C dla mikrokontrolerów 8051

część 12

W ostatniej części cyklu poświęconego obsłudze klawiatur za pomocą mikrokontrolerów programowanych w języku C przedstawiamy najbardziej ekonomiczny - z punktu widzenia wydajności pracy mikrokontrolera - sposób obsługi klawiatury: za pomocą przerwań sprzętowych.

Obsługa klawiatury w C, część 3

Przerwania i klawiatura

Dotychczas we wszystkich prezentowanych przykładach klawiatury były obsługiwane z wykorzystaniem techniki zwanej odpytywaniem lub przeglądaniem (*pooling*). Teraz pokażę przykład klawiatury, która obsługiwana będzie na żądanie, po naciśnięciu klawisza. Do jej wykonania wykorzystam przerwanie zewnętrzne INT0 mikrokontrolera 8051, jak to pokazano na **rys. 6**.

Podobnie jak poprzednio wykorzystujemy przyciski zajmujące bity od 0 do 4 portu P1. W sumie daje to pięć klawiszy. Identycznie jak w poprzednich przykładach stanem aktywnym jest stan niski - jego pojawienie się na wejściu mikrokontrolera oznacza wciśnięcie klawisza. Do poszczególnych bitów portu podłączone są rezystory *pull-up*, których zadaniem jest zwiększenie odporności mikrokontrolera na zakłócenia. Jeśli używasz niewielkiej, lokalnej klawiatury podłączonej blisko układu mikrokontrolera (przewód nie dłuższy niż 15...20 cm) - możesz ich nie stosować. Wejścia portów podłączone są do wejść ośmiowejściowej bramki NAND (7430). Jej zadaniem jest wygenerowanie sygnału przerwania w momencie naciśnięcia klawisza. Na wyjściu bramki NAND (negacja iloczynu) pojawia się jednak sygnał o fazie przeciwnej niż nam potrzebna: w sytuacji, gdy wszystkie wejścia bramki są w stanie wysokim (czyli nie jest wciśnięty żaden z klawiszy), wyjście bramki jest w stanie niskim. Gdy na dowolnym z wejść pojawi się stan logiczny niski, to wyjście bramki przyjmuje stan wysoki. Odpowiada to narastającemu zboczu sygnału na wejściu przerwania INT0 po wciśnięciu klawisza. Opadające zbocze, które wyzwala przerwanie (jest to związane z zasadą działania mikrokontrolera z rodziny 8051, który wymaga opadającego zbocza lub poziomu niskiego, aby uruchomić procedurę obsługi przerwania), pojawi się dopiero po zwolnieniu przycisku. Niestety, jest to zbyt późno, ponieważ nie będziemy w stanie sprawdzić, który przycisk został wciśnięty. W związku z tym wymagane jest użycie układu inwertera dla odwrócenia fazy sygnału. Faktycznie realizowana jest więc funkcja AND, a nie NAND.

Można również pokusić się o wykonanie bramki AND za pomocą diod (**rys. 7**). Jest to rozwiązanie bardzo oszczędne, jeśli rozpatrywać je pod kątem ceny użytych do konstrukcji elementów. Może nie tak „eleganckie“ jak z wykorzystaniem układów scalonych, jednak tanie i równie skuteczne.

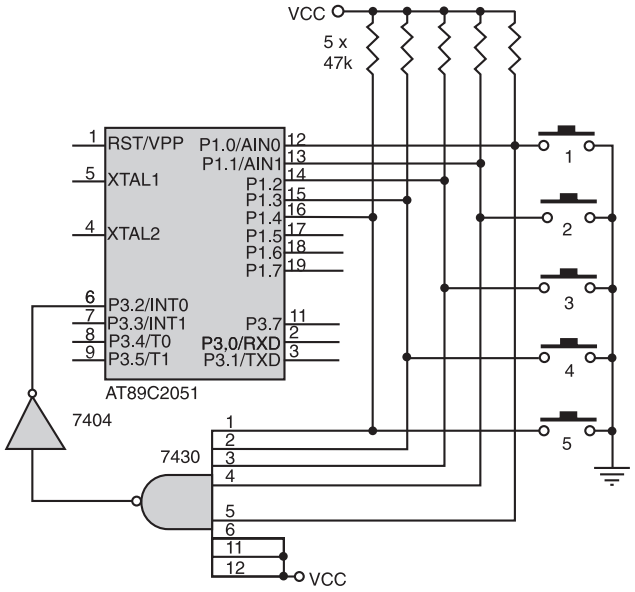
Nowatorska jest w tym przykładzie tylko metoda. Nie angażujmy mikrokontrolera w przeglądanie stanu klawiatury, zajmujemy się nią tylko wówczas, gdy jest to konieczne. Na życzenie możliwe jest również wyłączenie obsługi klawiszy poprzez prostą blokadę przyjmowania przerwania z INT0 (zerowanie bitu IE0). Metoda ta jak każda ma swoje wady i zalety: nie będziemy ich tu dyskutować, zajmijmy się programem obsługi.

Najważniejszą częścią programu jest funkcja obsługująca przerwanie. Zajmuje się ona rozpoznaniem, który z klawiszy został naciśnięty, ustawia bity zmiennej status. Podobnie jak poprzednio funkcja *printf* wysyła, korzystając z UART, numer wciśniętego klawisza.

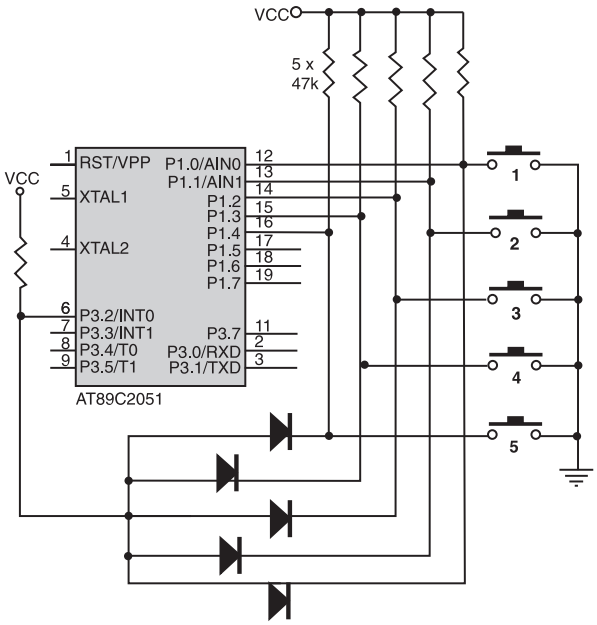
Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl

List. 6. Przykładowy program obsługujący klawiaturę w przerwaniu INT0

```
/* prosty program demonstracyjny „przerwanie generowane po wciśnięciu klawisza”
   wykorzystane są bity P1.0 do P1.4 oraz wejście zewnętrznego przerwania INT0
   rezonator kwarcowy 8MHz */

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

#define PortKey P1              //definicja bitu portu klawisza
#define Key_1  0b00000001       //maski dla poszczególnych bitów klawiszy
#define Key_2  0b00000010
#define Key_3  0b00000100
#define Key_4  0b00001000
#define Key_5  0b00010000
#define mask   0b00011111       //maska bitów klawiatury dla operacji logicznych

unsigned char status = 0;       //status, ustawiany w obsłudze przerwania


//odczyt klawiatury podłączonej do PortKey
//jest to funkcja obsługi przerwania używająca banku rejestrów 1
void KbdRead() interrupt 1 using 1
{
   int i;
   unsigned char p;

   p = PortKey;                 //odczyt bitów portu
   p = ~p & mask;
   for (i=0; i<1300; i++);      //pauza 20 ms dla rezonatora 8MHz
   status = PortKey;            //ponowny odczyt bitów klawiatury dla weryfikacji
   status = ~status & mask;
   if (p != status) status = 0;    //ustawienie zmiennej status
}

//początek programu głównego
void main(void)
{
   PortKey |= ~mask;            //ustawienie linii klawiatury
   EX0 = 1;                     //zezwolenie na przyjmowanie przerwań INT0
   EA = 1;                      //załączenie przerwań
   while (1)                    //pętla nieskończona
   {
       if (status)
       {
           if (status && Key_1) printf(„%s\n”,”Klawisz 1");
           if (status && Key_2) printf(„%s\n”,”Klawisz 2");
           if (status && Key_3) printf(„%s\n”,”Klawisz 3");
           if (status && Key_4) printf(„%s\n”,”Klawisz 4");
           if (status && Key_5) printf(„%s\n”,”Klawisz 5");
           status = 0;
       }
   }
}
```



Rys. 6. Schemat elektryczny klawiatury generującej sygnał przerwania po naciśnięciu dowolnego przycisku



Rys. 7. Schemat elektryczny klawiatury generującej sygnał przerwania po naciśnięciu dowolnego przycisku z bramką AND zbudowaną z diod