LITEcomp – aplikacje
Jednokanałowy woltomierz z pamięcią wyników pomiarów

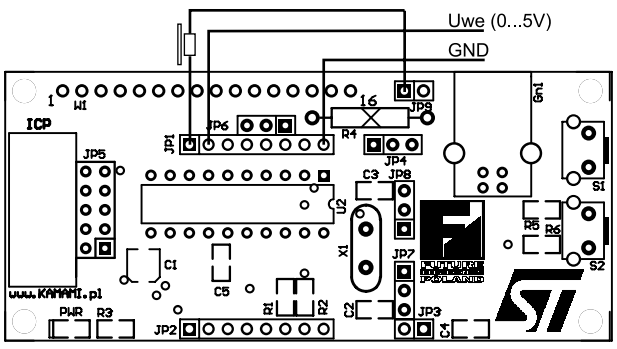
W kolejnej części cyklu aplikacji dla komputerka LITEcomp przedstawiamy obsługę przetwornika A/C wbudowanego w mikrokontroler ST7FLITE19 oraz – niejako przy okazji – wewnętrznej pamięci EEPROM. Te peryferia zostaną wykorzystane do zbudowania prostego jednokanałowego woltomierza z nieulotną pamięcią wyników pomiarów.

st7.ep.com.pl

Zakup taniego multimetru wyświetlającego aktualną wartość mierzonego napięcia lub prądu nie stanowi większego problemu. Znalezienie w równie przystępnej cenie multimetru umożliwiającego zapamiętanie wyników pomiarów jest już znacznie trudniejsze. Oczywiście woltomierz, którego oprogramowaniem zajmiemy się w tym artykule nie może zostać porównany nawet do najtańszego miernika uniwersalnego, gdyż umożliwia jedynie pomiar napięcia stałego, na dodatek w dość mocno ograniczonym zakresie. Należy go raczej potraktować jako przykład zastosowania przetwornika A/C oraz pamięci EEPROM, niż jako pełnowartościowy przyrząd pomiarowy.

Woltomierz w najprostszej wersji nie wymaga podłączenia żadnych elementów zewnętrznych. Poza buzzerem z wbudowanym generatorem, który w tym przykładzie sygnalizuje zapis wyniku pomiaru do pamięci. Jest on elementem opcjonalnym i nie jest wymagany do poprawnej pracy modułu, jednakże warto go podłączyć do modułu LITEcomp. Buzzer będzie przydatny w przypadku dodania funkcji komparatora, dzięki czemu osiągnięcie bądź przekroczenie nastawionych wartości napięcia będzie sygnalizowane dźwiękiem. Buzzer należy podłączyć pomiędzy wyprowadzenie PB0 a plus zasilania (**rys. 1**). Ustawienie stanu niskiego na wyjściu PB0 spowoduje wygenerowanie sygnału dźwiękowego.

Mierzone napięcie (z zakresu 0...5 V) jest podawane na wyprowadzenie PB1. Rozdzielczość odczytu wynosi 0,01 V. Zakres pomiarowy woltomierza można poszerzyć do 0...50 V przez dodanie zewnętrznego dzielnika 1:10. Rozdzielczość odczytu w tym przypadku zostanie ograniczona do 0,1 V. Program został napisany w języku C, w środowisku Raisonance RIDE. Szczegółowy opis instalacji środowiska RIDE oraz



Rys. 1.

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).

tworzenia projektu przedstawiono w oddzielnym artykule (opublikowanym w EP7/2007, dostępnym także na płycie CD–EP8/2007B).

Procedury obsługi wyświetlacza LCD

Wynik pomiaru wyświetlany jest na alfanumerycznym wyświetlaczu LCD 2x16 znaków. Ponieważ we wcześniej prezentowanych projektach wyświetlacz LCD był obsługiwany za pomocą programów napisanych w asemblerze, konieczne jest przygotowanie procedur jego obsługi w języku C.

Ogólna zasada sterowania wyświetlaczem jest oczywiście niezależna od języka, w którym jest pisany program, dlatego też przedstawione procedury zostaną omówione skrótowo Poniżej przedstawione są wszystkie niezbędne procedury obsługi wyświetlacza LCD w trybie 4–bitowym bez odczytu flagi zajętości. Kod obsługi wyświetlacza został wydzielony do osobnego pliku, z którego została utworzona namiastka biblioteki funkcji niezbędnych do sterowania wyświetlaczem. Kod ten może być wykorzystywany w następnych projektach dla komputerka LITEcomp, jak również w innych aplikacjach mikrokontrolerów ST7LITE. Jednorazowy wysiłek utworzenia z kodu biblioteki funkcji pozwoli w przyszłości na uniknięcie powtarzania tej samej czynności dla kolejnych projektów i przyśpieszy znacznie proces tworzenia aplikacji. Z funkcjami obsługi wyświetlacza powiązane są dwa pliki: w pliku *LCD.h* znajdują się definicje instrukcji sterownika HD44780 oraz konfiguracja połączeń, natomiast w pliku *LCD.c* znajdują się funkcje obsługi wyświetlacza.

Funkcje obsługi wyświetlacza zostały przygotowane do wyświetlacza pracującego w trybie 4–bitowym, przy czym zarówno linie danych, jak i linie sterujące muszą należeć do tego samego portu. Dodatkowo magistrala danych musi zajmować starszą połówkę portu. Taka konfiguracja połączeń została zastosowana w module LITEcomp. Zastosowanie przedstawionych w artykule funkcji w innym układzie wymaga spełnienia powyższego warunku. Przykładową konfigurację przedstawiono poniżej:

```
#define LCD_PORT  PADR
#define LCD_EN    (1   <<   1)
#define LCD_RS    (1   <<   3)
```

List. 1.

```
//funkcja opóźniająca o 10 mikrosekund dla kwarcu 16 MHz
void wait_10us(void)
{
unsigned char i;
 nop_();
for(i = 0; i < 10; i++);
}

// funkcja opóźniająca o wielokrotność 10 mikrosekund
void LCD_Delay(unsigned char delay)
{
for( ;delay > 0; delay--)
 wait_10us();
}
```

List. 2.

```
void LCD_Write(unsigned char dataToOut)
{
LCD_PORT |= LCD_EN;
LCD_OutNibble(dataToOut >> 4); // zapis starszego półbajtu
LCD_PORT &= ~LCD_EN;

LCD_PORT |= LCD_EN;
LCD_OutNibble(dataToOut & 0x0F); // zapis młodszego półbajtu
LCD_PORT &= ~LCD_EN;
}
```

List. 3.

```
void LCD_WriteCommand(unsigned char commandToWrite)
{
LCD_PORT &= ~LCD_RS; // Niski stan na linii RS -> zapis instrukcji
LCD_Write(commandToWrite); // zapis instrukcji
LCD_Delay((commandToWrite>2)?5:165); // jeśli instrukcja CLEAR lub HOME to
czekaj
// ok 1,65ms, w przeciwnym razie czekaj ok. 50us.
}
```

List. 4.

```
void LCD_WriteData(unsigned char dataToWrite)
{
LCD_PORT |= LCD_RS; //Wysoki stan na linii RS -> zapis danej
LCD_Write(dataToWrite); // zais danej
LCD_Delay(5); // oczekiwanie na wykonanie instrukcji ok. 50us
}
```

List. 5.

```
void LCD_WriteText(char * text)
{
while(*text)
 LCD_WriteData(*text++);
}
```

Funkcje opóźniające

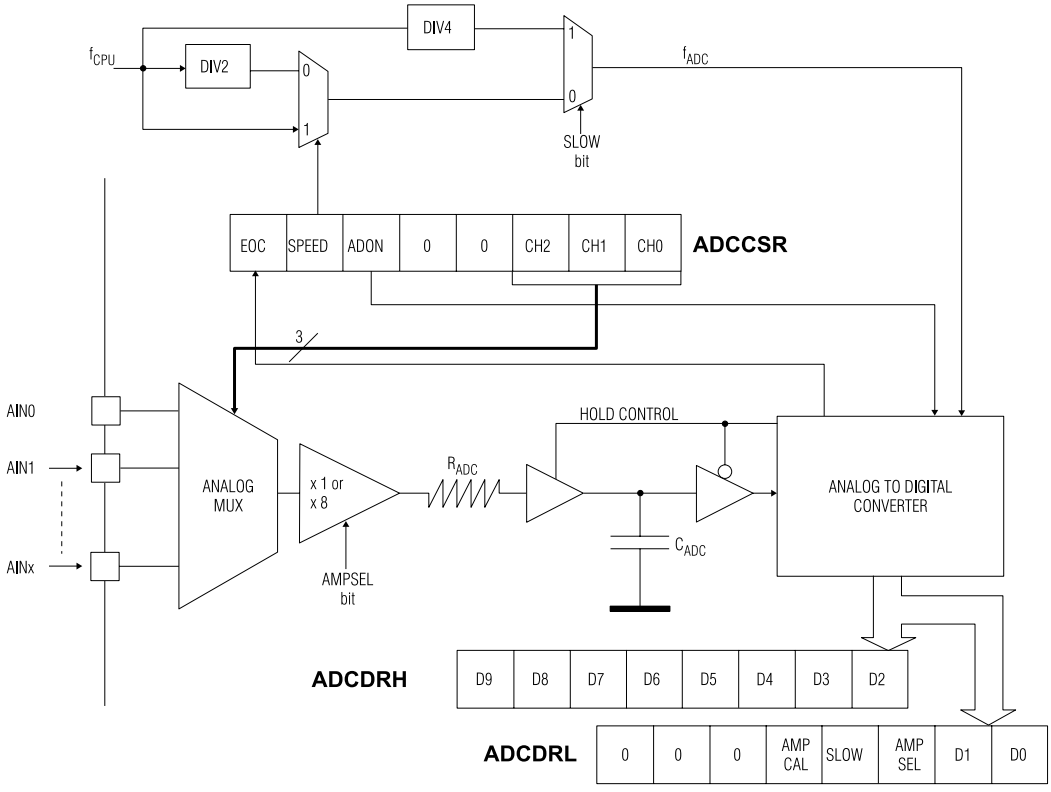
Wyświetlacz LCD podłączony do modułu LITEcomp ma wyprowadzenie R/!W na stałe podłączone do masy. Z tego powodu niemożliwy jest odczyt flagi zajętości wyświetlacza. W związku z tym po każdej operacji zapisu konieczne jest odczekanie czasu przewidzianego na wykonanie poszczególnych instrukcji. Przygotowałem dwie funkcje opóźniające. Pierwsza z nich odmierza czas ok. 10 µs, natomiast druga odmierza wielokrotność 10 µs dla częstotliwości taktowania mikrokontrolera równej 16 MHz. Kody procedur opóźniających przedstawiono na **list. 1**.

Zapis do wyświetlacza

Zapis instrukcji różni się od zapisu danej tylko stanem na linii RS, zasadnicza część procedur jest wspólna. Dlatego też wspólny fragment kodu realizujący zapis do wyświetlacza został umieszczony w oddzielnej funkcji. Wyświetlacz pracuje w trybie 4–bitowym, w związku z tym zapis odbywa się półbajtami. Jako pierwszy zapisywany jest starszy półbajt, a następnie młodszy półbajt. Kod funkcji przedstawiono na **list. 2**.

Zapis instrukcji

Sterownik HD44780 wykonuje większość instrukcji w czasie ok. 46 µs. Wyjątkiem są instrukcje CLEAR oraz HOME, które są wykonywane w czasie ok. 1,65 ms. Dlatego w zależności od kodu instrukcji



Rys. 2.

List. 6.
```
void LCD_GoTo(unsigned char x, unsigned char y)
{
LCD_WriteCommand(HD44780_DDRAM_SET + x + (64 * y));
}
```

List. 7.
```
void LCD_Init(void)
{
unsigned char i;
PADDR = 0b11111010; // inicjalizacja portów
PAOR = 0b11111010;
LCD_Delay(50); // oczekiwanie na ustabilizowanie napięcia zasilania
for(i = 0; i < 3; i++)
 {
 LCD_PORT |= LCD_EN;
 LCD_OutNibble(0x03);
 LCD_PORT &= ~LCD_EN;
 LCD_Delay(250);
 LCD_Delay(250);
 }
LCD_PORT |= LCD_EN;
LCD_OutNibble(0x02);
LCD_PORT &= ~LCD_EN;
LCD_Delay(100);

LCD_WriteCommand(HD44780_FUNCTION_SET | HD44780_FONT5x7 | HD44780_TWO_LINE |
HD44780_4_BIT);
LCD_WriteCommand(HD44780_DISPLAY_ONOFF | HD44780_DISPLAY_OFF);
LCD_WriteCommand(HD44780_CLEAR);
LCD_WriteCommand(HD44780_ENTRY_MODE | HD44780_EM_SHIFT_CURSOR | HD44780_EM_
INCREMENT);
LCD_WriteCommand(HD44780_DISPLAY_ONOFF | HD44780_DISPLAY_ON | HD44780_CUR-
SOR_OFF | HD44780_CURSOR_NOBLINK);
}
```

List. 8.
```
unsigned int GetADC(unsigned char channel)
{
unsigned int value;
ADCDRL |= (1 << SLOW);// fADC = 4MHz
ADCCSR = channel; // wybór kanału
ADCCSR |= (1 << ADON); // włączenie przetwornika
while((ADCCSR&(1 << EOC))==0);//oczekiwanie na zakończenie konwersji
value = (ADCDRL & 0x02);//
value |= (ADCDRH << 2); // odczytanie wyniku
ADCCSR &= ~(1 << ADON); // wyłączenie przetwornika
return value;
}
```

czas opóźnienia wynosi 50 µs, albo 1,65 ms. Kod funkcji zapisującej do wyświetlacza instrukcję przedstawiono na **list. 3**.

Zapis danej

Funkcja zapisuje daną do pamięci RAM sterownika HD44780. W zależności od ustawionego adresu może to być pamięć CGRAM lub DDRAM. Operacja zapisu danych do sterownika HD44780 według dokumentacji sterownika trwa maksymalnie 46 µs. W związku z tym generowane jest opóźnienie trwające 50 µs. Kod funkcji zapisującej daną do wyświetlacza przedstawiono na **list. 4**.

Wyświetlenie tekstu

Funkcja wyświetlająca tekst przyjmuje jako parametr wskaźnik do łańcucha znaków zakończonego zerem. Tekst jest wyświetlany pod aktualnymi współrzędnymi, dlatego przed wywołaniem funkcji wyświetlającej tekst należy ustawić odpowiednie współrzędne. Tekst może być przechowywany zarówno w pamięci RAM, jak i Flash. Kod procedury wyświetlającej tekst przedstawiono na **list. 5**.

Ustawienie współrzędnych

Funkcję ustawiającą adres pamięci RAM sterownika HD44780, pod który będą zapisywane dane przeznaczone do wyświetlenia przedstawiono na **list. 6**.

Inicjalizacja wyświetlacza

Proces inicjalizacji wyświetlacza HD44780 jest stosunkowo skomplikowany. Od jego przebiegu zależy poprawna praca wyświetlacza. Podczas inicjalizacji ustawiane są takie parametry pracy jak typ interfejsu, rozmiar znaków itp. Szczegółowy opis inicjalizacji wyświetlacza zawarty jest w dokumentacji sterownika HD44780 i nie będzie w tym miejscu powielany. Kod funkcji inicjującej sterownik przedstawiono na **list. 7**.

Przetwornik A/C

Mikrokontroler ST7FLITE19 posiada 10–bitowy przetwornik A/C o siedmiu wejściach. Schemat blo-

kowy przetwornika oraz powiązane z nim rejestry przedstawiono na **rys. 2**.

Przetwornik analogowo–cyfrowy może być taktowany sygnałem o częstotliwości F_{CPU}, $F_{CPU}/2$ lub $F_{CPU}/4$. Częstotliwość pracy przetwornika jest określana za pomocą bitów rejestru ADCCSR: SPEED i SLOW. Wzmacniacz wejściowy przetwornika umożliwia ośmiokrotne wzmocnienie sygnału wejściowego, przy czym maksymalną wartością napięcia wejściowego przy włączonym wzmocnieniu wynosi 430 mV. Wzmocnienie wzmacniacza wejściowego jest sterowane bitem AMPSEL. Przetwarzanie jest wyzwalane ustawieniem bitu ADON, natomiast jego zakończenie jest sygnalizowane poprzez ustawienie bitu EOC. Bity CH2...CH0 odpowiedzialne są za wybór kanału wejściowego. Wynik konwersji jest zwracany w rejestrach ADCDRH...ADCDRL, przy czym osiem starszych bitów wyniku znajduje się w rejestrze ADCDRH, natomiast dwa najmłodsze bity wyniku pomiaru znajdują się na bitach 1...0 rejestru ADCDRL.

Pomiar napięcia

Pomiar napięcia jest realizowany poprzez jednorazowe wyzwolenie przetwarzania napięcia podanego na wybrany kanał wejściowy przetwornika A/C i oczekiwaniu w pętli na ustawienie bitu EOC sygnalizującego koniec przetwarzania. Parametrem funkcji jest numer kanału wbudowanego multipleksera służącego do wyboru wejścia analogowego, z którego podawany jest mierzony sygnał. Kod funkcji przedstawiono na **list. 8**.

Wyświetlenie wartości napięcia

Konwersja i wyświetlenie wartości napięcia realizowane są przez funkcję *DisplayVolts*. Ponieważ zastosowanie w tak małym mikrokontrolerze jakim jest ST7FLITE19 bibliotecznych funkcji języka C do konwersji liczby zmiennoprzecinkowej zużyłoby całą dostępną pamięć, postanowiłem dokonać tego „na piechotę". W celu zminimalizowania operacji na zmiennych *float* wynik pomiaru jest przetwarzany na miliwolty, dzięki czemu większość operacji dzielenia dokonywana jest na liczbach całkowitych. Program zajmuje przez to mniej pamięci Flash i wykonuje się znacznie szybciej, niż w przypadku operacji na liczbach rzeczywistych, których obsługa w mikrokontrolerach 8–bitowych jest bardzo czasochłonna i pamięciożerna. Kod procedury przedstawiono na **list. 9**.

List. 9.
```
void DisplayVolts(unsigned int voltage)
{
float napiecie;
unsigned char jednostki,dziesietneCzesci,setneCzesci;
unsigned int reszta;

napiecie = voltage * 4.8828125; // konwersja na miliwolty

// rozbicie na poszczególne cyfry
jednostki = (unsigned int)(napiecie / 1000);
reszta = (unsigned int)napiecie % 1000;
dziesietneCzesci = (unsigned char)(reszta / 100);
setneCzesci = (unsigned char)((reszta % 100)/10);

// wyświetlenie wyniku
LCD_WriteData(jednostki + ,0');
LCD_WriteData(,.');
LCD_WriteData(dziesietneCzesci + ,0');
LCD_WriteData(setneCzesci + ,0');
LCD_WriteData(,V');
}
```

List. 10.
```
void CheckKeys(void)
{
if((PADR & SW1)==0) // jeśli wciśnięto przycisk S1
 {
 WriteVoltsToEEPROM(); // zapis napięcia do pamieci EEPROM
 while((PADR & SW1)==0); // oczekiwanie na zwolnienie przycisku
 }
if((PADR & SW2)==0) // jeśli wciśnięto przycisk S2
 {
 ReadVoltsFromEEPROM(); // odczyt napięcia z pamięci EEPROM
 while((PADR & SW2)==0); // oczekiwanie na zwolnienie przycisku
 }
}
```

List. 11.
```
void WriteVoltsToEEPROM(void)
{
if(eeIndex < EE_SIZE) //
 {
 EECSR = 2; // ustawiamy bit E2LAT -> zapis
 *((unsigned int *)(EE_BASE + (eeIndex*2))) = volts; // zapis do buforów
 EECSR |= 1; // ustawiamy bit E2PGM -> wyzwolenie operacji zapisu
 while((EECSR & 1)!=0); // oczekiwanie za zakończenie procesu zapisu
 Beep(1); // sygnalizacja zapisu
 eeIndex++; // następna pozycja
 readIndex = eeIndex; // przygotowanie do odczytu
 }
else
 {
 eeIndex = 0;// ustawienie pzycji na dolną granicę obszaru pamięci EEPROM
 Beep(2); // sygnalizacja osiągnięcia granicy obszaru
 }
}
```

Sprawdzenie stanu przycisków

Zapis wyniku pomiaru napięcia następuje po naciśnięciu przycisku S1 modułu LITEcomp, natomiast odczyt uprzednio zapamiętanego wyniku następuje po naciśnięciu przycisku S2. Stany przycisków są sprawdzane w nieskończonej pętli głównej. Wydzielono oddzielną funkcję sprawdzającą stan przycisków i wywołującą po wykryciu wciśnięcia przycisku odpowiednią funkcję. Po wykonaniu takiej funkcji następuje oczekiwanie na zwolnienie przycisku. Kod funkcji sprawdzającej stan przycisków przedstawiono na **list. 10**.

Pamięć EEPROM

Mikrokontroler ST7FLITE19 jest wyposażony w 128 bajtów nieulotnej pamięci EEPROM. Pamięć EEPROM jest zmapowana w przestrzeni adresowej mikrokontrolera. Dostęp do niej odbywa się poprzez typowe operacje na pamięci, jednakże kontrolowany jest poprzez bity E2LAT i E2PGM znajdujące się w rejestrze EECSR. Bit E2LAT określa kierunek operacji wykonywanej na pamięci (0–odczyt, 1–zapis), natomiast bit E2PGM służy do zainicjowania programowania pamięci. W jednym cyklu możliwe jest zaprogramowanie maksymal-

List. 12.
```
void ReadVoltsFromEEPROM(void)
{
unsigned char i;

readIndex--; // poprzednio zapamiętany pomiar

if(readIndex < 0) // jeśli przekroczono dolną granicę obszaru pamięci EEPROM
  {
  readIndex = EE_SIZE; // ustawienie pozycji na górną granicę obszaru
  Beep(2); // sygnalizacja osiągnięcia granicy obszaru
  }
else // w przeciwnym razie
 {
  EECSR = 0; // zerowanie bitu E2LAT -> odczyt
  volts = *((unsigned int *)(EE_BASE + ((readIndex)*2))); // odczyt z pamięci
  LCD_GoTo(0,0); // ustawienie współrzędnych
  DisplayVolts(volts); // wyświetlenie odczytnej wartości napięcia
  LCD_GoTo(12,0); // ustawienie współrzędnych
  LCD_WriteText(„M>”);
  DisplayPosition(readIndex); // wyświetlenie numeru pozycji w pamięci
  Beep(1); // sygnalizacja odczytu
 }
}
```

nie 32 bajtów danych, przy czym wszystkie dane muszą znajdować się w obrębie tego samego 32–bajtowego obszaru pamięci. Oznacza to, że 11 najstarszych bitów adresu musi mieć tę samą wartość. W przypadku niespełnienia tego warunku dane nie zostaną poprawnie zapisane do pamięci EEPROM. Przerwanie procesu programowana, czy to wskutek zaniku napięcia, czy też zresetowania mikrokontrolera, może zafałszować zawartość programowanej w tym momencie pamięci EEPROM.

Z procedurami obsługi pamięci EEPROM powiązane są następujące stałe:

```
#define  EE_BASE  0x1000
#define  EE_SIZE  16
```

Stała EE_BASE określa adres początku obszaru pamięci EEPROM. W przypadku mikrokontrolera ST7FLITE19 jest to liczba 0x1000. Stała EE_SIZE określa liczbę pomiarów przechowywanych w pamięci EEPROM i nie jest równa maksymalnemu rozmiarowi pamięci, chociaż użytkownik może zmienić ją na maksymalną wartość. Ponieważ jeden pomiar zajmuje dwa bajty pamięci, to maksymalna liczba pomiarów możliwa do zapamiętania w pamięci EEPROM jest równa 64. Warto jednak zostawić nieco wolnego miejsca w pamięci EEPROM dla dodatkowych funkcji, np. przechowywanie nastaw programowego komparatora napięcia.

Zapis wartości napięcia do pamięci EEPROM

Zapisanie aktualnej wartości napięcia do pamięci EEPROM następuje po naciśnięciu przycisku S1. Po zapełnieniu całego obszaru przewidzianego do przechowywania wyników pomiarów następuje powrót na początek obszaru pamięci i poprzednio zapisane wyniki są nadpisywane. Każdy rekord zajmuje dwa bajty pamięci EEPROM. Kod funkcji przedstawiono na **list. 11**.

Odczyt wartości napięcia z pamięci EEPROM

Naciśnięcie przycisku S2 wywołuje funkcję odczytu i wyświetlenia wartości napięcia z ostatnio zapisanej pozycji. Po odczytaniu zapamiętanej wartości następuje dekrementacja adresu. Kolejne naciśnięcia przycisku S2 będą skutkowały odczytem wcześniejszych wyników. Zapamiętanie nowego wyniku spowoduje przypisanie nowej wartości do wskaźnika odczytu. Osiągnięcie wartości minimalnej spowoduje ustawienie wskaźnika na wartość maksymalną. Kod funkcji przedstawiono na **list. 12**.

Podsumowanie

Zaprezentowany w artykule program woltomierza jednokanałowego może stanowić bazę do dalszej rozbudowy. W prosty sposób można przekształcić go do pełnienia funkcji np. woltomierza wielokanałowego. Funkcja odczytu napięcia przystosowana jest do odczytu napięcia ze wszystkich dostępnych w mikrokontrolerze wejść analogowych. Program zajmuje nieco ponad połowę pamięci Flash mikrokontrolera, tak więc ambitni Czytelnicy mogą do niego dodać dodatkowe funkcje. Jeżeli takie projekty powstaną – autorów zapraszamy do publikacji w EP.

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