Poznajemy mikrokontrolery ST7Lite, część 2

Często o zastosowaniu mikrokontrolera w aplikacji decydują funkcje, które mogą pełnione przez porty wejścia/wyjścia. Mikrokontrolery ST7 oferują szereg trybów pracy tzw. linii IO, które czynią je bardzo atrakcyjnymi w wielu zastosowaniach. Możliwość rezygnacji z tranzystorów, czy dodatkowych układów dopasowujących port mikrokontrolera do obciążenia (driverów), przy szczupłości miejsca na płytce może być bardzo cenna. W tym odcinku zostanie przedstawiony pokrótce opis trybów pracy portów oraz sposoby ich konfiguracji. Opis zilustruje program obsługi wyświetlacza LCD, który może być przydatny w wielu aplikacjach.

Porty I/O mikrokontrolera ST7FLITE19/29

Każdy z portów zawiera 8 linii danych, które umożliwiają dwukierunkowe przesyłanie bajtów danych – obojętnie czy są to słowa sterujące układami peryferyjnymi, czy też zawierające informacje na temat ich stanu. Każdy pojedynczy bit portu może pełnić niezależną od pozostałych bitów funkcję będąc wejściem lub wyjściem cyfrowym, to jest takim, którego stan logiczny jest umownym napięciem „0" lub „1". Dodatkowo, pewne bity portów (będę je zamiennie nazywał liniami portów), mogą pełnić funkcje specjalne, np.: wejścia analogowe, wejścia sygnalizacji zewnętrznego zdarzenia (przerwania), mogą również wyprowadzać sygnały wbudowanych w strukturę układów peryferyjnych jak np. generator PWM, interfejs UART czy ICC (**tab. 2**). Wymaga to od projektanta urządzenia znacznej uwagi, ponieważ łatwo jest przypisać wyprowadzeniom portu sprzeczne ze sobą funkcje. Będzie o tym mowa w dalszej części artykułu.

Z odpowiednimi bitami portów skojarzone są rejestry konfiguracyjne. ST7 zawiera dwa rejestry skojarzone z funkcjami portów. Są to rejestr kierunku DDR i rejestr opcji OR. Trzecim rejestrem jest rejestr danych DR, który może zawierać dane wejściowe lub wyjściowe, w zależności od nastaw dwóch poprzednich. Tryby pracy poszczególnych linii portów są konfigurowane przez odpowiednie ustawienie związanych z nimi bitów w owych rejestrach zgodnie z zasadą, że linii numer X odpowiada bit konfiguracyjny również o tym numerze. W niektórych przypadkach funkcja wyprowadzenia portu może być również ustalana przez załączenie skojarzonego z nim układu peryferyjnego (np. generatora PWM). O takich przypadkach będzie mowa w przykładach programów użytkowych. Typowo jednak stan niski bitu X w rejestrze DDR związanym z danym portem powoduje ustawienie trybu pracy linii portu o numerze X jako wejścia. Jeśli rejestr OR na tejże pozycji ma ustawioną wartość „1", to jest to wejście z dołączonym rezystorem podciągającym (*pull–up*), jeśli zaś zero, to jest to wejście bezpotencjałowe. Praca linii portu jako wejścia wymaga ustawienia „1" na związanej z nim pozycji w rejestrze DDR. I w tym przypadku rejestr opcji pełni zbliżoną funkcję: „0" powoduje odłączenie, a „1" dołączenie rezystora podciągającego wyjście. Na **list. 1** przedstawiono fragment programu napisanego w asemblerze mikrokontrolera ST7 zawierający sposób konfigurowania portów.

Każda z linii portów IO może pracować jako wejście przerwania. W ten sposób nawet „maleńki" ST7FLITE może przyjmować kilka-

Biblioteki dla Protela
Pod adresem *st7.ep.com.pl* są dostępne (przygotowane przez nas) biblioteki dla Protela 99SE oraz Altium Designera/DXP2004 zawierające symbole biblioteczne i schematowe mikrokontrolerów rodziny ST7Lite.

List. 1. Przykładowe konfiguracje portu A i portu B.

```
  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 A,#$F0   ;port B: bity 0..3 wejścia, 4..7 wyjścia
  ld PBDDR,A
  ld A,#$FF   ;załączenie rezystorów zasilających
  ld PBOR,A
  clr   A     ;stan niski na wyjściach
  ld PBDR,A
```

naście zgłoszeń żądania obsługi od urządzeń zewnętrznych. Czasami będzie wymagane dodatkowe testowanie źródła sygnału przerwania w celu dokładnego ustalenia skąd pochodzi sygnał (zależy to od konkretnej aplikacji mikrokontrolera i w większości przypadków nie jest konieczne), ponieważ jeden wektor przerwania zewnętrznego skojarzony jest z czterema bitami portów. Na przykład bity portu A o numerze od 0 do 3 generują wspólny wektor przerwania oznaczony jako EI0. Tryb pracy wybranej linii portu jako wejścia przerwania zewnętrznego wymaga, aby linia ta była wejściem (np. DDRA.0=0) i aby odpowiednia pozycja rejestru opcji zawierała „1" (np. ORA.0=1).

Do zmiany stanu bitów portu nie wolno używać rozkazów BSET (ustawianie bitu) i BRES (kasowanie bitu). Próba ich użycia spowoduje zgłoszenie przez kompilator komunikatu o błędzie. Dotyczy to szczególnie rejestru danych. Zapis do rejestru danych, gdy linia pracuje jako wejściowa, nie powoduje zmiany jej stanu. Jest to szczególnie wygodne w sytuacjach, gdy linie portu pełnią funkcje mieszane tzn. pracują w różnych trybach.

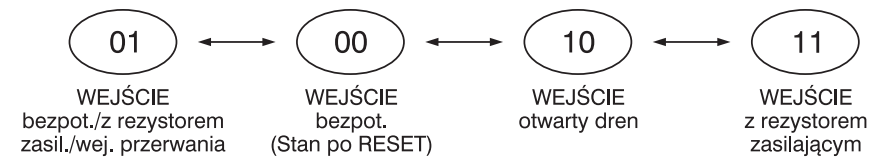
Na koniec tego krótkiego opisu funkcjonalnego jeszcze dwie uwagi. Jak łatwo zorientować się z tab. 1, część linii portów może służyć na przykład do zasilania diod LED, segmentu wyświetlacza LED lub bezpośrednio sterować bramką tyrystora. Nieprzypadkowo również linie te skojarzone są z wyjściami generatorów PWM. Producent podaje, że bez żadnych obaw mogą one zasilać układy zewnętrzne prądem ciągłym o wartości 20 mA, a wartość maksymalna jest określona jako aż 50 mA! Należy jednak zwracać szczególną uwagę na maksymalną, dopuszczalną dla danej obudowy, moc strat. Druga uwaga dotyczy sposobu zmiany funkcji linii portu z wejściowej na wyjściową. Na **rys. 3** przedstawiono graf przejść pomiędzy trybami zaczerpnięty z dokumentacji mikrokontrolera. Nie stosowanie się do niego może zaowocować zakłóceniami pracy mikrokontrolera przy zmianie trybu pracy portu, czy pojedynczego jego bitu. W dalszej konsekwencji, na skutek na przykład generowanych zgłoszeń obsługi przerwania zewnętrznego,

Tab. 2. Opis funkcji bitów portów IO mikrokontrolera ST7FLITE19/29

Port	Numer bitu	Funkcja
PA	0	Standardowe I/O o podwyższonej obciążalności prądowej Wejście CAPTURE timera LITE Wejście przerwania zewnętrznego EI0
	1	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wejście RELOAD timera LITE Wejście przerwania zewnętrznego EI0
	2	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wyjście generatora PWM0 Wejście przerwania zewnętrznego EI0
	3	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wyjście generatora PWM1 Wejście przerwania zewnętrznego EI0
	4	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wyjście generatora PWM2 Wejście przerwania zewnętrznego EI1
	5	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wyjście generatora PWM3 Doprowadzenie interfejsu ICC–ICCDATA Wejście przerwania zewnętrznego EI1
	6	Standardowe I/O Wyjście generatora zegarowego Wejście zegara interfejsu ICC–ICCCLOCK Wejście zewnętrznego sygnału BREAK Wejście przerwania zewnętrznego EI1
	7	Standardowe I/O o podwyższonej obciążalności prądowej[1] Wejście przerwania zewnętrznego EI1
PB	0	Standardowe I/O Wejście SS (Slave Select) interfejsu SPI Wejście analogowe 0 przetwornika A/C Wejście przerwania zewnętrznego EI3
	1	Standardowe I/O Doprowadzenie SCK (zegar) interfejsu SPI Wejście analogowe 1 przetwornika A/C Wejście przerwania zewnętrznego EI3
	2	Standardowe I/O Doprowadzenie sygnału MISO (Master Input/Slave Output) interfejsu SPI Wejście analogowe 2 przetwornika A/C Wejście przerwania zewnętrznego EI3
	3	Standardowe I/O Wejście MOSI (Master Output/Slave Input) interfejsu SPI Wejście analogowe 3 przetwornika A/C Wejście przerwania zewnętrznego EI2
	4	Standardowe I/O Wejście zegarowe zewnętrzne (taktowanie CPU) Wejście analogowe 4 przetwornika AD Wejście przerwania zewnętrznego EI2
	5	Standardowe I/O Wejście analogowe 5 przetwornika A/C Wejście przerwania zewnętrznego EI2
	6	Standardowe I/O Wejście analogowe 6 przetwornika A/C Wejście przerwania zewnętrznego EI2

powodować to może zupełnie nieprzewidywalne zachowanie układu, mimo jego poprawności.

Ze względu na wielofunkcyjność linii portu, czasami konieczne staje się odseparowanie mikrokontrolera od układów peryferyjnych. Dobrym przykładem jest programowanie mikrokontrolera w układzie, bez wylutowywania, czy wyciągania z podstawki. Dołączone z zewnątrz do linii interfejsu ICP układy peryferyjne mogą obciążać sygnały dostarczane przez programator lub



Rys. 3. Poprawne zmiany trybów pracy portów IO

List. 2. Konfiguracja 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
```

wręcz zwierać je do masy. Na **rys. 4** przedstawiono jedną z propozycji dołączenia programatora, gdy wejścia portu A są używane na przykład do pomiaru napięcia. Wówczas ich impedancja wejściowa jest bardzo duża, a spadek na dołączonym rezystorze maskującym nie wpływa w zasadniczy sposób na wynik pomiaru. Rezystory maskujące doskonale pełnią swą funkcję, co wypróbowałem w wielu aplikacjach.

Przykładowy program do obsługi modułu wyświetlacza LCD – 2 linie po 20 znaków, interfejs 4–bitowy.

Moduł alfanumerycznego wyświetlacza LCD wielokrotnie już był opisywany przez różnych autorów na łamach różnych czasopism i na stronach www, przy okazji różnych aplikacji. Do sterowania modułem LCD z interfejsem 4–bitowym używane są jego doprowadzenia RS i ENABLE. W prezentowanej aplikacji doprowadzenie READ/WRITE modułu podłączone jest na stałe do masy – można do niego tylko przesyłać dane, nie można ich odczytywać. Nóżkę ustawiania kontrastu również dołączono na stałe do masy rezygnując z możliwości regulacji. Opisowo, schemat połączeń wygląda następująco:

- doprowadzenie ENABLE jest dołączone do PA7
- doprowadzenie RS jest dołączone do PA4
- doprowadzenia interfejsu danych DB4...DB7 są dołączone odpowiednio do PA0...PA3
- doprowadzenia 1, 3 i 5 modułu LCD są dołączone do masy
- doprowadzenie 2 modułu LCD jest dołączone 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 pewna stała. Podany przykład pozwoli jednocześnie niejako zapoznać się ze strukturą programu napisanego w asemblerze, ze sposobami deklaracji stałych i zmiennych oraz konfigurowania 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.

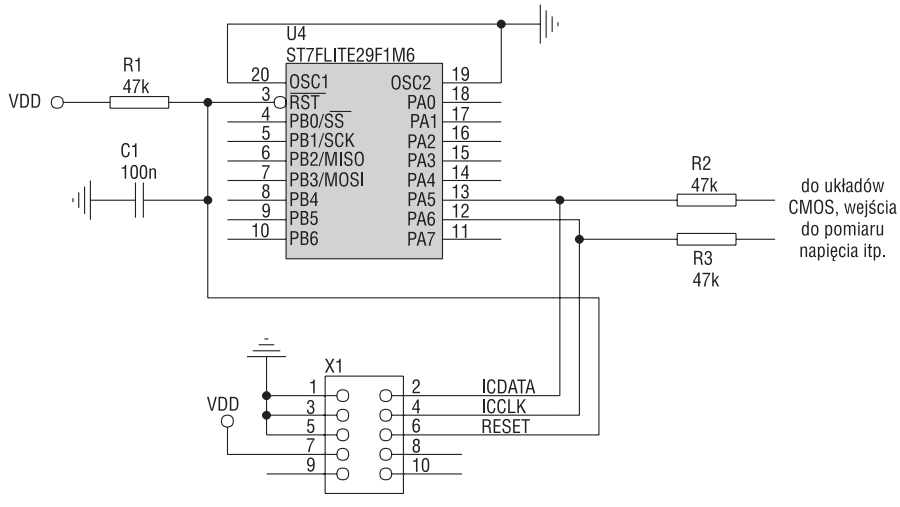
ST7FLITE29 pracuje bez oscylatora kwarcowego, wykorzystując wbudowany w strukturę generator zegarowy o częstotliwości 1 MHz. Aby go załączyć wystarczy ustawić odpowiednie bity opcji przy pomocy programatora (RC Oscillator= ON), a nóżki XTAL1 i XTAL2 dołączyć do masy. Jest to o tyle ważne, że do odmierzania czasów opóźnień używany jest timer Lite taktowany częstotliwością 32–krotnie mniejszą, niż częstotliwość generatora zegarowego. Jeśli będzie używany generator o innej częstotliwości, należy odpowiednio zmienić czasy opóźnień. Nastawy, między innymi timera Lite, przeprowadzane są przez funkcję *init*, a fragment dotyczący nastaw timerów przedstawiono na **list. 2**.

Po sygnale zerowania CPU mikrokontrolera 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 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.

Lista rozkazów ST7Lite

Wykaz poleceń asemblerowych wraz z opisem ich działania publikujemy na tekturce „CD" (tylko w EPo/oL).

Przy okazji: segmentacja pamięci to temat, który 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, ale umożliwia prawidłowe rozmieszczenie odpowiednich fragmentów kodu i zmiennych w konkretnych obszarach pamięci. Na **list. 3** przedstawiono



Rys. 4. Propozycja dołączenia programatora do aplikacji z mikrokontrolerem ST7FLITE19/29

przykładowe definicje segmentów pamięci oraz ich zastosowanie. Jeśli program jest pisany 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* przeprowadzającego konfigurację układów peryferyjnych i portów mikrokontrolera. Cały port A ustawiany jest jako wyjściowy, port B pozostaje w konfiguracji domyślnej, to znaczy pracuje jako port wejściowy. Następną wywoływany jest podprogram *lcd_init* i związany z nim podprogram czyszczenia ekranu *lcd_cls*. Dalej z pamięci Flash mikrokontrolera pobierane są cyklicznie do akumulatora znaki wskazywane przez rejestr indeksowy X (tryb adresowania bezpośredniego), a następnie przesyłane do wyświetlacza przez funkcję *lcd_data_write* (argument w akumulatorze). Operacja ta trwa aż do napotkania bajtu o wartości 0. 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 4–bitowe części i przesyła jedną po drugiej, począwszy od starszej. Stan portu A jest modyfikowany 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 jest zapisywana w pamięci modułu LCD na opadającym zboczu sygnału ENABLE (PA7). O tym, gdzie przesyłane są 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 tej procedury zmienna *del* musi zawierać pożądaną jako czas opóźnienia wielokrotność 1 ms. Przerwanie timera jest generowane 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 może być pomocna poprzednia część artykułu.

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

List. 3. 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 'program'
  segment byte at FFE0-FFFF 'intvect'

;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
```