Poznajemy mikrokontrolery ST7Lite, część 7

Pomiar napięcia, część 2

Jak wspomniano we wcześniejszych artykułach, wiele z mikrokontrolerów ST7 posiada wbudowany w strukturę przetwornik analogowo – cyfrowy. W tej części kursu pokazujemy w jaki sposób wykorzystać możliwości przetwornika we wołanej aplikacji i zmierzyć napięcie doprowadzone do jednego z wejść analogowych. W przykładzie posłużono się mikrokontrolerem ST7FLITE19.

Opis programu

Na początku pracy program inicjuje wskaźnik stosu, a do wyświetlacza LCD przesyłany jest ciąg instrukcji ustawiających go w trybie pracy z interfejsem 4–bitowym. Następnie ekran jest czyszczony a kursor ustawiany jest w pozycji HOME (adres znaku w DDRAM=0). Teraz z użyciem trybu adresowania indeksowego z przesunięciem 16–bitowym z pamięci programu pobierany jest tekt, który wyświetlany jest od tej pozycji kursora, aż do napotkania bajtu 0 na końcu definicji.

Wynik pomiaru napięcia jest wyświetlany w drugiej linii wyświetlacza. Kursor przemieszcza się do tej linii po wywołaniu procedury *gotoaxy*, oczywiście jeśli wcześniej wstawi się właściwe parametry do rejestru A przekazującego argumenty wywołania (4 starsze bity, to numer kolumny a 4 młodsze, to numer wiersza). Teraz kolejno wywoływane są procedury:

- pomiaru napięcia *volts*,
- konwersji wyniku pomiaru na postać liczby dziesiętnej *bin_2_dec*,
- konwersji liczby dziesiętnej na postać kodów ASCII *dec_2_ascii*,
- formatowania wyniku pomiaru *add_dp*.

Rolę bufora wyniku pełni zmienna w pamięci RAM o nazwie decims. Z dedykowanych jej komórek pamięci w trybie adresowania z przesunięciem 8–bitowym jest pobierany wynik pomiaru w postaci sformatowanego tekstu i przesyłany na wyświetlacz LCD. Na końcu tekstu jest dodawana litera „V“, następnie program odczekuje około 0,2 sekundy i cykl powtarza się tworząc w ten sposób nieskończoną pętlę realizowaną przez CPU mikrokontrolera.

Program główny jest bardzo prosty – procedury wywoływane są jedna po drugiej przekazując sobie nawzajem wynik pomiaru. Omówmy teraz kolejno ich działanie, zajmując się szczegółami ich implementacji. Niektóre z omawianych funkcji przydadzą się nam jeszcze w innych zastosowaniach.

Pomiar napięcia (*volts*)

Na **rys. 11** umieszczono algorytm działania funkcji pomiaru napięcia. Ma ona do spełnienia dwie ważne role: pierwszą z nich jest przeprowadzenie pomiaru napięcia przyłożonego do wejścia AIN4, drugą zamiana wyniku na jego reprezentację w miliwoltach.

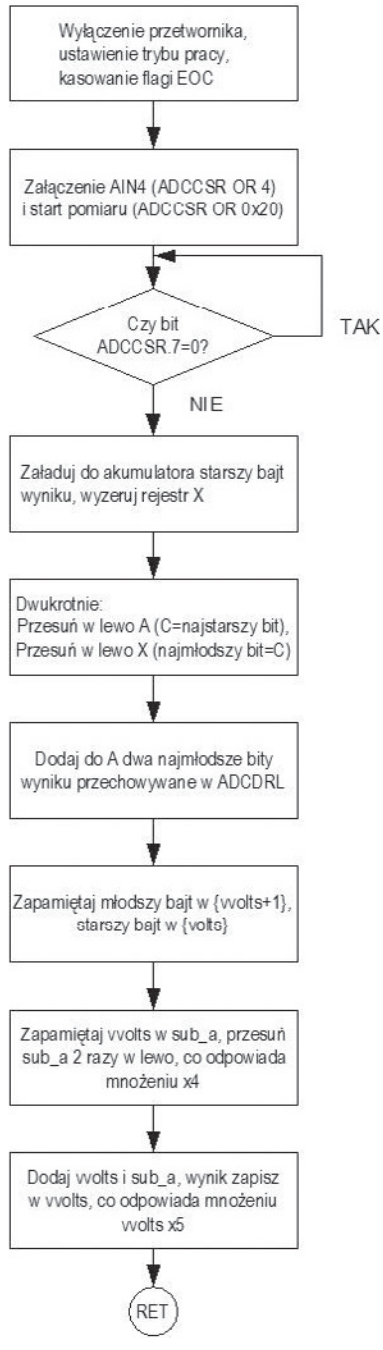
Jak pamiętamy napięcie referencyjne jest równe napięciu zasilania. W związku z tym jeśli napięcie zasilające wynosi 5 V a liczba bitów przetwornika jest równa 10, to napięcie może być mierzone z dokładnością do około 5 mV (a dokładnie 5 V/1024 4,888 mV). Ułóżmy zatem proporcję

$$\frac{1024}{\text{wynik pomiaru}} = \frac{5000\ \text{mV}}{X} \quad \text{i stąd}$$

$$X = \frac{5 \text{ x wynik pomiaru}}{1{,}024}$$

t.j. $X \approx 5$ x wynik pomiaru

Niestety, procedura mierząca będzie popełniać błąd: około 2,3%. W tym momencie, dla skrócenia programu i uproszczenia obliczeń godzimy się na niego. Jak łatwo wywnioskować z powyższych rozważań, wynik pomiaru musi być przemnożony przez 5. Można to zrobić na co najmniej trzy sposoby:



Rys. 11. Algorytm działania procedury pomiaru napięcia *volts*

List. 12. Fragmenty programu realizujące pomiar napięcia

```
st7/
;Program demonstrujący użycie przetwornika A/C
; i wyświetlacza LCD: 2 linie x 20 znaków

  #include "st7flite29.inc"

;segmenty pamięci
  BYTES
  segment byte at 80-FF 'ram0'
  segment byte at 100-1FF 'stack'
  segment byte at 200-27F 'ram1'
  segment byte at 1000-10FF 'eeprom'
  segment byte at E000-FFDF 'program'
  segment byte at FFE0-FFFF 'intvect'

  BYTES
;deklaracje zmiennych
  segment 'ram0'
del     DS.B
copya     DS.B
vvolts  DS.W
sub_a     DS.W
sub_b     DS.W
roznicaw  DS.W
decims DS.B 6

;deklaracje stałych
  WORDS
  segment 'program'
txt1 DC.B  "NAPIECIE(AIN4):",0

;-------------------------------------------------------
; początek programu głównego, inicjalizacja
; stosu, portów I/O oraz zmiennych
;-------------------------------------------------------
.init
  ld A,#$FF     ;port A jako wyjściowy
  ld PADDR,A    ;i stan niski na wszystkich wyjściach
  ld PAOR,A
  clr   A
  ld PADR,A

  ld PBDDR,A    ;port B jako wejściowy
  ld PBOR,A

  ld A,#$01     ;f_CPU = f_OSC/32 @ 1MHz
  ld MCCSR,A

  ld A,#$12     ;f_TIMER = f_CPU
  ld ATCSR,A

  ld A,#$DF     ;konfiguracja 12-bitowego timera
  ld ATRL,A     ;autoreload, przerwanie co ok.1 ms
  ld A,#$0F
  ld ATRH,A
  ret


;.....................................
;-------------------------------------------------------
; inicjalizacja wyświetlacza LCD w trybie interfejsu
; o długości słowa 4 bity
;-------------------------------------------------------
.lcd_init
  ld A,#100     ;pauza około 100 ms
  ld del,A
  call  delay
  ld A,PADR     ;RS(D4)=0,ENABLE(D7)=0,dane=0
  and   A,#$60
  ld PADR,A
  ld Y,#3       ;3-krotne przesłanie 0x03 do lcd
  or A,#3
  ld PADR,A
lcd_init_loop
  ld A,PADR     ;odczyt stanu rejestru danych portu A
  or A,#$80     ;ENABLE=1
  ld PADR,A
  nop
  nop
  and   A,#$7F     ;ENABLE=0
  ld PADR,A
  ld A,#5       ;opóźnienie 5 milisekund
  ld del,A
  call  delay
  dec   Y       ;czy juz powtórzono 3x?
  jrne  lcd_init_loop
  ld A,#2       ;zapisanie "2" do rejestru kontrolnego
  call  lcd_reg_write
  ld A,#$28     ;kolejne wartości inicjujące
($28,8,1,6,$C)
  call  lcd_reg_write
  ld A,#8
  call  lcd_reg_write
  ld A,#1
  call  lcd_reg_write
  ld A,#6
  call  lcd_reg_write
  ld A,#$C
  call  lcd_reg_write
  ret

;.....................................
;-------------------------------------------------------
; program główny, wyświetlenie tekstu zawartego w ROM
; i wyniku pomiaru napięcia na wejściu AIN4
;-------------------------------------------------------
.main
  rsp
  call  init
  call  lcd_init
  call  lcd_cls    ;czyszczenie LCD, ustawienie kursora
                   ;w pozycji HOME
                   ;wyświetlenie napisu
  clr   X       ;zerowanie rejestru indeksowego
main0
  ld A,(txt1,X)    ;pobranie kodu znaku do A
  and   A,#$FF     ;czy to koniec napisu (znak=0)?
  jreq  txt1end    ;tak,następny napis
  call  lcd_data_write  ;nie,zapis kodu znaku do LCD
  inc   X       ;następny znak z napisu
  jp main0
txt1end
  ld A,#$10     ;współrzędne dla 2-giej linii
  call  gotoaxy
main1
  call  volts      ;pomiar napięcia
  call  bin_2_dec     ;konwersja na liczbę dziesiętna
  call  dec_2_ascii     ;konwersja na ASCII
  call  add_dp     ;formatowanie wyniku
  ld Y,#5
  clr   X
main2
  ld A,(decims,X)  ;wyświetlenie wyniku pomiaru
  call  lcd_data_write
  inc   X
  dec   Y       ;czy wszystkie 6 znaków
  jrne  main2
  ld A,#'V'     ;wyświetlenie jednostki
  call  lcd_data_write
  ld A,#200     ;opóźnienie 0,2 sekundy
  ld del,A
  call  delay
  jra   txt1end    ;powtórz (pętla nieskończona)

;.....................................
```

- dodając do siebie 5–krotnie liczbę,
- wykorzystując funkcję mnożenia,
- składając wynik z operacji cząstkowych takich, jak dodawanie, przesunięcia itp.

Dla potrzeb prezentowanej aplikacji wybrano tę trzecią metodę.

Jak łatwo zauważyć mnożenie razy 5 można rozłożyć na trzy operacje: dwa mnożenia przez 2 i jedną sumę. Mnożenie przez 2 to nic innego, jak przesunięcie liczby w lewo o 1 bit. Aby pomnożyć liczbę razy 4 wystarczy przesunąć ją dwukrotnie w lewo. Podobnie jest z dzieleniem przez 2: zmienia się kierunek przesunięcia na prawy, ale zasada pozostaje bez zmian.

Z opisanego wyżej algorytmu korzysta procedura pomiarowa. Zapamiętuje ona wynik pomiaru w komórkach *sub_a* (używanych przez procedurę odejmowania) a następnie przesuwa go dwukrotnie w lewo. Po przesunięciu, do wyniku pomiaru nadal pamiętanego w komórkach *volts* jest dodawana wartość uzyskana w ten sposób. W ten prosty sposób wynik pomiaru mnożony jest przez 5 umożliwiając wyświetlenie zawartości rejestru przetwornika w miliwoltach.

Jacek Bogusz, EP
jacek.bogusz@ep.com.pl