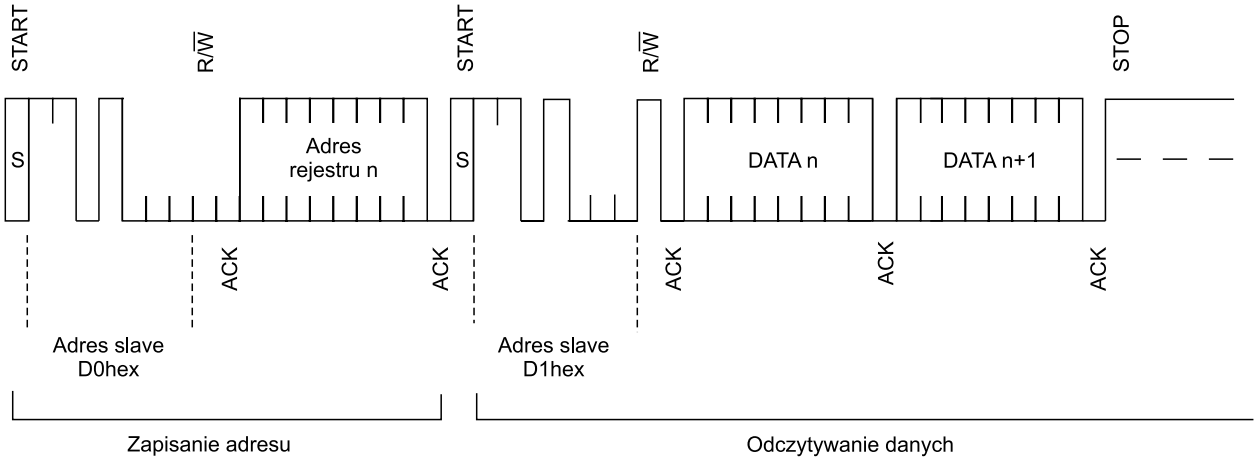
Adres	Dane								Funkcja/zakres format BCD	
	D7	D6	D5	D4	D3	D2	D1	D0		
0	ST	10 sekund			1 sekund				Sekundy	00…59
1	X	10 minut			1 minut				Minuty	00…59
2	CEB(2)	CB	10 godzin		1 godzin				Wiek/godziny	0…1/00…23
3	X	X	X	X	X	Dzień tygodnia			Dzień tyg.	01…07
4	X	X	10 dni		Dzień kalendarzowy				Dzień	01…31
5	X	X	X	10 minut	Miesiąc				Miesiąc	01…12
6	10 lat				Rok				Lata	00…99
7	OUT	FT	S	Kalibracja					Sterowanie	

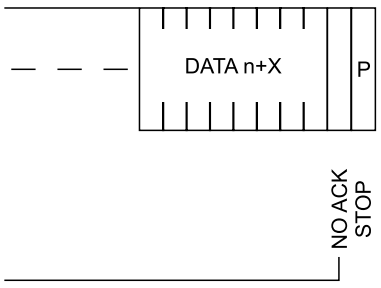
S – bit znaku, FT – bit testu częstotliwości, ST – bit zatrzymania zegara, OUT – poziom logiczny wyjścia, CEB – odblokowanie zliczania stuleci, CB – bit licznika stuleci

Rysunek 1. Schemat blokowy układu M41T00 oraz mapa jego rejestrów

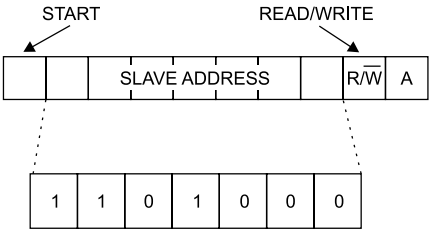
VBAT jest dołączony kondensator C28 o dużej pojemności ładowany z napięcia Vdd przez rezystor R22. Szczegółowy schemat połączeń można znaleźć na schemacie płytki ewaluacyjnej opublikowanym w Elektronice Praktycznej nr 2/2011.

Wyprowadzenie FT/OUT spełnia dwie funkcje. Jeżeli bit FT rejestru sterowania o adresie 7 jest wyzerowany, to na tym wyprowadzeniu pojawia się poziom wpisany do bitu OUT. Można tego wyjścia użyć np. do sterowania przekaźnikiem załączającym układy alarmu. Ustawienie bitu FT powoduje, że na wyjściu FT/OUT pojawia się przebieg o częstotliwości 512 Hz. Mierząc tę częstotliwość można wykonać kalibrację częstotliwości oscylatora przez wpisanie odpowiedniej wartości do bitów D0…D4 rejestru sterowania. Sposób kalibracji jest dokładnie opisany w dokumentacji układu.





Rysunek 3. Sekwencja odczytywanie danych z układu M41T00



Rysunek 2. Adres slave M41T00

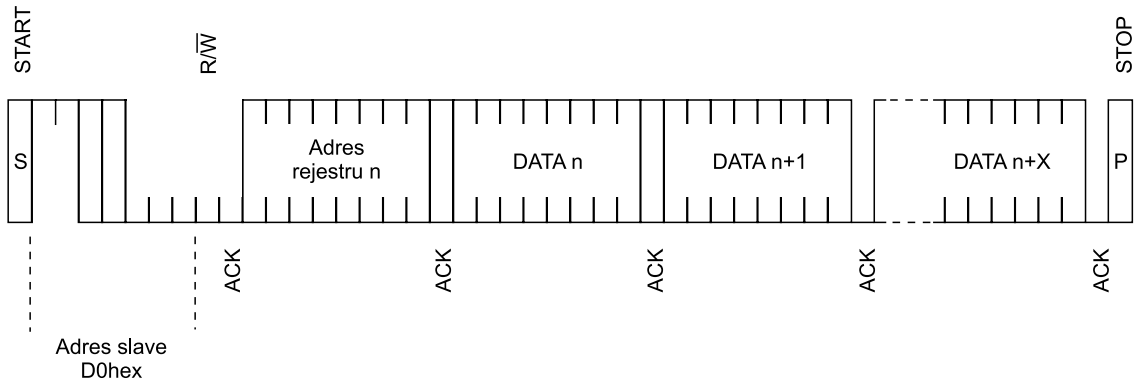
Przykładowy program zegara czasu rzeczywistego

Próby z układem rozpoczniemy od dołączenia linii interfejsu I²C do odpowiednich portów I/O mikrokontrolera. Najwygodniej będzie połączyć je do linii RC3 i RC4 mikrokontrolera, tak aby można było skorzystać ze sprzętowego interfejsu MSSP pracującego w trybie *I2C Master*. Po włączeniu zasilania układu zostaje wyzerowany bit FT i ustawiony bit OUT. Do pozostałych rejestrów wpisywane są wartości przypadkowe. Ponieważ w rejestrze sterującym umieszczone są bity kalibracji, to po pierwszym włączeniu zasilania konieczna będzie jego inicjalizacja. Na **listingu 1** zamieszczono funkcję zapisującą rejestr sterujący *control*.

Dla czytelności tego opisu przyjmijmy, że funkcje układu związane ze wskazaniem godziny będziemy nazywać *zegarem*, natomiast ze wskazaniem daty – *kalendarzem*.

W każdym układzie zegarowym czasu rzeczywistego muszą być wbudowane funkcje ustawiania zegara (godziny/minuty/sekundy) i kalendarza (dzień/miesiąc/rok). Ze względów praktycznych warto rozdzielić je funkcjonalnie. Zegar jest zwykle ustawiany częściej (korekta chodu, zmiana czasu z zimowego na letni itp.), a wówczas nie ma potrzeby wykonywania nastaw kalendarza. Z drugiej strony, można zmienić wskazania kalendarza nie zakłócając pracy zegara.

Dla przykładowych funkcji obsługujących zegar M41T00 została zdefiniowana struktura *czas* pokazana na **listingu 2**. Taka definicja powoduje, że kod będzie bardziej czytelny. Odwołanie się do zmiennej zawierającej na przykład licznik godzin wygląda



Rysunek 4. Sekwencja zapisywania danych do układu M4T00

tak: *czas.godz*. Zapisywanie rejestrów zegara wykonuje funkcja *WriteCzasRTC* pokazana na **listingu 3**. Adres pierwszego rejestru jest równy 0x00. Do tego rejestru jest wpisywany stan licznika sekund równy zmiennej *czas.sek*. Potem kolejno zapisywany jest rejestr minut równy zmiennej *czas.min* i rejestr godzin równy zmiennej *czas.godz*.

Procedura zapisywania rejestrów kalendarza (**listing 4**) ustawia wskaźnik adresu początkowego funkcji zapisu rejestrów na adres licznika dni (0x03). Dalej stan rejestrów kalendarza jest inicjowany analogicznie jak rejestrów zegara, z tym, że używane są wartości struktury odpowiadające kalendarzowi.

Odczytywanie rejestrów zegara jest trochę bardziej skomplikowane, niż ich zapisywanie. Jak to zostało pokazane na rys. 3, najpierw trzeba wysłać adres układu RTC z wyzerowanym bitem R/W, adres rejestru, sekwencję powtórzonego startu i ponownie adres układu RTC z ustawionym bitem R/W. Funkcja *ReadRTC* z **listingu 5** odczytuje wszystkie rejestry zegara (oprócz sterującego) i zapisuje ich wartość do struktury *czas*. Co oczywiste, ich odczytywanie musi być wykonywane cyklicznie, ponieważ jest ono niezbędne do wykonania funkcji wyświetlającej i bieżącą godzinę i datę.

Funkcja *DispTime* służy do wyświetlania na ekranie alfanumerycznego wyświetlacza LCD godzin minut i sekund. Jej wygląd pokazano na **listingu 6** a efekt działania na **fotografii 5**. Wywołanie funkcji *ReadRTC* powoduje odczytanie zwartości wszystkich rejestrów M41T00 i ich zapisanie w strukturze *czas*. Jeżeli zależy nam na wyświetlaniu sekund, to dobrze by było, gdyby zmiana na pozycji sekund była wyświetlana tak szybko, jak to tylko możliwe. Dlatego funkcja *ReadRTC* powinna być wywoływana w pętli. W układach, w których mikrokontroler zajmuje się tylko odczytywaniem i wyświetlaniem czasu, takie rozwiązanie jest do przyjęcia. W urządzeniach, które oprócz pomiaru czasu wykonują inne funkcje ciągłe (np. rejestracji sygnałów wejściowych) nieprzerwane odczytywanie danych z układu zegara nie jest dobrym rozwiązaniem, bo nie starczyłoby czasu na inne funkcje. Można to rozwiązać stosując na przykład mechanizm przerwań informujących program, że minął czas 1 sekundy i trzeba pobrać i wyświetlić czas z układu RTC. W naszych przykładach nie stosujemy tej metody, aby nie komplikować przedstawienia opisu zasadniczych procedur obsługi i funkcja *DispTime* jest wywoływana w pętli.

Po odczytaniu rejestrów za pomocą funkcji *ReadRTC* jest wykonywane sprawdzenie czy zmienił się licznik sekund od ostatniego odczytywania zegara. Jeżeli nie, to funkcja kończy działanie. Jeżeli tak, to jest aktualizowany stan wyświetlacza. Do sprawdzania zmian licznika sekund zdefiniowano dodatkowe pole w strukturze *czas* o nazwie *zmsek*. Ta zmienna jest inicjowana na początku programu. Jeżeli *czas.zmmsek* i *czas.msek* nie są sobie równe, to

Listing 1. Zapisanie rejestru *control*
```
void WriteControlRTC(int8 control){
  i2c_start();
  i2c_write(0xD0);                //adres slave R/W=0
  i2c_write(7);                   //adres rejestru control
  i2c_write(control);             //zapisz rejestr control
  i2c_stop();
}
```

Listing 2. Struktura czas
```
struct {
  int8 sek;
  int8 min;
  int8 godz;
  int8 dni;
  int8 data;
  int8 mies;
  int8 rok;
  int8 zmsek;
}czas;
```

Listing 3. Zapisywanie rejestrów *zegara* M41T00
```
void WriteCzasRTC(void){
  i2c_start();
  i2c_write(0xD0);                //adres slave R/W=0
  i2c_write(0);                   //adres pierwszego rejestru
  i2c_write(czas.sek);            //sekundy
  i2c_write(czas.min);            //minuty
  i2c_write(czas.godz);           //godziny
  i2c_stop();
}
```

Listing 4. Funkcja zapisania rejestrów *kalendarza* M41T00
```
void WriteDataRTC(void){
  i2c_start();
  i2c_write(0xD0);                //adres slave R/W=0
  i2c_write(3);                   //adres rejestru dni
  i2c_write(czas.dni);
  i2c_write(czas.data);
  i2c_write(czas.mies);
  i2c_write(czas.rok);
  i2c_stop();
}
```

Listing 5. Funkcja odczytania wszystkich rejestrów czasu M41T00
```
void ReadRTC(void){
  i2c_start();
  i2c_write(0xD0);                //adres slave R/W=0
  i2c_write(0);                   //adres pierwszego rejestru
  i2c_rstart();                   //powtórzony start
  i2c_write(0xD1);                //adres slave R/W=1
  czas.sek=i2c_read(0);           //sekundy
  czas.min=i2c_read(0);           //minuty
  czas.godz=i2c_read(0);          //godziny
  czas.dni=i2c_read(0);           //dni: Pn. Wt, Sr itp
  czas.data=i2c_read(0);          //dni miesiąca
  czas.mies=i2c_read(0);          //miesiace
  czas.rok=i2c_read(1);           //lata
  i2c_stop();
}
```

Listing 6. Funkcja wyświetlania czasu

```
void DispTime(void){
  ReadRTC();
  if(czas.sek==czas.zmsek)                          //sekundy bez zmian od ostatniego odczytu
  return;
  czas.zmsek=czas.sek;                              //nowa liczba sekund
  PosLcd(1,2);                                      //pozycja wyświetlania
  WriteRd(((czas.godz&0x30)>>4)+0x30);              //wyświetlenie godzin
  WriteRd((czas.godz&0x0f)+0x30);
  WriteRd(':');
  WriteRd(((czas.min&0x70)>>4)+0x30);               //wyświetlenie minut
  WriteRd((czas.min&0x0f)+0x30);
  WriteRd(':');
  WriteRd(((czas.sek&0x70)>>4)+0x30);               //wyświetlenie sekund
  WriteRd((czas.sek&0x0f)+0x30);
  if(czas.sek==0&&czas.min==0&&czas.godz==0){
    DispDay(8,1);                                   //wyświetlanie daty przy zmianie doby
    DispMies(1,1);
  }
}
```

Fotografia 5. Działanie funkcji DispTime

do zmiennej pomocniczej jest wpisywana zawartość *czas.msek* i jest wyświetlana wartość liczników zegara. W ten sposób wskazania zegara na wyświetlaczu LCD są aktualizowane tylko po zmianie wartości licznika sekund. Dodatkowo, przy zmianie liczników z 23:59:59 na 00:00:00 (start nowej doby) jest odczytywany kalendarz i uaktualniania data.

W licznikach liczących w kodzie BCD, 4 starsze bity licznika zawierają dziesiątki, a 4 młodsze jednostki zliczanej wartości. Niemniej, licznik układu zegara umożliwia operowanie jedynie na słowach 8-bitowych. Dlatego aby wyświetlić dziesiątki trzeba wartość słowa przesunąć o 4 bity w prawo. Wówczas na 4 młodszych bitach otrzymujemy wartość binarną cyfry dziesiątek. Pozostaje jeszcze jej konwersja na wartość kodu ASCII wyświetlanego przez wyświetlacz LCD. Znak „0" (zero) ma kod ASCII równy 0x30, znak „1" kod 0x31 itd. Co naturalne, konwersja polega na dodaniu do wartości binarnej stałej równej 0x30. Dziesiątki godzin są wyświetlane za pomocą funkcją *zapis_rd(((czas.godz&0x30)>>4)+0x30)*. Dodatkowe maskowanie *czas.godz&0x30* (zerowanie bitów 7, 6 i 3…0) wynika z tego, że w liczniku godzin 2 najstarsze bity są wykorzystywane do innych celów, natomiast najmłodsze 4 bity nie są brane pod uwagę.

Listing 7. Funkcja wyświetlania dnia tygodnia

```
void DispDay(char x, char y){
  PosLcd(x,y);
  switch (czas.dni&0x07){
    case 1:
      DispLcd("Pn");
      break;
    case 2:
      DispLcd("Wt");
      break;
    case 3:
      DispLcd("Sr");
      break;
    case 4:
      DispLcd("Cz");
      break;
    case 5:
      DispLcd("Pt");
      break;
    case 6:
      DispLcd("So");
      break;
    case 7:
      DispLcd("Nd");
      break;
  }
}
```

Listing 8. Wyświetlanie dnia i nazwy miesiąca

```
void DispMies(char x, char y){
  PosLcd(x,y);
  WriteRd(((czas.data&0x30)>>4)+0x30);                    //dziesiątki dni miesiąca
  WriteRd((czas.data&0x0f)+0x30);                         //jednostki dni miesiąca
  WriteRd(' ');
  switch ((((czas.mies&0x10)>>4)*10)+(czas.mies&0x0f)){   //wyświetlanie nazwy miesiąca
    case 1:
      DispLcd("Sty");
      break;
    case 2:
      DispLcd("Lut");
      break;
    case 3:
      DispLcd("Mar");
      break;
    case 4:
      DispLcd("Kwi");
      break;
    case 5:
      DispLcd("Maj");
      break;
    case 6:
      DispLcd("Cze");
      break;
    case 7:
      DispLcd("Lip");
      break;
    case 8:
      DispLcd("Sie");
      break;
    case 9:
      DispLcd("Wrz");
      break;
    case 10:
      DispLcd("Paz");
      break;
    case 11:
      DispLcd("Lis");
      break;
    case 12:
      DispLcd("Gru");
      break;
  }
}
```

Podobnie, ale bez przesunięcia, w procedurze wyświetlania jednostek są maskowane 4 najstarsze bity i jest dodawana wartość 0x30.

Funkcja *DispTime* wyświetla kolejno zawartość odczytanych z M41T00 liczników zegara tj. godzin, minut i sekund. Wyświetlane wartości są rozdzielone znakami dwukropka.

Nazwę dnia tygodnia wyświetla funkcja *DispDay* pokazana na **listingu 7**. Argumenty *x* i *y* są współrzędnymi nazwy na ekranie wyświetlacza LCD. Przed wywołaniem tej funkcji w zmiennej *czas.dni* powinna być aktualna wartość przeczytana z zegara funkcją *ReadRTC*. Licznik dni tygodnia zlicza w kodzie BCD i zmienia się w zakresie od 1 do 7. Bity D3…D7 nie są znaczące i dlatego wartość zmiennej *czas.dni* jest maskowana za pomocą iloczynu logicznego ze stałą 0x07. Nazwa miesiąca i dzień miesiąca są wyświetlane przez funkcję *DispMies* (**listing 8**) i od-

Listing 9. Fragment pętli ustawiania godzin

```
while(1){
  DispDec(tem,1,2);
  while((PORTA&0x0f)==0x0f);        //klawisz został naciśniety
  delay_ms(30);                     //oczekiwanie na zakończenie drgań
  kl=(PORTA&0x0f);                  //kod klawisza odczytany z linii portu
  if(kl==SW_SW2){                   //zwiększanie licznika godzin
    ++tem;
    if(tem==24) tem=0;              //licznik modulo 24
  }
  if(kl==SW_SW3){                   //zmniejszanie licznika godzin
    --tem;
    if(tem==0xff) tem=23;
  }
  if(kl==SW_SW1) break              //klawisz SW1 - koniec ustawiania
  delay_ms(250);
}
```

Listing 10. Funkcja ustawiania *zegara*

```
void SetCzasRTC(void){
unsigned char kl, tem;
  ClrLcd();
  ReadRTC();                                          //odczytanie RTC
  czas.sek=0x80;                                      //zatrzymanie zegara
  PosLcd(1,1);
  DispLcd("godziny SW1 akc.");                        //wyświetlanie "pomocy"
  PosLcd(8,2);
  DispLcd("+SW2 -SW3");
  PosLcd(3,2);
  WriteRd(':');
  WriteRd('-');WriteRd('-');                          //w pozycji minut 2 kreski
  tem=(((czas.godz&0x30)>>4)*10)+(czas.godz&0x0f);    //binarna liczba godzin
                                                      //pętla ustawiania godzin
  while(1){
    DispDec(tem,1,2);
    while((PORTA&0x0f)==0x0f);                        //klawisz został naciśniety
    delay_ms(30);
    kl=(PORTA&0x0f);
    if(kl==SW_SW2){
      ++tem;
     if(tem==24)
     tem=0;
    }
    if(kl==SW_SW3){
      --tem;
     if(tem==0xff)
     tem=23;
    }
    if(kl==SW_SW1) break;
    delay_ms(350);
  }
  while((PORTA&0x0f)!=0x0f);                          //czekaj na zwolnienie
                                                      //klawisza
  delay_ms(300);
  czas.godz=((tem/10)<<4)|(tem%10);
  PosLcd(1,1);
  DispLcd("minuty  SW1 akc.");
  PosLcd(8,2);
  DispLcd("+SW2 -SW3");
  PosLcd(6,2);
  tem=(((czas.min&0x70)>>4)*10)+(czas.min&0x0f);      //liczba minut binarnie
                                                      //pętla ustawiania minut
  while(1){
    DispDec(tem,4,2);
    while((PORTA&0x0f)==0x0f);                        //klawisz został naciśniety
    delay_ms(30);
    kl=(PORTA&0x0f);
    if(kl==SW_SW2){
      ++tem;
      if(tem==60)
      tem=0;
    }
    if(kl==SW_SW3){
      --tem;
      if(tem==0xff)
      tem=59;
    }
    if(kl==SW_SW1) break;
    delay_ms(350);
  }
  czas.min=((tem/10)<<4)|(tem%10);
  WriteCzasRTC();
  ClrLcd();
  PosLcd(13,1);
  DispLcd("SW0");
  PosLcd(11,2);
  DispLcd("start");
  czas.zmsek=1;
  DispTime(1,2);
                                                      //pętla uruchomienia zegara
  while(1){
    kl=klaw();
    if(kl==SW_SW0)
    break;
  }
  czas.sek=0;                                         //uruchomienie zegara zegar
  WriteCzasRTC();
  ClrLcd();
}
```

Listing 11. Ustawienie *kalendarza*

```
void SetDataRTC(void){
unsigned char kl,tem;
  while((PORTA&0x0f)!=0x0f);                          //czekaj na zwolnienie
                                                      //klawisza
  delay_ms(250);
  ReadRTC();
  ClrLcd();
  PosLcd(1,1);
  DispLcd("dzien tygodnia ");                         //ustawanie dnia tygodnia
  PosLcd(4,2);
  DispLcd("zmSW2 accSW1   ");
  tem=czas.dni&7;
  while(1){
    DispDay(1,2);                                     //wyświetlanie dnia
                                                      //tygodnia
    kl=klaw();
    if(kl==SW_SW2){
      ++tem;
      if(tem==8)
```

powiadają wartościom zmiennych *czas.data* i *czas.mies* odczytanych za pomocą funkcji *ReadRTC*. Argumenty *x* i *y* są współrzędnymi daty na ekranie LCD.

Każdy zegar powinien być wyposażony w funkcje nastaw wskazań godziny i daty. Najczęściej do tego celu wykorzystuje się klawiaturę składającą się z kilku przycisków. W naszym rozwiązaniu do ustawiania czasu i daty użyjemy przycisków SW0…SW3 modułu AVT5275. Zwierają one do masy linie SW0…SW3 złącza J17 podciągane do plusa zasilania rezystorami R18…R21. Linie SW0…SW3 złącza J17 należy połączyć odpowiednio z liniami RA0…RA3 złącza J13 portu PORTA.

Odczytywanie stanu klawiszy może być wykonywane na 2 sposoby. Pierwszy z nich czeka na zwolnienie klawisza, a potem na jego przyciśnięcie. Kod naciśnietego klawisza jest zwracany po każdym przyciśnięciu i zwolnieniu. Użycie tego sposobu do ustawiania na przykład minut jest możliwe, ale trochę kłopotliwe, bo wymaga wielokrotnego naciskania klawisza.

Drugi sposób polega na wykryciu naciśnięcia i przytrzymania klawisza. Kod klawisza jest generowany w odstępach czasu ustalonych opóźnieniem. Na **listingu 9** pokazano fragment nieskończonej pętli ustawiania zegara wykorzystujący ten sposób. Instrukcja *while((PORTA&0x0f)==0x0f);* czeka na przyciśnięcie dowolnego klawisza (wymuszenie stanu niskiego na jednej z linii PA0…PA3). Kiedy na jednej z linii pojawi się poziom niski, jest odmierzane opóźnienie likwidujące drgania styków i do zmiennej *kl* jest przepisywany stan 4 młodszych bitów rejestru portu PORTA. Zależnie od przyciśniętego klawisza, w pętli jest wykonywana modyfikacja parametrów, ale musi być wyróżniony pojedynczy klawisz, którego naciśnięcie powoduje wyjście z nieskończonej pętli *while(1)* instrukcją *break*. Dobranie opóźnienia w pętli (tutaj *delay_ms(250)*) umożliwia szybszą lub wolniejszą zmianę ustawianych parametrów. Szybsza zmiana jest wygodniejsza, jeżeli zakres zmian jest większy (ma przykład ustawianie minut). Większe opóźnienie jest lepsze w wypadku mniejszych zakresów, na przykład przy ustawianiu godzin. Opóźnienie można dobrać eksperymentalnie, tak aby ustawianie parametrów było najwygodniejsze.

Funkcje wykorzystujące klawiaturę i wyświetlacz do ustawiania parametrów nie są zbyt skomplikowane, ale zazwyczaj zajmują sporo zasobów mikrokontrolera w porównaniu na przykład z funkcjami komunikacji poprzez interfejs I²C. Na **listingu 11** zamieszczono funkcję ustawiania zegara i kalendarza w układzie M41T00. Przed wywołaniem funkcji nastaw są odczytywane bieżąca godzina i data, zapisywane do struktury *czas*,

i te wartości są modyfikowane. Za pomocą funkcji *SetCzasRTC* ustawia się liczniki minut i godzin, a licznik sekund jest zerowany. W zmiennej *czas.sek* jest ustawiany najstarszy bit. Dzięki temu po jej zapisaniu do rejestrów zegara zostanie zatrzymany układ oscylatora i liczniki układu zegara zatrzymają się.

Żeby funkcje ustawiania mogły być używane nie tylko przez autora programu, warto na ekranie wyświetlacza wyświetlić informacje pomagające wykonać potrzebne czynności bez dodatkowej instrukcji. Podczas ustawiania godzin górnej linijce wyświetlacza pojawia się napis „godziny SW1 akc" informujący, że ustawiane są godziny, a przyciśnięcie klawisza SW1 spowoduje zaakceptowanie wartości. W dolnej linijce są wyświetlane odczytane z zegara godziny, w miejscu minut dwie kreski, oraz napis „+SW2 –SW3". Naciśnięcie klawisza SW2 spowoduje zwiększanie licznika godzin, a klawisza SW3 jego zmniejszanie.

Przed wejściem w pętlę ustawiania godzin, zawartość licznika godzin jest konwertowana z kodu BCD na wartość binarną i zapisywana w zmiennej lokalnej *tem*. Binarna wartość *tem* będzie modyfikowana, a po zaakceptowaniu ponownie przekonwertowana na BCD i wpisana do zegara.

Działanie pętli ustawiania zostało już dokładniej opisane wcześniej (list. 9). Przyciśniecie klawisza SW2 powoduje zwiększenie zmiennej *tem*. Po osiągnięciu wartości 24 zmienna *tem* jest zerowana. Przyciśnięcie klawisza SW3 dekrementuje zmienną *tem*. Po osiągnięciu wartości 0xFF do *tem* jest wpisywana wartość 23. Ustawienie licznika minut przebiega podobnie, jak ustawienie licznika godzin. Inne są oczywiście wyświetlane komunikaty pomocy, ale funkcje klawiszy i sposób zmiany licznika jest taki sam w przypadku minut modulo 60).

Po ustawieniu i zaakceptowaniu minut liczniki są zapisywane do M41T00 przez funkcję WriteCzasRTC. Ponieważ wcześniej w liczniku sekund został ustawiony najstarszy bit, to od tego momentu oscylator jest zatrzymany (a tym samym – zliczanie czasu). Po naciśnięciu dowolnego klawisza zegar zostaje uruchomiony przez wpisanie do licznika sekund samych zer łącznie z najstarszym bitem.

Na **listingu 12** pokazano funkcję ustawiania daty: dnia tygodnia, dnia miesiąca i nazwy miesiąca (**fotografia 6**). Dzień tygodnia jest zmieniany przez naciskanie i zwalnianie klawisza SW2. Użyto tu funkcji odczytywania stanu klawiszy *Klaw*. Naciśnięcie klawisza SW0 akceptuje ustawioną wartość. Dzień i nazwa miesiąca są ustawiane w pętlach, tak jak w funkcji ustawiania czasu.

Tomasz Jabłoński, EP

Listing 11. c.d.

```
      tem=1;
      czas.dni=tem;
    }
    if(kl==SW_SW1)
    break;
  }
  delay_ms(250);
  while((PORTA&0x0f)!=0x0f);                    //czekaj na zwolnienie
                                                //klawisza
  delay_ms(250);
  ClrLcd();
  PosLcd(1,1);
  DispLcd("dzien miesiaca ");                   //ustawianie dnia miesiąca
  PosLcd(8,2);
  DispLcd("+SW2 -SW3");
  tem=(((czas.data&0x70)>>4)*10)+(czas.data&0x0f); //dzień miesiąca binarnie
  while(1){
    DispDec(tem,2,2);
    while((PORTA&0x0f)==0x0f);                  //klawisz został
                                                //naciśniety
    delay_ms(30);
    kl=(PORTA&0x0f);
    if(kl==SW_SW2){
      ++tem;
      if(tem==32)
      tem=1;
    }
    if(kl==SW_SW3){
      --tem;
      if(tem==0)
      tem=31;
    }
    if(kl==SW_SW1)
    break;
    delay_ms(250);
  }
  delay_ms(250);
  while((PORTA&0x0f)!=0x0f);                    //czekaj na zwolnienie
                                                //klawisza
  delay_ms(250);
  czas.data=((tem/10)<<4)|(tem%10);
  ClrLcd();
  PosLcd(1,1);
  DispLcd("miesiac ");                          //ustawianie dnia miesiąca
  PosLcd(8,2);
  DispLcd("+SW2 -SW3");
  tem=(((czas.mies&0x70)>>4)*10)+(czas.mies&0x0f); //dzień miesiąca  binarnie
  while(1){
    DispMies(1,2);
    while((PORTA&0x0f)==0x0f);                  //klawisz został
                                                //naciśniety
    delay_ms(30);
    kl=(PORTA&0x0f);
    if(kl==SW_SW2){
      ++tem;
      if(tem==13)
      tem=1;
    }
    if(kl==SW_SW3){
      --tem;
      if(tem==0)
      tem=12;
    }
    if(kl==SW_SW1)
    break;
    delay_ms(250);
    czas.mies=((tem/10)<<4)|(tem%10);
  }
  while((PORTA&0x0f)!=0x0f);                    //czekaj na zwolnienie
                                                //klawisza
  delay_ms(250);
  WriteDataRTC();
  ClrLcd();
}
```

Listing 12. Funkcje pomocnicze wyświetlania liczby dziesiętnej i obsługi klawiatury

```
                                        //wyświetlenie wartości data
                                        //w postaci 2 cyfr dziesiętnie
                                        //00hex....99hex
void DispDec(int8 data, int8 pos){
  PosLcd(pos);
  zapis_rd((data/10)+0x30);
  zapis_rd((data%10)+0x30);
}

                                        //obsługa klawiatury
int8 klaw(void){
  int8 kl;
  while((input_a()&0x0f)==0x0f)         //czekaj na przyciśnięcie klawisza
  kl=input_a()&0x0f;
  delay_ms(30);
  kl=input_a()&0x0f;
  while((input_a()&0x0f)!=0x0f);        //czekaj na zwolnienie klawisza
  return(kl);
}
```

Fotografia 6. Ustawianie miesiąca