LITEcomp – aplikacje Termometr z DS18B20

Znowu termometr na DS18B20. Tak, z odrobiną rezygnacji, zareagowała zapewne część Czytelników na widok samego tytułu. Nie przerzucajmy jednak machinalnie kartek dalej. Mimo oklepanego tematu, artykuł wart jest przeczytania.

st7.ep.com.pl

Głównym zamiarem autora tego artykułu było zaprezentowanie praktycznej aplikacji komputerka LITEcomp z mikrokontrolerem ST7FLITE19. Mimo, że projektów dotyczących termometrów z układem DS18B20 było już tyle, że publikowanie kolejnego wydawało się tylko „odgrzewaniem kotleta", to nieco odmienne od najczęściej stosowanego podejście do tematu stwarzało szansę na przygotowanie ciekawego materiału.

W zdecydowanej większości wcześniej opisywane termometry były zbudowane w oparciu o mikrokontrolery AVR i oprogramowane w Bascomie. Nie spotkałem się natomiast z projektem takiego termometru na mikrokontrolerze z rodziny ST7, co tłumaczy chyba tylko brak kompilatora Bascom dla tychże mikrokontrolerów. Stojąc więc przed wyborem aplikacji dla LITEcomp zdecydowałem się w pierwszej kolejności zaprezentować właśnie termometr z układem DS18B20. Pozwoli to na poruszenie dwóch, z pozoru trudnych, zagadnień: obsługi magistrali 1–Wire oraz alfanumerycznego wyświetlacza LCD w asemblerze ST7. Wyświetlacz LCD jest podstawowym elementem większości obecnych układów mikroprocesorowych, natomiast magistrala 1–Wire umożliwia komunikację z wieloma interesującymi układami scalonymi, jak chociażby dobrze znanym wszystkim termometrem DS18B20. Mikrokontrolery ST7 dopiero zdobywają popularność wśród hobbystów w Polsce, więc wymienione wyżej zagadnienia nie są wystarczająco dobrze opracowane i znane użytkownikom, jak to ma miejsce np. w przypadku mikrokontrolerów AVR.

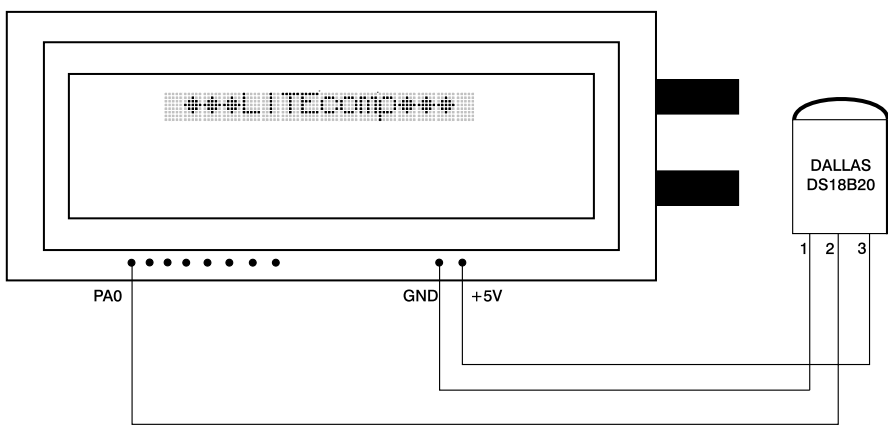
Bazą termometru jest moduł LITEcomp z dołączonym wyświetlaczem alfanumerycznym LCD. Jedynym elementem zewnętrznym jest układ DS18B20, który dołączono do linii PA0. Do linii tej podłączony jest też przycisk S1 (nie będzie on wykorzy-

LITEcomp

LITEcomp jest prostym komputerkiem wykonanym na mikrokontrolerze ST7FLITE19. LITEcomp jest w ramach promocji dodawany bezpłatnie do książki „Mikrokontrolery ST7LITE w praktyce" (autor Jacek Bogusz). Książka jest dostępna w *sklep.avt.pl* (numer katalogowy KS–260905).

Dla praktyków

Zachęcamy Czytelników do przysyłania własnych opracowań na komputerku LITEcomp. Najlepsze opisy opublikujemy w EP, wszystkie nadesłane projekty uhonorujemy interesującymi nagrodami (książki, zestawy uruchomieniowe, narzędzia – wykaz opublikujemy za miesiąc).



Rys. 1. Schemat dołączenia układu DS18B20 do modułu LITEcomp

List. 1. Procedura zapisu instrukcji/danej do wyświetlacza

```
.writetolcd
PUSH A              ;zapamiętania akumulatora na stosie

OR   A, #$0F        ;ustawienie 4 młodszych bitów akumulatora
LD   tmp, A         ;zapamiętanie zawartośći akumulatora w zmiennej

LD   A,PADR         ;odczyt zawartosci rejestru PADR
OR   A,#$F0         ;ustawienie 4 starszych bitów
AND  A,tmp          ;iloczyn logiczny akumulatora i zmiennej
LD   PADR,A         ;zapis zawartości akumulatora do PADR

BSET PADR,#EN
NOP
BRES PADR,#EN       ;impuls na linii EN
POP  A              ;zdejmujemy uprzednio zapamiętany bajt ze stosu
SWAP A              ;i zamieniamy jego połówki

OR   A,#$0F         ;ustawienie 4 młodszych bitów
LD   tmp,A          ;zapamietanie A w zmiennej tmp

LD   A,PADR         ;załadowanie do A stanu portu
OR   A,#$F0         ;ustawienie 4 starszych bitów
AND  A,tmp          ;iloczyn logiczny ze zmienną tmp
LD   PADR,A         ;zapisanie stanu portu

BSET PADR,#EN       ;
NOP
BRES PADR,#EN       ;impuls na linii EN
CALL waitlcd        ;oczekiwanie na zakończenie wykonywania instrukcji
RET
```

stywany w tym przykładzie) oraz rezystor podciągający 4,7 kΩ, stosowanie zewnętrznego rezystora podciągającego nie jest więc konieczne. Schemat dołączenia układu DS18B20 do modułu LITEcomp przedstawiono na **rys. 1**.

Obsługa wyświetlacza LCD

Wyświetlacz LCD dołączony do komputerka LITEcomp pracuje w trybie 4–bitowym z wyprowadzeniem R/W na stałe połączonym do masy. W związku z tym, nie jest możliwe sprawdzanie flagi zajętości wyświetlacza, czego konsekwencją jest konieczność generowania programowych opóźnień pomiędzy kolejnymi operacjami zapisu danych do wyświetlacza.

Przyjęcie 4–bitowego trybu pracy wyświetlacza komplikuje nieco procedurę zapisu bajtu do niego. Ponadto z uwagi na to, że do portu A mikrokontrolera oprócz wyświetlacza dołączony jest również układ DS18B20, należy zadbać, aby podczas zapisu danych do wyświetlacza nie ulegał zmianie stan na pozostałych wyprowadzeniach. Jedyną różnicą pomiędzy zapisem instrukcji, a zapisem danej, jest stan panujący na linii RS. Zdecydowałem się wyodrębnić wspólną część kodu w postaci procedury *writetolcd* (**list. 1**).

Powyższy kod nie precyzuje typu operacji (czy zapisujemy daną, czy rozkaz), tak więc konieczne jest przygotowanie dwóch kolejnych procedur, z których każda będzie wywoływała procedurę *writetolcd*. Jedyną czynnością dodatkową będzie ustawienie odpowiedniego stanu na linii RS. Zapis rozkazu następuje przy niskim stanie na linii RS (**list. 2)**, zapis danej następuje przy wysokim stanie na linii RS (**list. 3**).

Wyświetlenie łańcucha znaków

W celu wyświetlenia napisów przygotowałem makroinstrukcję realizującą funkcję wyświetlenia na wyświetlaczu ciągu znaków zakończonego zerem (**list. 4**). Argumentem tej makroinstrukcji jest adres pierwszego znaku napisu.

Procedury obsługi magistrali 1–wire

Układ DS18B20 jest dołączony do wyprowadzenia PA0. Ustalanie odpowiednich stanów na wyprowadzeniu, a tym samym na magistrali, odbywa się poprzez modyfikację bitu w rejestrze kierunku (PADDR), a nie jak można było się spodziewać, w rejestrze danych (PADR). Jest to podyktowane tym, że zgodnie ze specyfikacją magistrali, stan wysoki jest wymuszany poprzez rezystor podciągający magistralę do linii zasilania, a nie jak to zwykle ma miejsce poprzez wyjściowy tranzystor portu. Tak więc mikrokontroler w celu wystawienia stanu wysokiego na magistralę musi ustawić linię wyjściową w stan wysokiej impedancji (dodatkowo podczas trwania stanu wysokiego układ master musi mieć możliwość odczytania stanu panującego na magistrali). Zarówno stan wysokiej impedancji, jak i tryb wejściowy uaktywniany jest poprzez wyzerowanie bitu w rejestrze kierunku (przy wyzerowanym bicie w rejestrze danych). Natomiast w celu wystawienia stanu niskiego konieczne jest zwarcie magistrali do masy poprzez tranzystor wyjściowy portu (wyprowadzenie już nie musi pracować w trybie wejściowym, tak więc zrealizujemy to poprzez zapis jedynki do rejestru kierunku). Należy jednak zadbać o to, aby w rejestrze danych bit odpowiadający wyprowadzeniu wykorzystywanemu do obsługi magistrali 1–wire był zawsze wyzerowany. W przeciwnym razie opisany wyżej mechanizm nie zadziała.

Procedura opóźniająca

Zadaniem pierwszej procedury (**list. 5**) jest generowanie opóźnienia wymaganego przez specyfikację magistrali 1–wire. Wartość opóźnienia należy przed wywołaniem procedu-

List. 2. Procedura zapisu rozkazu do wyświetlacza

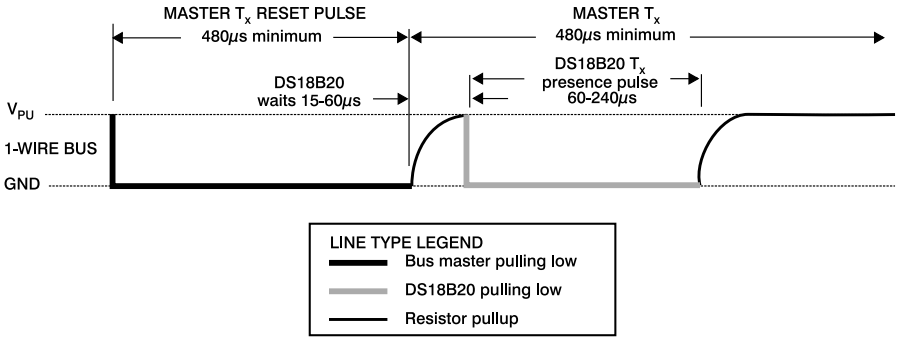
```
.writecommand
BRES PADR,#RS         ;niski stan na linii RS -> zapis rozkazu
CALL writetolcd
RET
```

List. 3. Procedura zapisu danej do wyświetlacza

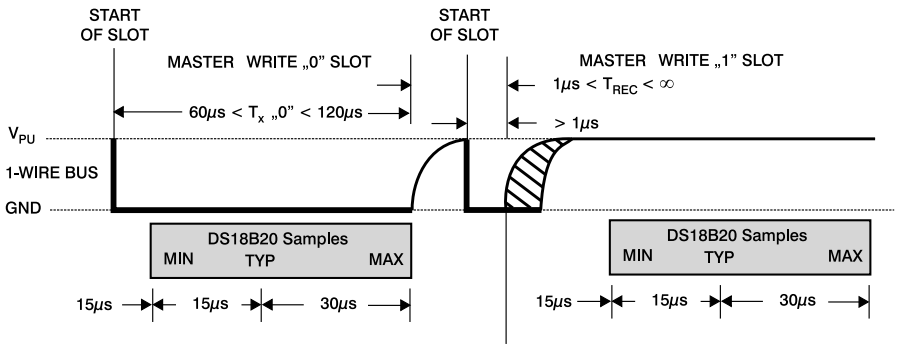
```
.writedata
BSET PADR,#RS         ;wysoki stan na linii RS -> zapis danej
CALL   writetolcd
RET
```

List. 4. Makroinstrukcja wyświetlająca ciąg znaków zakończony zerem

```
writetext MACRO pstr
  LOCAL wt1,wt2
  CLR   X             ; wyzerowanie rejestru x
wt1
  LD A, (pstr,X)      ; załadowanie do A znaku spod adresu będącego sumą
                      ; pstr i rejestru X
  JREQ  wt2           ; jeśli odczytany znak jest zerem (koniec napisu) to
skok do wt2
  INC   X             ; inkrementacja rejestru X (przygotowanie do pobrania
                      ; następnego znaku)
  CALL  writedata     ; wyświetlenie znaku na wyświetlaczu
  JRA   wt1           ; skok do wt1
wt2
  MEND
```



Rys. 2. Przebiegi czasowe sygnału RESET oraz impulsu obecności



Rys. 3. Przebiegi czasowe podczas zapisu bitu na magistralę 1–wire

ry umieścić w rejestrze A. Liczba zapisana do rejestru A w przybliżeniu odpowiada czasowi opóźnienia określonemu w mikrosekundach.

Wysłanie sygnału RESET

Każda transmisja magistralą 1–wire jest rozpoczynana od wysłania przez układ nadrzędny sygnału RESET. W odpowiedzi na ten sygnał układ podrzędny wysyła impuls obecności informujący układ nadrzędny o fakcie podłączenia co najmniej jednego układu do magistrali 1–wire. Odebranie przez procedurę sygnału obecności sygnalizowane jest wyzerowaniem flagi przeniesienia C, natomiast nieodebranie sygnału obecności spowoduje jej ustawienie. Przebiegi czasowe sygnału RESET oraz impulsu obecności przedstawione są na **rys. 2**.

Z przedstawionych przebiegów wynika, że wysłanie sygnału RESET polega na wystawieniu na magistralę stanu niskiego na okres min. 480 µs i zwolnieniu magistrali na okres ok. 65 µs, po którym układ podrzędny wystawi impuls obecności (stan niski) trwający od 60 do 240 µs. Procedurę wysyłającą sygnał RESET przedstawiono na **list. 6.**

List. 5. Procedura odmierzania opóźnienia

```
; opóźnienie o A + 2,5 us dla
f=16MHz
.delay
  NOP
  DEC  A
  JRNE delay
  RET
```

Zapis bitu

Przebiegi czasowe podczas zapisu bitu na magistralę przedstawia **rys. 3**. Zapis bitu o wartości jedynki logicznej polega na wymuszeniu przez układ nadrzędny stanu niskiego na okres maksymalnie 15 µs, a następnie zwolnieniu magistrali. Zapis bitu o wartości zera logicznego polega na wymuszeniu stanu niskiego na magistrali na okres co najmniej 60 µs. Na **list. 7** przedstawiono procedurę realizującą zapis bitu.

Odczyt bitu

Przebiegi czasowe podczas odczytu bitu z magistrali przedstawiono na **rys. 4**. Odczyt bitu z układu podrzędnego inicjowany jest przez układ nadrzędny wymuszeniem stanu niskiego na magistrali na okres 1 µs, a następnie zwolnieniem magistrali. Przed upływem 15 µs układ nadrzędny dokonuje sprawdzenia stanu magistrali, który odzwierciedla stan transmitowanego bitu. Procedurę realizującą odczyt bitu przedstawiono na **list. 8**.

Zapis bajtu

Procedura zapisu bajtu (**list. 9**) polega na ośmiokrotnym wywołaniu procedury zapisu pojedynczego bitu z uprzednim ustawieniem wskaźnika C na odpowiednią (zależną od wartości konkretnego bitu w bajcie) wartość.

Odczyt bajtu

Procedura odczytu bajtu (**list. 10**), podobnie jak zapisu, polega na ośmiokrotnym wywołaniu procedury od-

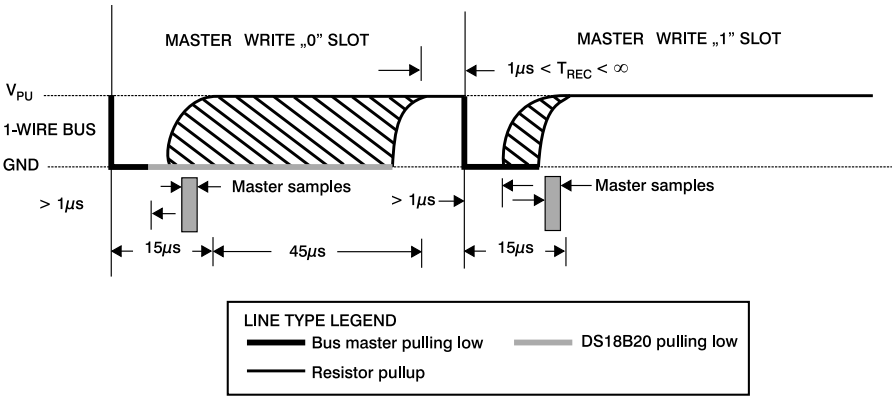
List. 6. Procedura wysyłająca sygnał RESET

```
.owreset
  BSET  PADDR,#DQ     ; wymuszamy stan niski na linii DQ
  LD A,#240           ;
  CALL  delay         ; | czekamy ok 480 <m>s
  LD A,#240           ; |
  CALL  delay
  BRES  PADDR,#DQ     ;zwalniamy linię DQ
  LD A,#110           ; czekamy ok 110 µs
  CALL  delay

  SCF                 ; ustawiamy wskaźnik C
  BTJT  PADR,#DQ,owr1 ; jeśli linia DQ jest w stanie
                      ; wysokim skaczemy do owr1
  RCF                 ; w przeciwnym razie zerujemy wskaźnik C
owr1
  LD A,#180           ;
  CALL  delay         ;czekamy do końca slotu
  LD A,#180           ;
  CALL  delay         ;
  RET
```

List. 7. Procedura realizująca zapis bitu

```
.owwritebit
  BSET  PADDR,#DQ     ; wymuszamy na linii DQ stan niski
  LD A,#13            ;
  CALL  delay         ; czekamy ok 15 µs
  JRNC  owb1          ; jeśli wskaźnik C jest wyzerowany
                      ; to skaczemy do owb1
  BRES  PADDR,#DQ     ; w przeciwnym razie zwalniamy linię DQ
owb1
  LD A,#100           ;
  CALL  delay         ; czekamy ok. 100 µs
  BRES  PADDR,#DQ     ; zwalniamy linię DQ
  LD A,#100           ;
  CALL  delay         ; czekamy do końca slotu
  RET
```



Rys. 4. Przebiegi czasowe podczas odczytu bitu z magistrali

List. 8. Procedura realizująca odczyt bitu

```
.owreadbit
  BSET PADDR,#DQ        ; wymuszamy na linii DQ stan niski
  NOP                   ; czekamy nie dłużej niż 1 µs
  BRES PADDR,#DQ        ; zwalniamy linię DQ
  LD A,#10              ;
  CALL delay            ; czekamy ok 12 µs
  SCF                   ; ustawiamy wskaźnik przeniesienia
  BTJT PADR,#DQ,owrb1   ; jeśli linia DQ jest w stanie wysokim
                        ; to SLAVE nadaje "1"
  RCF                   ; w przeciwnym razie nadaje "0" i zerujemy wskaźnik C
.owrb1
  LD A,#100             ; czekamy do końca slotu
  CALL delay
  RET
```

czytu pojedynczego bitu. Po każdym odczycie bitu wskaźnik C jest przesuwany do rejestru A. Po ośmiokrotnym wykonaniu tej operacji w rejestrze A (oraz w zmiennej *owdata*) znajduje się odebrany z układu podrzędnego bajt.

Odczyt temperatury z układu DS18B20

Procedury odczytu temperatury z układu DS18B20 nie trzeba dokładnie omawiać, bowiem na ten temat napisano już chyba wszystko, co tylko się da. Dokładny opis algorytmu odczytu wyniku pomiaru temperatury jest zawarty w nocie katalogowej DS18B20, więc zainteresowanych odsyłam do lektury tego dokumentu. Trochę miejsca chciałbym natomiast poświęcić zagadnieniu przekształcenia dwubajtowego wyniku pomiaru na liczbę będącą wartością temperatury w stopniach Celsjusza. Dokładny opis formatu poszczególnych bajtów wyniku zawarty jest w nocie katalogowej układu DS18B20 i nie będzie tutaj powielony. Podczas pisania programu przyjąłem pewne założenie, które pozwoli na szybkie i bezproblemowe przekształcenie wyniku konwersji na oczekiwaną postać. Mianowicie zrezygnowałem z wyświetlania części ułamkowej temperatury (układ jest termometrem pokojowym, a nie lekarskim lub laboratoryjnym) oraz z pomiaru temperatur ujemnych (w pokoju zazwyczaj temperatura nie spada poniżej zera). Układ DS18B20 jako wynik konwersji temperatury zwraca dwa bajty – młodszy (LSB) oraz starszy (MSB). Przy uwzględnieniu założeń, o których wspomniałem, cały proces przekształcenia wyniku można opisać równaniem:

$$\text{Temperatura} = (\text{LSB}/16) + (\text{MSB}*16)$$

Warto zauważyć, że zarówno mnożenie, jak i dzielenie odbywa się z udziałem liczby będącej potęgą liczby 2, zamiast czasochłonnych operacji mnożenia i dzielenia (mikrokontrolery ST7 nie posiadają instrukcji dzielenia, co dodatkowo spowolniłoby proces konwersji) można więc zastosować operację przesunięcia w lewo (dla mnożenia) i w prawo (dla dzielenia). Sprawę można jeszcze bardziej uprościć – zarówno dzielenie, jak i mnożenie przez 16 można zastąpić (z pewnymi ograniczeniami) instrukcją... SWAP (czyli zamianą połówek bajtu). Jest to możliwe dlatego, że w przypadku mnożenia mnożony przez 16 czynnik nigdy nie przekroczy wartości 15 (więc wynik nie wyjdzie poza zakres 8–bitowej liczby). Dzięki temu, po zamianie połówek bajtu starszy półbajt jest wartością młodszego półbajtu sprzed wykonania operacji pomnożoną przez 16, natomiast młodszy półbajt zawiera same zera (nie „zakłóca" wyniku). Natomiast w przypadku „dzielenia" za pomocą instrukcji SWAP młodszy półbajt sprzed wykonania operacji trafia na miejsce starszego półbajtu „zakłócając" wynik i dlatego należy po wykonaniu operacji wyzerować starszy półbajt instrukcją AND A, #$0F. Procedurę pomiaru temperatury przedstawiono na **list. 11**.

List. 9. Procedura zapisu bajtu

```
.owwrite
  LD owdata,A           ; zapamiętujemy daną do zapisania w owdata
  LD X,#8               ; ilość powtórzeń pętli
.ow1
  LD A,owdata           ; przywracamy daną do zapisania
  RRC  A                ; obrót akumulatora w prawo (LS bit do C)
  LD owdata,A           ; zapamiętujemy obrócony A w owdata
  CALL owwritebit       ; wysyłamy bit na magistralę
  DEC  X                ; zmniejszamy licznik pętli
  JRNE ow1              ; jesli większy od zera skaczemy na początek pętli
  RET
```

List. 10. Procedura odczytu bajtu

```
.owread
  LD A,#0               ; zerujemy rejestr A i zmieną owdata
  LD owdata,A
  LD X,#8               ; ilośc powtórzeń pętli
or1
  CALL owreadbit        ; odczutyjemy jeden bit
  LD A,owdata           ; owdata do A
  RRC  A                ; obrót A w prawo (C na pozycję MS bitu)
  LD owdata,A           ; a do owdata
  DEC  X                ; zmniejszamy licznik pętli
  JRNE or1              ; jeśli niezerowy to skocz na początek pętli
  RET
```

Obliczenie minimalnej i maksymalnej temperatury

Przydatną funkcją termometru jest funkcja „min–max" umożliwiająca wyświetlenie minimalnej i maksymalnej temperatury osiągniętej w czasie pomiaru. Jest ona przedstawiona na **list. 12**.

Wyświetlenie liczby w systemie dziesiętnym

Gdy wynik pomiaru temperatury odczytany z układu DS18B20 zostanie przetworzony na wartość odpowiadającą zmierzonej temperaturze

List. 11. Procedura pomiaru temperatury

```
; Procedura pomiaru temperatury
; Temperatura zostaje umieszczona w zmiennej temper
; oraz pozostaje w rejestrze A
.gettemp
  BRES PADR,#0
                    ; pomiar temperatury
  CALL owreset      ; reset magistrali 1wire
  JRC  mainloop     ; sprawdzenie czy DS18B20 jest podłączony
                    ; jeśli nie, to skok na początek pętli głównej
  LD A,#$CC
  CALL owwrite      ; zapis komendy SKIP ROM
  LD A,#$44
  CALL owwrite      ; zapis komendy CONVERT

                    ; oczekiwanie na zakończenie pomiaru
  LD X,#255
loopwait1
  CALL waitlcd
  DEC  X
  JRNE loopwait1
                    ; / oczekiwanie na zakończenie pomiaru

  CALL owreset      ; reset magistrali 1wire
  LD A,#$CC
  CALL owwrite      ; zapis komendy SKIP ROM
  LD A,#$BE
  CALL owwrite      ; zapis komendy READ SCRATCHPAD

  CALL owread
  LD tmplsb,A       ; odczyt młodszego bajtu wyniku pomiaru
  CALL owread
  LD tmpmsb,A       ; odczyt starszego bajtu wyniku pomiaru
                    ; /pomiar temperatury

                    ; konwersja wyniku pomiaru na wartość w stopniach
  LD A,tmplsb
  SWAP A            ; młodszy bajt dzielimy przez 16
  AND  A,#$0F       ; zerujemy starszą połówkę bajtu
  SWAP tmpmsb       ; starszy bajt mnozymy przez 16
  ADD  A,tmpmsb     ; dodajemy uprzednio przetworzone bajty do siebie
  LD temper,A       ; wynik zapisujemy do zmiennej temper
                    ; / konwersja wyniku pomiaru na wartość w stopniach
  RET
```

pozostaje nam tylko ją wyświetlić na wyświetlaczu. Możliwości konwersji na postać dziesiętną jest kilka. Ja wybrałem najprostszą i najbardziej, moim zdaniem, uniwersalną. Postąpiłem zgodnie z ogólnie znaną zasadą zamiany liczby na dowolny inny system liczbowy, czyli poprzez dzielenie liczby przez podstawę systemu, aż do wyzerowania się wyniku. Sposób ten pozwala na zamianę dowolnie dużych liczb (oczywiście ograniczonych długością liczb obsługiwanych przez procedury dzielenia). Algorytm ten zwraca wynik „od końca" – pierwszą otrzymaną cyfrą będzie cyfra jednostek, kolejną dziesiątek itd., tak więc konieczne będzie wyświetlenie cyfr w odwrotnej kolejności. Zostało to zrealizowane poprzez odkładanie na stos otrzymywanych w wyniku działania procedury cyfr, aby zaraz po zakończeniu konwersji można było je zdjąć ze stosu (i wyświetlić na wyświetlaczu) już we właściwej kolejności. Do zrealizowania dzielenia dwóch liczb 8–bitowych wykorzystałem procedurę pochodzącą z noty aplikacyjnej „ST7 math utility routines" umieszczonej na stronie *http://www.stmcu.com*. Korzystając z tej procedury należy pamiętać o wyzerowaniu wskaźnika przeniesienia bezpośrednio przed jej wywołaniem. W przeciwnym razie „pozostałości" po uprzednio wykonywanych operacjach mogą wprowadzić w błąd procedurę dzielenia, która może potraktować niezerowe argumenty jako zerowe i zwrócić w wyniku zero (co może być przyczyną powstawania uciążliwych błędów, o czym sam miałem okazję się przekonać podczas pisania tego programu). Procedurę konwersji dziesiętnej przedstawiono na **list. 13**.

Omówiłem już wszystkie niezbędne procedury, które posłużą do stworzenia programu pełniącego funkcję termometru pokojowego. Pełny listing programu zamieścimy na płcie CD–EP6/2007B, można go też pobrać ze strony *st7.ep.com.pl*.

Program termometru zajmuje nieco ponad 600 bajtów pamięci programu oraz 13 bajtów pamięci danych. Jest to niewielka część dostępnych zasobów mikrokontrolera ST7FLITE19, tak więc pozostaje jeszcze sporo miejsca na dodatkowe funkcje, jak chociażby sterowanie urządzeniami zewnętrznymi w zależności od temperatury. Niewielkim nakładem pracy układ można dostosować do pełnienia funkcji np. termostatu.

Radosław Kwiecień, EP
radoslaw.kwiecien@ep.com.pl

List. 12. Funkcja „min–max" umożliwiająca wyświetlenie minimalnej i maksymalnej temperatury

```
.getminmax
    ; obliczenie temperatury minimalnej i maksymalnej
  LD X,temper   ; do X ładujemy aktualną temperaturę
  CP X,mintemp  ; porównujemy z wartością temperetury minimalnej
  JRPL lminta   ; jeśli temper > mintemp, skok do etyiety lminta
  LD mintemp,X  ; w przeciwnym razie uaktualniamy mintemp
lminta
  LD X,temper   ; do X ładujemy aktualną temperaturę
  CP X,maxtemp  ; porównujemy z wartością temperatury maksymalnej
  JRMI lmaxta   ; jeśli temper < maxtemp, skok do etykiety lmaxta
  LD maxtemp,X  ; w przeciwnym razie uaktualniamy maxtemp
lmaxta
    ; /obliczenie temperatury minimalnej i maksymalnej
  RET
```

List. 13. Procedura konwersji dziesiętnej

```
.writedecimal
  CLR  cnt
wd1
  LD number_a,A         ;
  LD A,#10              ;
  LD number_b,A         ; dzielimy przez 10
  RCF                   ; wyzerowanie flagi przeniesienia
  CALL div_bxb          ;
  LD A,remaind          ; reszta z dzielenia do A
  ADD  A,#'0'           ; dodajemy kod znaku ,zero'
  PUSH A                ; wysyłamy na stos
  INC  cnt              ; zwiększamy licznik
  LD A,result           ; wynik z dzielenia do A
  ADD  A,#0             ; dodajemy liczbę zero (zawartość się nie
                        ; zmienia, flagi są modyfikowane)
  JRNE wd1              ; jeśli wynik niezerowy (konwersja
                        ; niedokończona) to skok do wd1
wd2                     ; w przeciwnym razie konwersja jest ukończona
  POP  A                ; zdejmujemy znak ze stosu
  CALL writedata        ; i wyświetlamy go na wyświetlaczu
  DEC  cnt              ; zmniejszamy licznik
  JRNE wd2              ; jeśli niezerowy to skok do wd2
  RET
```