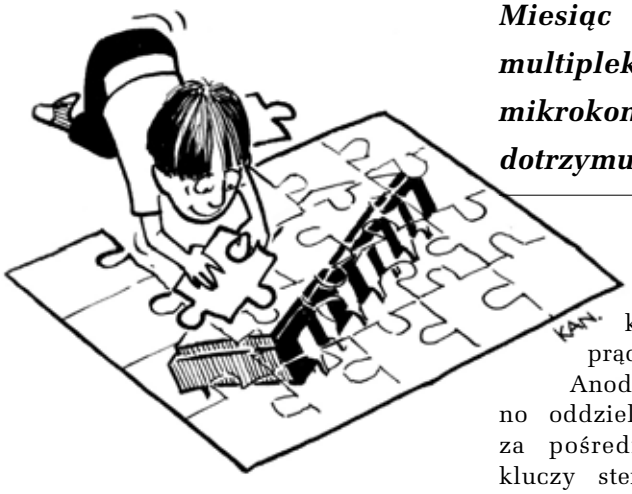
Podstawy projektowania systemów mikroprocesorowych, część 5



Miesiąc temu zapowiedzieliśmy pokazanie multipleksowego sposobu sterowania za pomocą mikrokontrolera '51 wyświetlaczy LED. Obietnicy dotrzymujemy - zapraszamy do lektury!

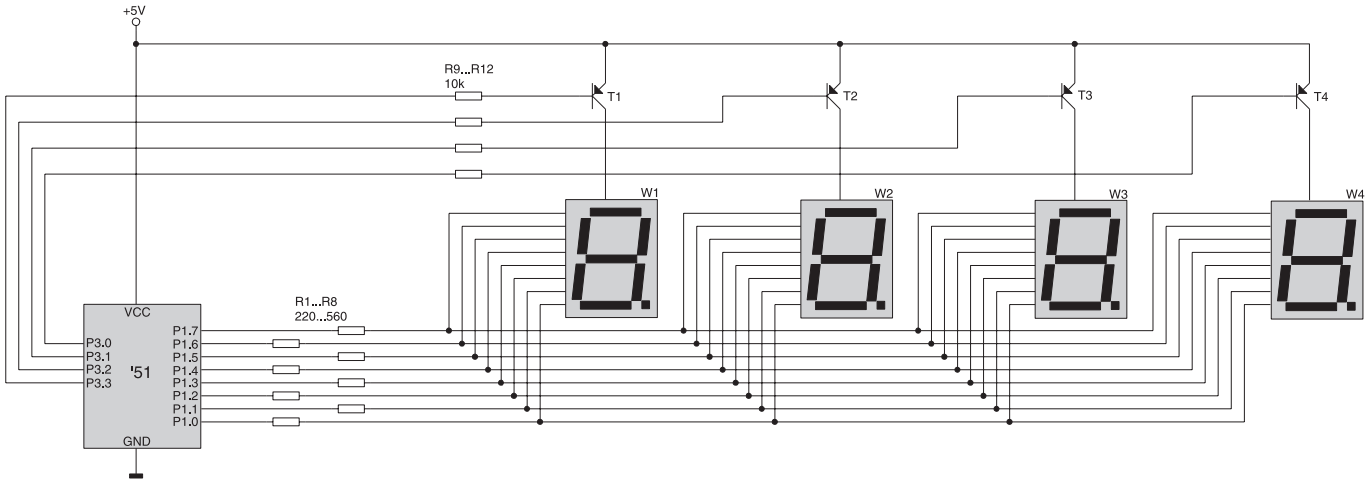
Schemat układu wykorzystującego tę zasadę sterowania wyświetlaczami siedmiosegmentowymi pokazano na **rys. 19**. W układzie tym wykorzystano wyświetlacze ze wspólną anodą, których odpowiadające sobie katody połączono wspólnie z odpowiednim wyprowadzeniem mikrokontrolera. Rezystory ograniczające prąd mają mniejsze wartości niż w układach z wyświetlaniem statycznym w związku z koniecznością zapewnienia odpowiednio dużego prądu średniego (tutaj $^1/_4$ natężenia prądu świecącego wyświetlacza) gwarantującego pożądaną jasność świecenia - im więcej grup diod byłoby sterowanych, tym rezystory te powinny mieć mniejszą wartość, nie należy jednak przekraczać dopuszczalnego prądu linii portu.

Anody wyświetlaczy dołączono oddzielnie do plusa zasilania za pośrednictwem tranzystorów-kluczy sterowanych przez mikrokontroler. Układ ten działa według następującego algorytmu (dla jednego wyświetlacza):

- ustawienie stanów linii portu P1 odpowiadających informacji wyświetlanej przez wyświetlacz W1 (kod odpowiadający kształtowi wyświetlanego znaku),
- włączenie wyświetlacza 1 (P3.3=0),
- odczekanie czasu, w trakcie którego wyświetlacz świeci,
- wyłączenie wyświetlacza 1 (P3.3=1).

Powyższe kroki należy powtórzyć jeszcze trzykrotnie (dla wyświetlaczy W2, W3 i W4), a następnie rozpocząć cykl od początku (od wyświetlacza W1). Czas trwania pojedynczego cyklu powinien być tak dobrany, aby częstotliwość zapalania pojedynczego wyświetlacza nie była mniejsza niż 30...40 Hz (aby przekroczyć możliwości rejestrowania zmian przez ludzkie oko), a najlepiej oscylowała w okolicach 100 Hz, co pozwoli na wyświetlanie pozbawione efektu migotania znanego z ekranów monitorów i telewizorów. Dalsze zwiększanie częstotliwości odświeżania nie poprawia już jakości wyświetlanej informacji, powoduje za to wzrost ilości zakłóceń radioelektrycznych generowanych przez wyświetlacz oraz niepotrzebnie marnuje moc obliczeniową mikrokontrolera. Warto zauważyć, że w takim układzie można łatwo sterować jasnością świecenia poszczególnych wyświetlaczy - wystarczy dobrać odpowiednio czasy trwania świecenia danego wyświetlacza - im dłuższy ten czas, tym jaśniejsze świecenie.

Czytelnicy zaznajomieni z urządzeniami wykonywanymi np. w technice TTL zauważą, że brak pomiędzy wyprowadzeniami mikrokontrolera a wyświetlaczem ukła-



Rys. 19. Schemat 4-cyfrowego wyświetlacza LED sterowanego multipleksowo wprost z programowym generowaniem kształtu wyświetlanego znaku

List. 1. Przykładowy program sterujący działaniem 4-cyfrowego wyświetlacza LED

```
      ;komórki pamięci od adresu 030h do 033h zawierają kombinację stanów
      ;segmentów wyświetlacza: 0 - oznacza segment zapalony



   MOV P1,030H       ;przesłanie danej do portu P1
   CLR P3.3          ;włączenie pierwszego wyświetlacza
   LCALL CZEKAJ      ;wywołanie procedury opóźniającej
   SETB P3.3         ;wyłączenie pierwszego wyświetlacza

   MOV P1,031H       ;przesłanie danej do portu P1
   CLR P3.2          ;włączenie drugiego wyświetlacza
   LCALL CZEKAJ      ;wywołanie procedury opóźniającej
   SETB P3.2         ;wyłączenie drugiego wyświetlacza

   MOV P1,032H       ;przesłanie danej do portu P1
   CLR P3.1          ;włączenie trzeciego wyświetlacza
   LCALL CZEKAJ      ;wywołanie procedury opóźniającej
   SETB P3.1         ;wyłączenie trzeciego wyświetlacza

   MOV P1,033H       ;przesłanie danej do portu P1
   CLR P3.0          ;włączenie czwartego wyświetlacza
   LCALL CZEKAJ      ;wywołanie procedury opóźniającej
   SETB P3.0         ;wyłączenie czwartego wyświetlacza

      ;koniec podprogramu wyświetlającego
```

List. 2. Procedura realizująca opóźnienie wykorzystywane przez program z list. 1

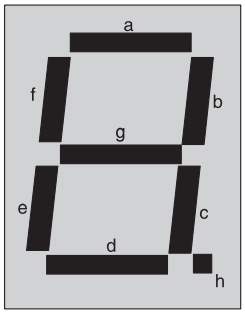
```
CZEKAJ:

   MOV R6,#0FFH      ;ustawienie początkowej wartości rejestru
   DJNZ R6,$         ;pozostanie w pętli do chwili wyzerowania R6

   RET
```

du dekodera, np. BCD/7-segmentowego wykorzystywanego w urządzeniach zbudowanych z klasycznych układów cyfrowych. Zgodnie z regułą maksymalnego upraszczania sprzętu operację dekodowania informacji wykonuje tutaj mikrokontroler. Pozwala to jednocześnie na zmniejszenie komplikacji układu, jak i na większą liczbę możliwych do wyświetlenia znaków. Przykładowy program sterujący działaniem wyświetlacza zgodnego ze schematem z rys. 19 przedstawiono na **list. 1**. Procedurę realizującą opóźnienie pokazano na **list. 2**.

Powyższa procedura opóźniająca dla systemu taktowanego częstotliwością 12 MHz zapewnia częstotliwość odświeżania wyświetlacza około 480 Hz. Wydawać się może, że jest to wartość zbyt duża, jednak należy uwzględnić fakt, że oprócz obsługi wyświetlania w głównej pętli programowej będą wykonywane też inne zadania powierzone mikrokontrolerowi, co spowoduje obniżenie częstotliwości odświeżania. W zależności od konfiguracji programu wartość opóźnienia może być dobrana w celu uzyskania określonych parametrów odświeżania. Dobrym sposobem jest również umieszczenie obsługi wyświetlacza w podprogramie obsługi przerwania licznikowego. Sytuację taką przedstawiono na **list. 3**.



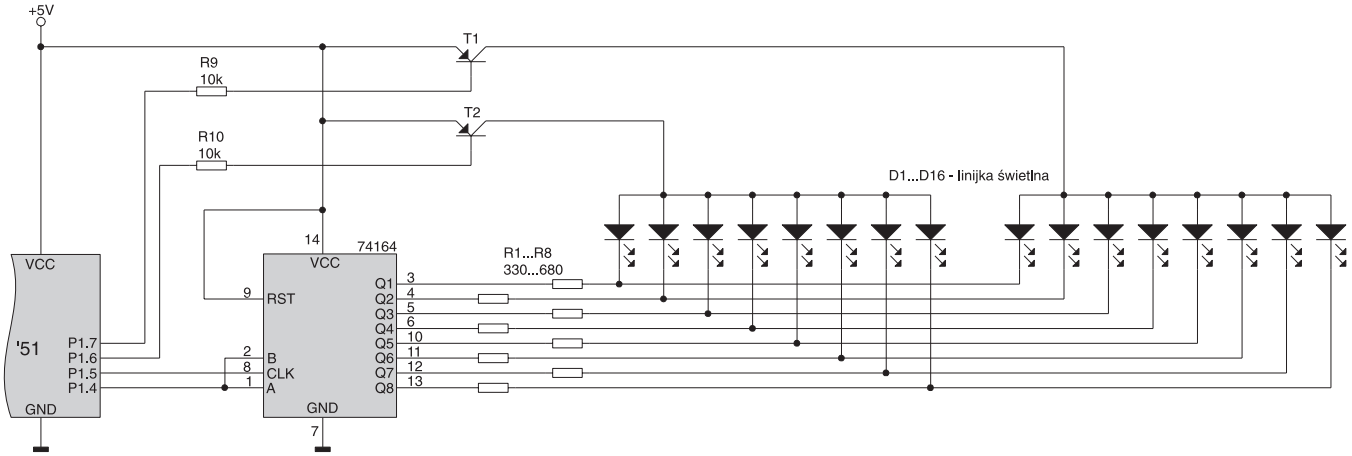
	a P1.0	b P1.1	c P1.2	d P1.3	e P1.4	f P1.5	g P1.6	h P1.7	wartość (hex)	wartość z zapaloną kropką (hex)
0	0	0	0	0	0	0	1	1	0C0H	040H
1	1	0	0	1	1	1	1	1	0F9H	079H
2	0	0	1	0	0	1	0	1	0A4H	024H
3	0	0	0	0	1	1	0	1	0B0H	030H
4	1	0	0	1	1	0	0	1	099H	019H
5	0	1	0	0	1	0	0	1	092H	012H
6	0	1	0	0	0	0	0	1	082H	002H
7	0	0	0	1	1	1	1	1	0F8H	078H
8	0	0	0	0	0	0	0	1	080H	000H
9	0	0	0	0	1	0	0	1	090H	010H

Rys. 20. Sposób tworzenia kształtów wyświetlanych znaków

Program ten zapewnia dla częstotliwości taktowania 12 MHz odświeżanie wyświetlacza z częstotliwością około 60 Hz. Częstotliwość ta jest niezależna od sposobu działania programu głównego, co pozwala na precyzyjne jej określenie.

W obu przypadkach założono, że dane przeznaczone do wyświetlenia zostały już odpowiednio przygotowane za pomocą innego podprogramu. W zasadzie nic nie stoi na przeszkodzie, aby na wyświetlaczu zapalić dowolną kombinację segmentów - najczęściej jednak chcemy, aby wyświetlacz siedmiosegmentowy wyświetlał cyfry - odpowiednie przekodowanie umożliwi nam program pokazany na **list. 4**.

Przedstawiony program korzysta z tablicy, w której zapisano stan poszczególnych wyprowadzeń portu P1 koniecznych do uzyskania na wyświetlaczu siedmiosegmentowym odpowiednich cyfr. Podane wartości liczbowe odpowiadają sytuacji, w której wyprowadzenia segmentów wyświetlacza (a, b, c, d, itd.) podłączone są kolejno do linii portu P1.0, P1.1 do P1.6. Wyprowadzenie kropki wyświetlacza (oznaczane jako *h* lub *dp*) dołączone jest do linii P1.7. Oczywiście nic nie stoi na przeszkodzie, aby linie mikrokontrolera były połączone z innymi segmentami wyświetlacza (np. w sposób upraszczający projektowanie płytki drukowanej) - zmieni się jedynie reprezentacja liczbowa wyświetlanych znaków (konieczność innego zakodowania). Sposób uzyskania odpowiednio zakodowanych liczb pokazano na **rys. 20**. Kodowanie polega na umieszczeniu w tablicy zera w przypadku zapalenia segmentu i jedynki w przypadku jego zgaszenia - otrzymany w ten sposób ciąg zer i jedynek jest ośmiobitową liczbą dwójkową, której najmłodszy bit odpowiada wyprowadzeniu P1.0. Przekształcenie na postać szesnastkową jest tylko formalnością skracającą i zwiększającą czytelność zapisu. Tak otrzymaną liczbę należy wpisać na odpowiednią pozycję w tablicy kodów umieszczoną w programie. Oprócz cyfr na wyświetlaczu sied-



Rys. 21. Multipleksowe sterowanie wyświetlaczy LED z wykorzystaniem rejestru 74164

miosegmentowym można uzyskać także wyświetlanie niektórych liter alfabetu oraz wielu znaków umownych - w zależności od zaistniałych potrzeb.

W sytuacji, gdy nie jest możliwe wykorzystanie całego portu mikrokontrolera do sterowania wyświetlaczem LED (np. z powodu braku wolnych linii lub zbyt małej ich obciążalności), dobrym sposobem na zbudowanie układu wyświetlającego jest zastosowanie rejestru szeregowo-równoległego. Schemat takiego układu przedstawiono na **rys. 21**. Wykorzystano w nim scalony rejestr szeregowo-równoległy 74164, który wraz z dwoma tranzystorami i dziesięcioma rezystorami steruje linijką świetlną zbudowaną z diod LED. Ogromną zaletą tego układu jest fakt, że do sterowania szesnastu diod LED potrzeba zaledwie 4 linii mikrokontrolera! Dodatkowo poprzez szeregowe połączenie kilku rejestrów '164 możemy sterować kolejną grupą 16 diod przypadających na każdy dołączony układ bez konieczności wykorzystywania kolejnych wyprowadzeń mikrokontrolera. Niewielka liczba wykorzystywanych linii portu wiąże się jednak z nieco bardziej rozbudowaną częścią programową w stosunku do zwykłego wyświetlacza multipleksowanego z bezpośrednim sterowaniem diod z linii portu. Choć ogólna zasada programowania nie zmienia się - nadal jest to wyświetlacz multipleksowany - to nieco inaczej przebiega procedura wystawiania sygnałów odpowiedzialnych za zaświecenie się lub

List. 3. Listing programu, w którym obsługę wyświetlacza umieszczono w podprogramie obsługi przerwania licznikowego

```
   ;POZYCJA - zmienna bajtowa przechowująca numer wyświetlanej pozycji
   ;komórki pamięci 030H do 033H przechowują dane do wyświetlenia

{program główny - część inicjująca}

   MOV POZYCJA,#1          ;pozycja 1 będzie wyświetlana jako pierwsza

   MOV TMOD,#0             ;ustawienie trybu pracy licznika
   MOV TL0,#0FFH           ;i wpisanie wartości początkowych
   MOV TL1,#00FH

   SETB EA                 ;włączenie przerwań
   SETB ET0                ;zezwolenie na przerwanie od licznika T0
   SETB TR0                ;włączenie licznika 0

{program główny}


PRZERWANIE:                ;podprogram obsługi przerwania licznika T0

   PUSH PSW                ;zapisanie na stos rejestru stanu
   PUSH ACC                ;i akumulatora

   MOV TL0,#0FFH           ;wpisanie do licznika
   MOV TH0,#00FH           ;wartości początkowych

   MOV A,POZYCJA           ;wpisanie do akumulatora aktualnej pozycji

   CJNE A,#1,NIE_1         ;sprawdzenie, czy wyświetlić pozycję pierwszą
   SETB P3.0               ;jeśli tak, to gasi wyświetlacz 4
   MOV P1,030H             ;przepisuje do portu daną dla wys. 1
   CLR P3.3                ;włącza wyświetlacz 1
   MOV POZYCJA,#2          ;następną pozycją będzie 2
   SJMP KONIEC             ;skok do końca procedury

NIE_1:
   CJNE A,#2,NIE_2         ;jeśli nie pierwsza, to sprawdza, czy druga
   SETB P3.3               ;jeśli tak, to gasi wyświetlacz 1
   MOV P1,031H             ;przepisuje daną dla wys. 2
   CLR P3.2                ;włącza wyświetlacz 2
   MOV POZYCJA,#3          ;następną pozycją będzie 3
   SJMP KONIEC             ;skok do końca procedury

NIE_2:
   CJNE A,#3,NIE_3         ;jeśli nie druga, to sprawdza, czy trzecia
   SETB P3.2               ;jeśli tak, to gasi wyświetlacz 2
   MOV P1,032H             ;przepisuje daną dla wys. 3
   CLR P3.1                ;włącza wyświetlacz 3
   MOV POZYCJA,#4          ;następną pozycją będzie 4
   SJMP KONIEC             ;skok do końca procedury

NIE_3:                     ;jeśli nie trzecia, to musi być czwarta
   SETB P3.1               ;gasi wyświetlacz 3
   MOV P1,033H             ;przepisuje daną dla wys. 4
   CLR P3.0                ;włącza wyświetlacz 4
   MOV POZYCJA,#1          ;następną pozycją będzie 1

KONIEC:                    ;koniec procedury wyświetlającej

   {inne rozkazy wykonywane w przerwaniu}

   POP ACC                 ;pobranie akumulatora
   POP PSW                 ;i rejestru stanu ze stosu
   RETI                    ;powrót z przerwania
```

List. 4. Program odpowiadający za przekodowanie liczb BCD na kody znaków wyświetlacza LED

```
   ;komórki 040H do 043H - wartości liczbowe z zakresu od 0 do 9 przeznaczone
   ;do wyświetlenia na odpowiednich pozycjach wyświetlacza
   ;komórki 030H do 033H - przekodowane dane, gotowe do wyświetlenia


PRZEKODUJ:

   MOV R0,#040H      ;wpisanie do rejestrów adresów pierwszej danej
   MOV R1,#030H      ;i pierwszego wyniku przekodowania

   MOV DPTR,#KODY    ;wpisanie do rejestru DPTR adresu tablicy kodów

   MOV R7,#4         ;wpisanie do rejestru liczby pozycji (4 pozycje)
PETLA:
   MOV A,@R0         ;przesłanie danej do akumulatora
   MOVC A,@A+DPTR    ;pobranie odpowiednich kodów z tablicy
   MOV @R1,A         ;przesłanie informacji do komórki wyniku
   INC R0            ;zwiększenie adresu komórki danych
   INC R1            ;i wyniku
   DJNZ R7,PETLA     ;pozostanie w pętli w celu czterokrotnego wykonania

   RET               ;powrót do programu głównego


KODY:                ;tablica przekodowań

   DB 0C0H           ;cyfra 0
   DB 0F9H           ;cyfra 1
   DB 0A4H           ;cyfra 2
   DB 0B0H           ;cyfra 3
   DB 099H           ;cyfra 4
   DB 092H           ;cyfra 5
   DB 082H           ;cyfra 6
   DB 0F8H           ;cyfra 7
   DB 080H           ;cyfra 8
   DB 090H           ;cyfra 9
```

List. 5. Program ilustrujący sposób wpisywania do rejestrów ‘164 kodów odpowiadających kształtom wyświetlanych znaków

```
   MOV R7,#8         ;liczba bitów do przesłania do rejestru

   MOV A,030H        ;przesłanie do akumulatora danej do wyświetlenia
                     ;(z adresu wykorzystywanego wcześniej)
PETLA:
   RLC A             ;przesuń akumulator w lewo - najstarszy bit w C
   MOV P1.4          ;wystaw bit na linię danych rejestru
   CLR P1.5          ;generacja impulsu
   SETB P1.5         ;zegarowego
   DJNZ R7,PETLA     ;pozostanie w pętli do czasu wysłania wszystkich bitów

   ;tutaj można już włączyć zasilanie danej grupy diod, np. CLR P1.7
```

zgaśnięcie wybranej diody - zamiast prostego wywołania pojedynczego rozkazu MOV konieczna staje się sekwencja przesłań (zakładamy, że po włączeniu zasilania linie portu P1 pozostają w stanie wysokim) pokazana na **list. 5**.

Paweł Hadam