C dla mikrokontrolerów 8051

część 11

W drugiej części artykułu przedstawiamy sposoby obsługi klawiatur o dużej liczbie przycisków. Proponowane rozwiązania sprzętowe oraz opisywane procedury sprowadzono praktycznie, dzięki czemu mogą być bezproblemowo stosowane we własnych aplikacjach przez Czytelników EP.

Obsługa klawiatury w C, część 2

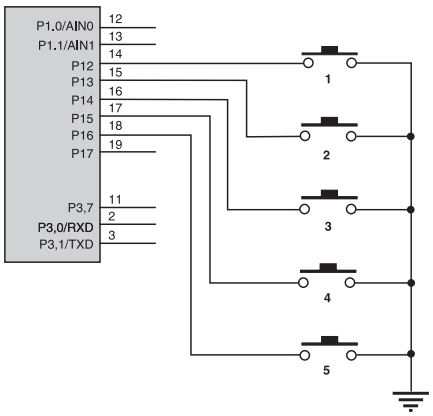
Klawiatura zbudowana z 5 przycisków

Najprostszą klawiaturę można zbudować podłączając klawisze w taki sam sposób jak w poprzednim przykładzie (**rys. 3**). Poszczególnym klawiszom można nadać funkcje adekwatne do potrzeb. W prezentowanym przykładzie ponumerowałem je od 1 do 5. Ty możesz nazwać klawisze na przykład „dodaj", „wprowadź", „pomiar" i temu podobne.

Funkcje klawiszy powinny mieć również swoje odzwierciedlenie w programie sugerując nazwy funkcji obsługi. W poprzednich przykładach zajmowaliśmy się szczegółowo sposobem obsługi pojedynczego klawisza podłączonego do mikrokontrolera. Ponieważ zasada działania nadal jest taka sama - nie ma większych różnic związanych z odczytem stanu klawiszy. Przytoczę więc przykład programu i zajmę się opisem innych rozwiązań. Program ten, podobnie jak poprzednio, to przykład pewnego rozwiązania. Jeśli będzie to twoim życzeniem, możesz indywidualnie testować każdy z bitów portu a nawet różnych portów. Wszystko zależy od sposobu połączeń używanego przez Ciebie mikrokontrolera z resztą układu. Z doświadczenia wiem jednak, że rozwiązania sprzętowe mocno wpływają na rozmiar programu. Staram się więc dobierać takie rodzaje połączeń, aby pisany program był jak najmniejszy i aby warunki jego pracy były zoptymalizowane pod kątem szybkości wykonywania i łatwości implementacji kodu.

Jak podłączyć większą liczbę przycisków

Do tego momentu stosowana przez nas liczba przycisków była na tyle mała, że nie przekraczała liczby wolnych linii wejścia/wyjścia mikrokontrolera. Co zrobić, gdy istnieje potrzeba podłączenia na przykład 8 przycisków, a do dyspozycji jest tylko 6 wolnych linii I/O? Abstrahując od rozwiązań wykorzystujących układy transmisji szeregowej i dodatkowe rejestry, przedyskutujmy kilka rozwiązań.



Rys. 3. Jedna z najprostszych metod podłączenia 5 klawiszy do mikrokontrolera

Matryca klawiszy

Matryca zbudowana z pojedynczych przycisków to jedno z najprostszych rozwiązań. Przyciski wówczas łączy się tak, aby zwierały umowne wiersze z umownymi kolumnami (**rys. 4**). W prezentowanym przykładzie wiersze klawiatury są dołączone do bitów 5, 6 i 7 portu P1, natomiast kolumny do bitów 2, 3 i 4 tego samego portu. Często buduję również układy, w których klawiatura podłączona jest jako zewnętrzna pamięć danych w ten sam sposób z tą różnicą, że kolumny stanowią bity danych a wiersze bity wybranych adresów (lub odwrotnie). Liczbę możliwych do podłączenia w ten sposób klawiszy wy-

List. 3. Program odczytuje stan klawiatury podłączonej do portu PortKey. Jest rozszerzeniem funkcji odczytu pojedynczego klawisza i działa w bardzo zbliżony sposób

```
/* prosty program demonstracyjny "odczyt klawiszy podłączonych do P1"
   klawisz włączony pomiędzy masę a bit portu P1, stan aktywny = L
   klawisze podłączone od P1.2 do P1.6; rezonator kwarcowy 8MHz */

#include <reg51.h>          //dołączenie definicji rejestrów mikrokontrolera
#include <stdio.h>          //biblioteka zawierająca funkcję printf()

#define PortKey  P1;        //definicja bitu portu klawisza
unsigned char buf;          //zmienna przechowująca stan klawiszy

//opóźnienie około 1 milisekundy dla kwarcu 8MHz
void Delay(unsigned int time)
{
   unsigned int j;

   while (time >= 1)        //wykonanie pętli FOR zajmuje około 1 msek.
   {                        //pętla jest powtarzana TIME razy
   for (j=0; j<65; j++);
   time-;
   }
}

//odczyt klawiatury podłączonej do PortKey
char KbdRead()
{
   unsigned char b;

b = PortKey;     //odczytaj stan portu klawiatury
b |= 3;          //ustaw dwa najmłodsze, nie używane przez klawiaturę bity
   return (~b);  //zwróć odczytaną wartość po zanegowaniu
}


//początek programu głównego
void main(void)
{
   while (1)                //pętla nieskończona
   {
       buf = KbdRead();
       if (buf)             //jeśli rezultat różny od 0 - sprawdź
       {
           Delay(20);       //opóźnienie 20ms
           if (buf == KbdRead())    //ponowny odczyt klawisza i akcja,
           {                //jeśli nadal wciśnięty
               if (buf && 0x04) printf("%s\n", "Klawisz 1");
               if (buf && 0x08) printf("%s\n", "Klawisz 2");
               if (buf && 0x10) printf("%s\n", "Klawisz 3");
               if (buf && 0x20) printf("%s\n", "Klawisz 4");
               if (buf && 0x40) printf("%s\n", "Klawisz 5");
           }
       }
   }
}
```

List. 4. Program obsługi klawiatury matrycowej wyposażony jest w funkcję, która zwraca numer naciśniętego przycisku

```
/* prosty program demonstracyjny "odczyt klawiatury matrycowej"
   wiersze P1.2 do P1.4, kolumny P1.5 do P1.6 (3x3 = 9 klawiszy)
   rezonator kwarcowy 8MHz */

#include <reg51.h>          //dołączenie definicji rejestrów mikrokontrolera
#include <stdio.h>          //dołączenie prototypu funkcji printf

#define PortKey  P1     //definicja bitu portu klawisza

#define Column_1 0b11111011 //definicje stanów bitów kolumn
#define Column_2 0b11110111
#define Column_3 0b11101111
#define Dummy    0b00011100 //wartość dla ustawienia bitów kolumn na "1"

#define Row_1    0b00100000 //definicje połączeń bitów wierszy
#define Row_2    0b01000000
#define Row_3    0b10000000

//opóźnienie około 1 milisekundy dla kwarcu 8MHz
void Delay(unsigned int time)
{
   unsigned int j;

   while (time >= 1)     //wykonanie pętli FOR zajmuje około 1 msek.
   {                 //pętla jest powtarzana TIME razy
       for (j=0; j<65; j++);
       time--;
   }
}

//odczyt portu klawiatury
unsigned char PortKeyRead(unsigned char column)
{
   unsigned char curr, prev;

   PortKey |= Dummy;         //ustawienie na wartość "1" bitów kolumn
   PortKey &= column;        //stan niski kolumny 1
   prev = PortKey;           //odczyt portu klawiatury
   prev &= 0xE0;             //maskowanie wszystkich bitów oprócz wierszy
   if (prev == 0xE0) return (0);   //jeśli wszystkie bity są jedynkami dla
                             //danej kolumny nie wciśnięto żadnego klawisza

   Delay(20);                // powtórny odczyt i porównanie
   curr = PortKey;
   curr &= 0b11100000;
   if (curr == prev)
   {
       while (prev = curr)  //czekaj na zwolnienia klawisza
       {
           prev = PortKey;
           prev &= 0xE0;
       }
       return(~curr);        //zamiana aktywnych bitów z 0 na 1
   }
   else return(0);           //zwróć 0 w przypadku różnic odczytów
}

//odczyt klawiatury podłączonej do PortKey
//funkcja zwraca numer wciśniętego klawisza
unsigned char KeyNumber()
{
   unsigned char b;

   b = PortKeyRead(Column_1);      //odczyt: kolumna 1 - wiersz 1, 2, 3
   if (!b)
   {
       if (b && Row_1) return(3);
       if (b && Row_2) return(6);
       if (b && Row_3) return(9);
   }
   b = PortKeyRead(Column_2);      //odczyt: kolumna 2 - wiersz 1, 2, 3
   if (!b)
   {
       if (b && Row_1) return(2);
       if (b && Row_2) return(5);
       if (b && Row_3) return(8);
   }
   b = PortKeyRead(Column_3);      //odczyt: kolumna 3 - wiersz 1, 2, 3
   if (!b)
   {
       if (b && Row_1) return(1);
       if (b && Row_2) return(4);
       if (b && Row_3) return(7);
   }
   return(0);                      //nie wciśnięto żadnego klawisza, zwróć 0
}

//początek programu głównego
void main(void)
{
   while (1)                       //pętla nieskończona
   {
       char buf = KeyNumber();     //odczyt klawisza połączony z deklaracją buf
       if (!buf) printf("%s\n",buf);   //wysłanie numeru klawisza przez RS232
   }
}
```

licza się jako wynik mnożenia liczby wierszy przez liczbę kolumn.

Wróćmy do schematu z rys. 4. Wykorzystujemy 3 bity jako wiersze i 3 jako kolumny. Maksymalnie można więc podłączyć 9 klawiszy wykorzystując 6 bitów portu P1. Prawda, że to duża oszczędność? Nieco bardziej skomplikowane układowo rozwiązanie może wykorzystywać szynę adresową i danych. Typowo, ośmiobitowy mikrokontroler z rodziny 8051 ma możliwość zaadresowania do 64 kB pamięci zewnętrznej. Daje to liczbę 16 linii adresowych (A0 do A15). Słowo danych ma długość 8 bitów (D0 do D7). Jeśli nie są wykorzystywane inne urządzenia znajdujące się w przestrzeni adresów zewnętrznych mikrokontrolera, klawiatura może mieć 16 x 8 = 128 (!) klawiszy i pracować jako pamięć zewnętrzna. Oczywiście procedura odczytu tej „pamięci" jest trochę bardziej skomplikowana.

Zasada działania klawiatury zbudowanej z matrycy przycisków jest bardzo prosta. Bity kolumn ustawione są do pracy jako linie wyjściowe, natomiast linie wierszy jako wejściowe. Mikrokontroler ustawia stan niski na jednej z linii kolumn i odczytuje stan linii wierszy. Pojawienie się stanu niskiego na jednym lub kilku bitach wierszy, zawsze oznacza w takiej sytuacji naciśnięcie jednego lub kilku klawiszy. Myślę, że najlepiej pokaże to przykład programu pokazanego na **list. 4**.

Funkcja odczytu klawiatury jest podzielona na dwie części, w celu optymalizacji kodu wynikowego. Pierwsza z nich o nazwie *PortKeyRead()*jest używana wielokrotnie dlatego też stanowi osobny fragment programu. Zatrzymajmy się przy niej na moment. Konstrukcja *PortKey |= Dummy* ustawia wyjścia portów kolumn na wartość logicznej „1". Po tym poleceniu, logiczna operacja AND (*PortKey &= Column*) ustawia stan niski na wyjściu danej kolumny. Teraz należy zbadać, czy na którymś z wejść portu odpowiadającym wierszom klawiatury, pojawił się stan niski. Odczytywany jest więc port klawiatury *prev = PortKey* i maskowane wszystkie bity, które nie mają wpływu na testowanie stanów wierszy (*prev &= 0xE0*). Jeśli powstała w ten sposób wartość jest równa 0xE0 oznacza to, że wszystkie wejścia są w stanie wysokim i żadna z linii wejściowych nie jest w stanie niskim. Odpowiada to sytuacji, w której nie wciśnięto żadnego klawisza - wówczas funkcja zwraca wartość 0.

Inaczej jest w sytuacji, gdy wartość jest różna od 0xE0. Wykonywana jest wówczas instrukcja *Delay(20)* powodująca opóźnienie na czas około 20 milisekund. Po niej ponownie odczytywany jest z użyciem tej samej metody stan klawiatury. Odczytana wartość porównywana jest ze starą i jeśli są zgodne oznacza to, że klawisz jest nadal wciśnięty. Aby uniknąć automatycznego powtarzania, program czeka na zwolnienie klawisza a następnie zwraca zanegowany stan bitów wierszy. W przeciwnym wypadku uznaję, że było to przypadkowe naciśnięcie (ewentualnie zakłócenie) i funkcja zwraca wartość 0.

PortKeyRead() wykorzystywana jest przez funkcję *KeyNumber()*. Jej rolą jest

List. 5. Program do odczytu klawiszy podłączonych za pomocą multipleksera 74157

```
/* prosty program demonstracyjny "odczyt klawiszy podłączonych przez 74157"
  wykorzystane są bity P1.0 do P1.3 oraz P3.7 do sterowania wyborem
   połówki odczytywanego bajtu; rezonator kwarcowy 8MHz */

#include <reg51.h>          //dołączenie definicji rejestrów mikrokontrolera
#include <stdio.h>          //biblioteka zawierająca funkcję printf()

#define PortKey P1;         //definicja bitu portu klawisza
sbit A_B = P3^7;            //wybór połówki bajtu odczytywanej przez 74157


//opóźnienie około 1 milisekundy dla kwarcu 8MHz
void Delay(unsigned int time)
{
   unsigned int j;

   while (time >= 1)        //wykonanie pętli FOR zajmuje około 1 msek.
   {                        //pętla jest powtarzana TIME razy
       for (j=0; j<65; j++);
       time--;
   }
}

//odczyt klawiatury podłączonej do PortKey
char KbdRead()
{
   unsigned char L_nibble, H_nibble;

   A_B = 1;                 //odczyt starszej połówki bajtu (klawisze parzyste)
   H_nibble = PortKey;
   H_nibble &= 0x0F;
   H_nibble <<= 4;
   A_B = 0;                 //odczyt młodszej połówki bajtu (klawisze niep.)
   L_nibble = PortKey;
   L_nibble &= 0x0F;

   return (~(H_nibble | L_nibble));   //zwróć odczytaną wartość po zanegowaniu
}

//początek programu głównego
void main(void)
{
   while (1)                //pętla nieskończona
   {
       buf = KbdRead();
       if (buf)             //jeśli rezultat różny od 0- sprawdź
       {
           Delay(20); //opóźnienie 20ms
           if (buf == KbdRead())    //ponowny odczyt klawisza i podjęcie
                                    //akcji, jeśli nadal wciśnięty
           {
               if (buf && 0x01) printf("%s\n", "Klawisz 1");
               if (buf && 0x02) printf("%s\n", "Klawisz 2");
               if (buf && 0x04) printf("%s\n", "Klawisz 3");
               if (buf && 0x08) printf("%s\n", "Klawisz 4");
               if (buf && 0x10) printf("%s\n", "Klawisz 5");
               if (buf && 0x20) printf("%s\n", "Klawisz 6");
               if (buf && 0x40) printf("%s\n", "Klawisz 7");
               if (buf && 0x80) printf("%s\n", "Klawisz 8");
           }
       }
   }
}
```

obliczenie numeru naciśniętego klawisza. Kolejno odczytywane są poszczególne kolumny i jeśli zwrócona wartość jest różna od 0, rozpatrywane są bity zwróconej zmiennej. Tak więc wiersz po wierszu, kolumna po kolumnie przeglądana jest cała klawiatura. Metoda ta nosi nazwę odpytywania (z angielskiego *pooling*).

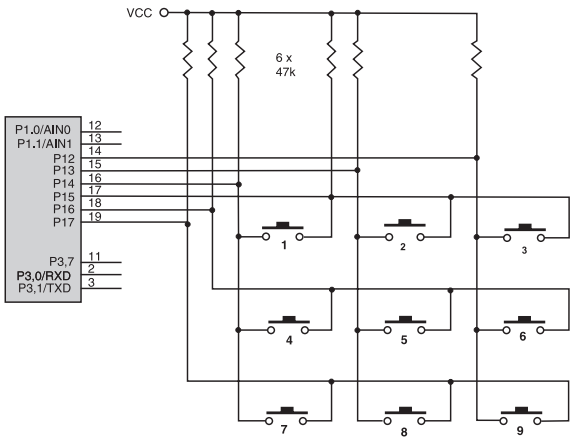
Program główny wywołuje funkcję *KeyNumber()* i używa jej do wyznaczenia numeru wciśniętego klawisza. Można w nim spotkać konstrukcję *char buf = KeyNumber()* - nie używałem jej do tej pory. Zmienna w języku C musi być zadeklarowana przed pierwszym użyciem. To jest właśnie dosłowny przykład, jednak nie polecam go do użytku. W pewnym momencie możesz przestać panować na ilością zmiennych i przez to również wykorzystaniem zasobów mikrokontrolera. Lepsza jest jawna deklaracja na początku programu czy funkcji, aniżeli ukryta w kodzie programu. Nie mniej jednak dobrze jest wiedzieć, że można to zrobić. Zwrócony numer klawisza zamieniany jest przez funkcję *printf()* na wartość typu łańcuch znaków i wysyłany przy pomocy UART.

Wykorzystanie dodatkowego układu multipleksera

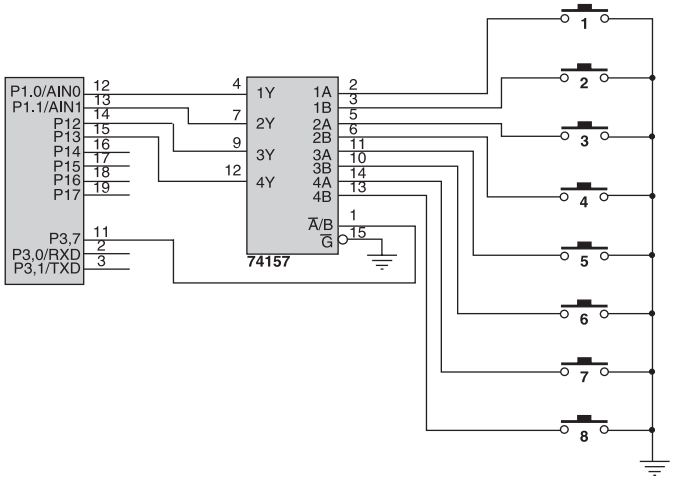
Łatwą metodą rozszerzenia ilości podłączonych do mikrokontrolera klawiszy jest użycie dodatkowego zewnętrznego układu multipleksera. Przykład takiego rozwiązania umieszczono na **rys. 5**. Wykorzystano w nim układ 74157. Jest to poczwórny multiplekser 2x1. Poziom logiczny na wyprowadzeniu 1 (!A/B) wybiera aktywne wejście z dwóch grup: (1A...4A/1B...4B) i dołącza je do wyjść układu (1Y...4Y), a tym samym do portu wejściowego mikrokontrolera.

Proste rozwiązania lokalnych klawiatur nie wymagają rezystorów *pullup*, ponieważ układy z serii TTL traktują wejście nie podłączone jako znajdujące się w stanie wysokim. W celu wybrania odpowiedniej czwórki przycisków, posługuję się wyprowadzeniem P3.7 mikrokontrolera. Dwie czterobitowe połówki składane są do postaci jednego bajtu, który to następnie rozpatrywany jest przez program z **list. 5**.

Jacek Bogusz
jacek.bogusz@ep.com.pl



Rys. 4. Matryca zbudowana z przycisków to jedno z najprostszych rozwiązań stosowanych przy zwiększaniu liczby klawiszy



Rys. 5. Wykorzystanie zewnętrznego układu multipleksera dla zwiększenia liczby podłączonych do portu mikrokontrolera klawiszy