Poznajemy mikrokontrolery ST7Lite, część 5

W tej części kursu pokażemy jak łatwo dołączyć do mikrokontrolera ST7Lite wyświetlacz LCD i wyświetlić na nim dowolny tekst.

Obsługa modułu wyświetlacza LCD 2 linie x 20 znaków w trybie z interfejsem 4–bitowym

Moduł alfanumerycznego wyświetlacza LCD wielokrotnie już był opisywany przez różnych autorów na łamach różnych czasopism i na stronach WWW, przy okazji różnych aplikacji. Nie będę więc powtarzał dobrze znanych informacji.

Do sterowania modułem LCD w trybie 4–bitowym są używane jego doprowadzenia RS i ENABLE (**rys. 9**). W prezentowanej aplikacji doprowadzenie READ/WRITE modułu podłączone jest na stałe do masy – można do niego tylko przesyłać dane, nie można ich odczytywać. Doprowadzenie kontrastu również dołączono na stałe do masy rezygnując z możliwości jego regulacji. Opisowo, schemat połączeń wygląda następująco:

- doprowadzenie ENABLE dołączone jest do PA7
- doprowadzenie RS dołączone jest do PA4
- doprowadzenia interfejsu danych DB4...DB7 dołączone są odpowiednio do PA0...PA3
- doprowadzenia 1, 3 i 5 modułu LCD dołączone są do masy
- doprowadzenie 2 modułu LCD dołączone jest do +5 V.

Program napisano dla mikrokontrolera ST7FLITE29, jednak poprzez zmianę adresów segmentów bardzo łatwo może być przeniesiony na mikrokontroler ST7FLITE19 lub inny z rodziny ST7. Zadaniem programu jest wyświetlenie napisu umieszczonego w pamięci Flash mikrokontrolera, zadeklarowanego jako stała. Podany przykład pozwoli jednocześnie zapoznać się ze strukturą programu napisanego w asemblerze, ze sposobami deklaracji stałych i zmiennych oraz przeprowadzania nastaw portów i układów peryferyjnych. Może posłużyć jako rodzaj szablonu przy tworzeniu własnych programów. Korzystając z możliwości wyświetlenia przez moduł LCD niemalże dowolnego ciągu znaków, można chociażby zmierzyć i wyświetlić napięcie doprowadzone do jednego z wejść analogowych mikrokontrolera, zbudować zegar czy interfejs użytkownika do dowolnej aplikacji sterującej. Program napisano z użyciem darmowego IDE (ST7 Visual Develop) udostępnianego przez firmę ST Microelectronics.

ST7FLITE29 pracuje bez oscylatora kwarcowego, wykorzystując wbudowany w strukturę generator zegarowy o częstotliwości 1 MHz. Aby go załączyć wystarczy ustawić odpowiednie bity opcji przy pomocy programatora (nastawa *RC Oscillator = ON*), a nóżki XTAL1 i XTAL2 dołączyć do masy. Jest to o tyle ważne, że do odmierzania czasów opóźnień używany jest timer *Lite* taktowany częstotliwością 32–krotnie mniejszą, niż częstotliwość generatora zegarowego. Jeśli używany będzie generator o innej częstotliwości, należy odpowiednio zmienić cza-

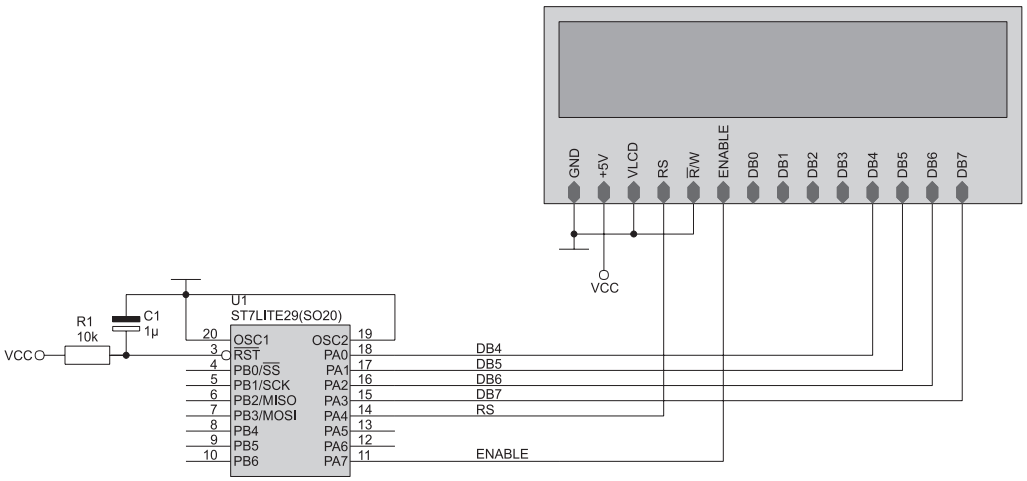
List. 8. Nastawy 12–bitowego timera Lite (fragment funkcji init)

```
  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  ;z funkcją autoreload, przerwanie co ok.1 ms
  ld A,#$0F
  ld ATRH,A
```

Prawie komplet

Na CD–EP10/2006B zamieściliśmy wszystkie dotychczas publikowane odcinki kursu ST7.



Rys. 9.

sy opóźnień. Nastawy, między innymi timera *Lite*, przeprowadzane są przez funkcję *init*, a fragment dotyczący nastaw timerów ukazany jest na listingu 2.

Po zerowaniu, CPU pobiera 2–bajtowy adres programu głównego umieszczony w tablicy wektorów przerwań pod adresem FFFF:FFFE a następnie rozpoczyna realizację wskazywanych w ten sposób instrukcji. W przytaczanym programie przykładowym (**list. 8**) są to instrukcje rozpoczynające się od etykiety *main*. Tablicę wektorów przerwań umieszczono w osobnym segmencie nazwanym *intvect*. Oczywiście nazwa jest dowolna, jej wybór nie jest podyktowany przez żadne kryteria. Można ten segment nazwać na przykład „przerwania". Nie ma to żadnego znaczenia.

Przy okazji: segmentacja pamięci to coś, co trzeba dobrze zrozumieć pisząc programy w asemblerze. Deklaracja segmentów to mapa pamięci zawarta w programie. Dzięki tej deklaracji oraz związanym z nią mnemonikom, programista może mieć wpływ na rozmieszczenie części programu użytkowego w pamięci mikrokontrolera. Ma to znaczenie nie tyle estetyczne, ile umożliwia odpowiednie umieszczenie odpowiednich fragmentów kodu lub zmiennych, w konkretnych rejonach pamięci. Na **list. 9** przedstawiono przykładową definicję segmentów pamięci oraz ich zastosowanie. Jeśli program pisany jest w języku C, to kompilator sam dba o odpowiednie rozmieszczenie zmiennych i stałych w pamięciach mikrokontrolera, natomiast gdy używany jest asembler, to o wszystko musi zadbać programista.

Program *main* wykonywany po sygnale *reset* rozpoczyna się od wywołania podprogramu *init* konfigurującego układy peryferyjne i porty mikrokontrolera. Cały port A ustawiany jest jako wyjściowy, port B pracuje w trybie domyślnym, to znaczy jako wejściowy. Następnie funkcja *main* wywołuje podprogram *lcd_init* i związany z nim podprogram czyszczenia ekranu *lcd_cls*. Dalej z pamięci Flash mikrokontrolera, do napotkania bajtu o wartości 0, pobierany jest do akumulatora ciąg znaków wskazywany w trybie adresowania bezpośredniego z rejestrem indeksowym X i przesyłany do wyświetlacza przez funkcję *lcd_data_write* (argument w akumulatorze). Rejestr X ustala offset dla pewnego stałego adresu w pamięci Flash mikrokontrolera, spod którego pobierany jest bajt zawierający kod znaku.

Funkcja przesyłająca dane (*lcd_write*) dzieli bajt z akumulatora na dwie połówki 4–bitowe i przesyła je jedną po drugiej, począwszy od starszej. Stan portu A modyfikowany jest przez funkcje logiczne AND i OR, aczkolwiek zaimplementowana lista rozkazów zmusza do wykonywania działań nie tyle na samym rejestrze danych portu, ile na jego kopii. Stosowanie funkcji logicznych ma tę zaletę, że jeśli nie używane w tym przykładzie programowania wyprowadzenia portu A zostaną użyte do sterowania dołączonym z zewnątrz układem peryferyjnym, to ich stan nie będzie zakłócony przez sygnały sterujące modułem wyświetlacza. Wartość zapisywana jest do pamięci modułu LCD przez opadające zbocze sygnału ENABLE (PA7). O tym, gdzie są przesyłane dane (czy do rejestru kontrolnego, czy do rejestru danych modułu LCD) decyduje stan sygnału RS (PA4).

Procedura *delay* korzysta z przerwań timera *Lite* zmniejszając stan zmiennej globalnej *del*. Przed wywołaniem procedury opóźnienia zmienna ta musi zawierać pożądaną jako czas opóźnienia wielokrotność 1 ms. Przerwanie timera generowane jest co około 1 ms, a każde wywołanie procedury obsługi powoduje zmniejszenie wartości zmiennej *del* o 1. Gdy zmienna osiągnie wartość 0, przerwanie jest blokowane i funkcja *delay* wykonuje instrukcję *ret* kończąc swoją pracę.

Starałem się aby program zawierał dużo czytelnych komentarzy tak, aby jego analiza nie była trudna nawet dla osób stawiających pierwsze kroki. Niestety wymagana jest chociażby podstawowa znajomość asemblera w czym mogą być pomocne artykuły zamieszczone w poprzednich wydaniach Elektroniki Praktycznej (komplet publikujemy na CD–EP10/2006B.

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

List. 9. Deklaracje segmentów pamięci dla ST7FLITE29 wraz z przykładami ich użycia

```
;deklaracje segmentów 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 ,pro-
gram'
  segment byte at FFE0-FFFF ,in-
tvect'

;deklaracje zmiennych
  segment ,ram0'
.del   DS.B 1
.copya DS.B 1

;deklaracje stałych i kod programu
  WORDS
  segment ,program'
........ tu umieścić kod programu
.........
  segment ,intvect'
;-----------------------------------
; wektory przerwań mikrokontrolera
ST7FLITE29
;-----------------------------------
  DC.W it_ret
  DC.W it_ret
  DC.W it_ret
.sci DC.W it_ret
.timb  DC.W timerb
.tima  DC.W it_ret
.spi DC.W it_ret
  DC.W it_ret
  DC.W it_ret
.ext3  DC.W it_ret
.ext2  DC.W it_ret
.ext1  DC.W it_ret
.ext0  DC.W it_ret
  DC.W it_ret
.soft  DC.W it_ret
.rst DC.W main

  END
```