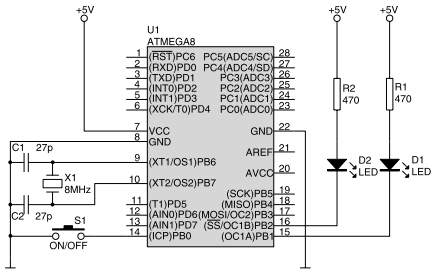
Obsługa elementów stykowych: przyciski, klawiatury, impulsatory, część 1

Z większością budowanych systemów mikroprocesorowych należy się komunikować. A dokładniej mówiąc należy mieć możliwość zmiany jego parametrów pracy lub możliwość wprowadzania do niego nowych danych. Komunikacja z systemem mikroprocesorowym może odbywać się w różny sposób. Ale najczęściej do zmiany jego parametrów pracy i danych służą przyciski lub klawiatury. O tym jak je obsługiwać za pomocą Bascoma piszemy w artykule.

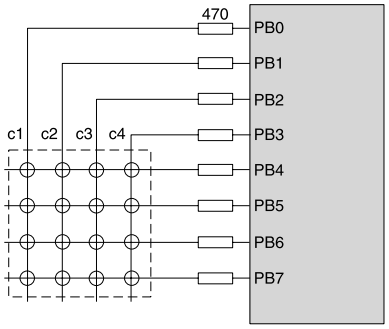
Wymienione elementy są często stosowane w systemach mikroprocesorowych, gdzie umożliwiają łatwą zmianę parametrów lub wprowadzenie danych do systemu. Obsługa przez mikrokontroler przycisków czy klawiatur nie jest trudna, ale przy wykorzystywaniu tego typu elementów należy mieć na uwadze, że styki przycisków podczas ich naciskania drgają przez pewien czas, a nie dają - jak można by się spodziewać - stabilnego stanu. Tego typu drgania mogą powodować błędną pracę systemu mikroprocesorowego. Nie ma z tym większego problemu, gdyż drgania styków można w bardzo prosty sposób usunąć programowo poprzez zastosowanie opóźnienia. Ponieważ z różnego typu przyciskami czy klawiaturami występuje wiele kłopotów przy próbie ich obsługi, więc w artykule zostaną przedstawione przykłady obsługi: od pojedynczych przycisków (z przypisaną jedną lub dwoma funkcjami) poprzez klawiatury matrycowe (ważne ze względu na koszty klawiatury AT) aż do często wykorzystywanych impulsatorów, których użycie upraszcza znacznie obsługę urządzenia, a bez wątpienia można je zaliczyć do elementów stykowych.

Pojedyncze przyciski

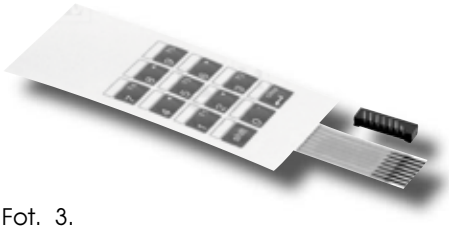
W przypadku niewielkiej liczby przycisków występujących w systemie mikroprocesorowym (poniżej 6) można je dołączyć bezpośrednio do linii portów mikrokontrolera. W przypadku wykorzystywania większej liczby przycisków lepszym rozwiązaniem są klawiatury matrycowe (przyciski połączone w matrycę), gdyż do ich obsługi jest potrzebna mniejsza liczba linii mikrokontrolera. Na **rys. 1** przedstawiono schemat z dołączonym do mikrokontrolera jednym przyciskiem. Diody LED D1 i D2 zastosowane zostały jako elementy sygnalizujące działanie przycisku S1. Mikrokontroler powinien reagować na naciśnięcie przycisku S1, czyli na stan niski na jego linii PB0. Aby przycisk poprawnie pracował linia ta musi zostać skonfigurowana jako wejściowa z włączonym rezystorem podciągającym. Można także zastosować zewnętrzny rezystor podciągający. W przypadku gdyby na wejście mikrokontrolera po przyciśnięciu przycisku był podawany stan wysoki (druga końcówka przycisku dołączana do linii zasilającej, a nie do masy) należy linię mikrokontrolera PB0 ustawić jako wejściową, przy czym należy zastosować zewnętrzny rezystor ściągający do masy. Nie należy włączać wewnętrznego rezystora podciągającego. W Bascom AVR do obsługi pojedynczych przycisków można wykorzystać dedykowaną instrukcję *Debounce*. Na **list. 1** przedstawiono przykładowy program, który po każdym naciśnięciu przycisku S1 gasi i zapala diodę D1. Pierwszym parametrem instrukcji *Debounce* jest linia portu wejściowego, do której podłączony został przycisk. Aliasowi S1 w programie przypisano bit 0 rejestru wejściowego *Pinb*. Poprzez ustawienie bitu 0 rejestru *Portb* włączony został rezystor podciągający do linii PB0. linii sterującej diodą LED D1 także został przypisany alias *Led1*. Aliasy niewątpliwie upraszczają odwoływanie się np. do rejestrów mikrokontrolera oraz ułatwiają analizę programu. Drugi parametr instrukcji *Debounce* określa na jaki stan przycisku ma odbywać się reakcja. W przykładzie po naciśnięciu przycisku podawany jest stan 0, czyli drugim parametrem jest wartość 0. W wymienionym przypadku z przyciskiem podłączonym do linii zasilającej byłaby to wartość 1. Następnym parametrem instrukcji *Debounce* jest etykieta, do której nastąpi skok po przyciśnięciu przycisku. Zamiast do etykiety może to być także skok do podprogramu (z czego skorzystano w przykładzie), ale należy wtedy dodatkowo zawrzeć w instrukcji *Debounce* parametr *SUB*. W przykładzie po naciśnięciu przycisku S1 nastąpi skok do podprogramu *Obs_s1*, w której zmieniany jest na przeciwny stan linii sterującej diodą LED. Czyli po każdym naciśnięciu przycisku S1 dioda będzie zapalana i gaszona. Instrukcja *Debounce* nie wstrzymuje działania programu i sama dba o eliminacje drgań styków przycisku. Czas potrzebny na eliminację drgań styków, z którego korzysta *Debounce* można określić za pomocą instrukcji *Config Debounce*. W programie czas opóźnie-



Rys. 1.



Rys. 2.



Fot. 3.

nia został określony na 30 ms, choć domyślnie wynosi 25 ms, co także jest wystarczające. Gdyby przy takim opóźnieniu dawały znać drgania styków można to opóźnienie zwiększyć. Jeśli by było potrzebne wstrzymywanie programu aż do naciśnięcia przycisku, to przed instrukcją *Debounce* można wykorzystać instrukcje wstrzymującą *Bitwait*. W programie każdy pojedynczy przycisk można obsłużyć za pomocą instrukcji *Debounce*, ale można także to zrobić z wykorzystaniem instrukcji warunkowych (*If... Then*) pamiętając o potrzebie eliminacji drgań styków. Jest także możliwe przypisanie do jednego przycisku wielu funkcji, które będą rozpoznawane przykładowo po czasie jego naciśnięcia. Zgodnie ze schematem z rys. 1 niech krótkie przyciśnięcie przycisku naprzemiennie zapala diodę D1, a dłuższe (ok. 2 sekundy) niech naprzemiennie zapala diodę D2. Program realizujący taki dwufunkcyjny przycisk S1 przedstawiono na **list. 2**. Diodzie D2 także przypisany został alias *Led2*. Tu do obsługi przycisku S1 także została wykorzystana instrukcja *Debounce*, której czas opóźnienia eliminację drgań styków wynosi domyślnie 25 ms. Po wykryciu naciśnięcia przycisku następuje skok do podprogramu *Obs_s1*, w którym liczony jest czas przyciśnięcia przycisku poprzez zmienną *Opozn*. Czas naciśnięcia przycisku liczony jest w pętli *Do...Loop* poprzez zliczanie przez zmienną *Opozn* opóźnień o czasie 10 ms do czasu, aż przycisk S1 zostanie puszczony lub zmienna *Opozn* osiągnie wartość 200, co będzie oznaczało, że przycisk został przyciśnięty na czas ok. 2 sekund (10 ms · 200). Następnie jest sprawdzany warunek i jeśli *Opozn* wynosi 200 to stan diody D2jest zmieniany na przeciwny, a jeśli mniej (przycisk został przyciśnięty na krócej niż 2 sekundy) zmieniany jest na przeciwny stan diody D1. Należy zauważyć, że dioda D1 będzie zmieniała swój stan dopiero po puszczeniu przycisku, a dioda D2 po przytrzymaniu naciśniętego przycisku przez czas co najmniej 2 sekund. Do jednego przycisku na przedstawionym przykładzie można także przypisać większą liczbę funkcji, które będą rozpoznawane po czasie naciśnięcia. Można także przypisywać przyciskowi inne funkcje w zależności od liczby jego naciśnięć w danym czasie, tak jak to ma miejsce np. w myszkach komputerowych. Podwójne przyciśnięcie przycisku może na przykład realizować inną funkcję. Działanie tego typu można zrealizować w podprogramie wywoływanym przez instrukcję *Debounce*, w której należy czekać na następne naciśnięcie tego przycisku przez określony czas. Czyli należy wykryć jego puszczenie i ponowne naciśnięcie w zadanym czasie.

List. 1.

```
'Program obslugi jednego przycisku S1 za po-
moca instrukcji Debounce
'Przycisk S1 umozliwia właczenie i wyłaczenie
diody LED

$regfile = „m8def.dat”
                'informuje kompilator o pliku
dyrektyw mikrokontrolera
$crystal = 8000000
                  'informuje kompilator o
czŕstotliwosci oscylatora taktuj¦cego mikro-
kontroler

Config Pinb.0 = Input
'linia PB0 jako wejsciowa
Config Pinb.1 = Output
'linia PB1 jako wyjsciowa
Config Debounce = 30
                'okrecla czas opoznienia (30
ms) instrukcji debounce, kiedy nie będzie
użyta ta opcja,

                  'domyślnym opóźnieniem jest
25 ms
Led1 Alias Portb.1
                  'przypisanie nazwie Portb.1
nazwy Led1
S1 Alias Pinb.0
                  'przypisanie nazwie Pinb.0
nazwy S1

Set Portb.0
                'dolaczenie do linii PB0 rezy-
stora podciagajacego

Do

 'poczatek nieskoczonej petli Do-Loop
'niesko˜czona pŕtla Do...Loop
  Debounce S1 , 0 , Obs_s1 , Sub
                'je'li nacisniety przycisk S1,
to skok do podprogramu Obs_s1
Loop

End
                'koniec programu

Obs_s1:
                'podprogram Obs_s1
  Toggle Led1
                  'zmiana na przeciwny stanu
wyjscia sterujacego dioda LED D1
Return
'powrot z podprogramu
```

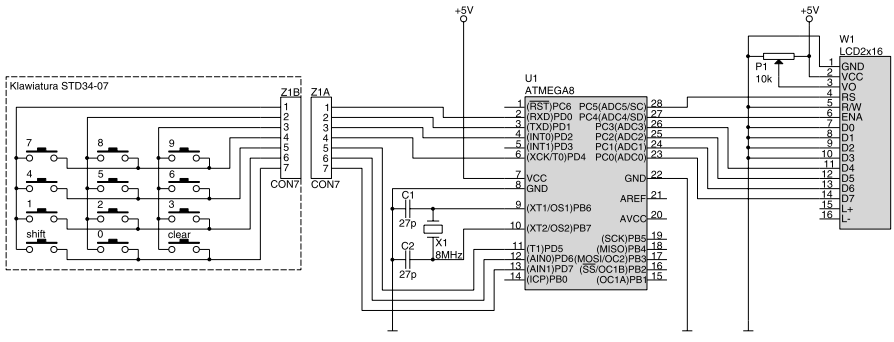
List. 2.

```
'Program obslugi jednego przycisku S1 za po-
moca instrukcji Debounce
'Krotkie przycisniecie S1 umozliwia właczenie
i wyłaczenie diody LED1
'natomiast przycisniecie przycisku S1 na czas
ok 2 sekund umozliwia
'wlaczenie lub wylaczenie diody LED2

$regfile = „m8def.dat”
                'informuje kompilator o pliku
dyrektyw mikrokontrolera
$crystal = 8000000
                  'informuje kompilator o
czŕstotliwosci oscylatora taktuj¦cego mikro-
kontroler

Config Pinb.0 = Input
'linia PB0 jako wejsciowa
Config Pinb.1 = Output
'linia PB1 jako wyjsciowa
Config Pinb.2 = Output
'linia PB2 jako wyjsciowa

Dim Opozn As Byte
                'pomocnicza zmienna zliczajaca
opoznienia

Led1 Alias Portb.1
                  'przypisanie nazwie Portb.1
nazwy Led1
Led2 Alias Portb.2
                  'przypisanie nazwie Portb.2
nazwy Led2
S1 Alias Pinb.0
                  'przypisanie nazwie Pinb.0
nazwy S1

Set Portb.0
                'dolaczenie do linii PB0 rezy-
stora podciagajacego

Do

 'poczatek nieskoczonej petli Do-Loop
'niesko˜czona pŕtla Do...Loop
  Debounce S1 , 0 , Obs_s1 , Sub
                'je'li nacisniety przycisk S1,
to skok do podprogramu Obs_s1
Loop

End
                'koniec programu

Obs_s1:
                'podprogram Obs_s1
  Opozn = 0
                 'zerowanie zmiennej liczacej
opoznienia
  Do
                'poczatek petli Do-loop
    Waitms 10
                'opoznienie 10 ms
    Incr Opozn
                 'zwiekszenie o jeden wartosci
zmiennej opoznienia
  Loop Until Opozn = 200 Or S1 = 1
                'jesli S1 puszczony lub zmien-
na opozn osiagnie wartosc 200 to opusc petle
  If Opozn = 200 Then
                'jesli opozn=200 (200*10 ms)=2
sekundy to
    Toggle Led2
                 'zmiana na przeciwny stanu
wyjscia sterujacego dioda LED D2
  Else
                'w przeciwnym razie
    Toggle Led1
                 'zmiana na przeciwny stanu
wyjscia sterujacego dioda LED D1
  End If
Return
'powrot z podprogramu
```

Klawiatury matrycowe

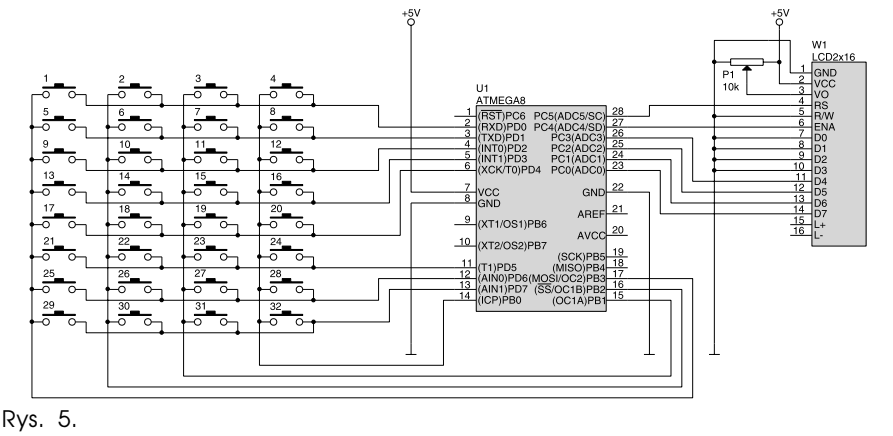
W przypadku jeśli w systemie występuje więcej niż 6 przycisków, łączenie ich wprost do indywidualnej linii mikrokontrolera staje się rozwiązaniem nieoptymalnym. W tym przypadku swoją zaletę ujawniają klawiatury matrycowe, które do swojego działania potrzebują mniej linii mikrokontrolera, ale są trudniejsze w obsłudze. Bascom AVR do obsługi klawiatur matrycowych (w której przyciski połączono w matrycę, podobnie jak się łączy wyświetlacze pracujące w trybie multipleksowym) ma dedykowaną funkcję *Getkbd()*, która nie wstrzymuje działania programu i umożliwia obsługę klawiatur 4x4 lub 4x6, czyli klawiatury 16-przyciskowej i 24-przyciskowej. Ale może to być także klawiatura matrycowa o mniejszej liczbie przycisków. Nie zostaną wtedy wykorzystane wszystkie linie danego portu mikrokontrolera, gdyż funkcja *Getkbd()* na swoje cele rezerwuje cały 8-bitowy port, z którego połowa pracuje jako wejście, a polowa jako wyjście, co przedstawiono na **rys. 2**.

W przykładzie wykorzystana została klawiatura STD3407 o matrycy 3x4 (12



Rys. 4.

przycisków), której wygląd przedstawiono na **rys. 3**. Na **rys. 4** przedstawiono sposób podłączenia do mikrokontrolera tego typu klawiatury, którą można obsłużyć za pomocą funkcji *Getkbd()*. Ponieważ funkcja *Getkbd()* wykorzystuje do swego działania cały port mikrokontrolera, niewykorzystana została jedna linia PD3. Na **list. 3** został przedstawiony program obsługujący taką klawiaturę, i której kody naciśniętych przycisków są wyświetlane na wyświetlaczu LCD. Przy czym są to kody przekonwertowane, bowiem kod naciśniętego przycisku nie musi odpowiadać znakowi przypisanemu danemu klawiszowi. Po przekodowaniu odczytanych kodów, przyciskowi „1” będzie odpowiadał kod 1, przyciskowi „2” kod 2 itp. Aby móc skorzystać z funkcji *Getkbd()* należy skonfigurować port z którego będzie ona korzystać przy odczycie klawiatury (port do którego została przyłączona klawiatura) za pomocą instrukcji *Config Kbd*. W przykładzie klawiatura została dołączona do portu PD, przy czym parametr *Debounce* określa czas odczytu pomiędzy kolejnymi odczytami stanu klawisza, który ustalono na 20 ms. Opcjonalny parametr *Delay* określa czas opóźnienia w milisekundach jaki zostanie wprowadzony po wykryciu naciśniętego przycisku. W przykładzie czas ten został określony na 100 ms. Dodatkowo można podać w tej instrukcji parametry *ROWS*, *ROWS5* i *ROWS6*, które określają linie portu dodatkowych dwóch wierszy klawiatury przy obsłudze klawiatur 4x6. W przykładzie funkcja *Getkbd()*, gdy nie został naciśniety żaden przycisk zwraca wartość 16 (przy matrycy 4x6 byłaby to wartość 24). Po każdym odebraniu kodu klawisza jest on przekodowywany na podstawie tablicy *Tab_klw*. Po odebraniu kodu klawisza określa on pozycje stałej, która zostaje pobrana z tablicy i jest nowym kodem naciśnietego przycisku. Nowy kod klawisza jest pobierany za pomocą funkcji *Lookup* i zapisywany w zmiennej *Konw_klaw*. Dla przykładu po naciśnięciu przycisku „1” zostanie odebrany kod 6, czyli z tablicy zostanie odczytana wartość 1, która następnie zostaje wyświetlona na LCD jako kod naciśniętego klawisza. W przypadku większej matrycy przycisków, jeśli wartości odczytach klawiszy będą z przedziału od 0 do 255, tego typu tablica, może być nieekonomiczna ze względu na to, że odczytany kod klawisza reprezentuje pozycje nowego kodu w tablicy. Inny sposób przekodowania kodów przycisków zostanie przedstawiony w następnym przykładzie i polega na znalezieniu w tablicy danego kodu (czyli tablica kodów będzie wielkości odpowiadającej ilości przycisków matrycy). Odczyt przycisków za pomocą funkcji *Getkbd()* można także umieścić w przerwaniu zgłaszanym co pewien czas od timera.



Rys. 5.

Należy wtedy do minimum zmniejszyć opóźnienie *Delay* w instrukcji *Config Kbd*. W przypadku większych lub mniejszych matryc klawiatur można stworzyć własne procedury ich obsługi. Dla przykładu zostanie przedstawiona obsługa klawiatury 8x4 w przerwaniu zgłaszanym przez Timer0 co ok. 20 ms. Obsługa klawiatury w przerwaniu ma spore zalety, gdyż odbywa się w tle działania programu głównego, w którym należy tylko sprawdzać czy nie odebrano nowego kodu naciśniętego klawisza. Schemat dołączenia klawiatury 8x4 do mikrokontrolera przedstawiono na **rys. 5**. Na **list. 4** przedstawiono program obsługujący klawiaturę matrycową 8x4. Oczywiście tego typu zadanie można rozwiązać na wiele innych sposobów, a w przykładzie jest tylko jeden z nich. Port PD został skonfigurowany jako wyjściowy, natomiast port PB jako wejściowy z rezystorami podciągającymi, które są niezbędne do działania tego typu klawiatury. Obsługa klawiatury odbywa się w dwóch etapach - kod naciśniętego klawisza otrzymuje się dopiero po obsłużeniu dwóch przerwań *Mult_kl* od Timer0. Czyli następują dwa sprawdzenia naciśniętego przycisku z opóźnieniem 20 ms, co eliminuje drgania styków. Jeśli pierwsze sprawdzenie będzie identyczne z drugim to zostaje obliczony i zwrócony kod naciśniętego przycisku. A dokładnie, po wywołaniu przerwania zostają skanowane kolejno wiersze klawiatury (instrukcja *Rotate* załącza po każdym jej wywołaniu kolejny wiersz klawiatury poddawany skanowaniu) i jeśli odczytana wartość kolumny będzie różna od 15 (4 bardziej znaczące bity portu wejściowego PB są maskowane) następuje wcześniejsze opuszczenie pętli *For*, gdyż wykryto naciśnięcie przycisku. Wartość zmiennej *I* wskazuje na numer wiersza, w którym wykryto naciśnięcie przycisku. Odczytana wartość z kolumn matrycy jest zapisywana w pierwszym elemencie tablicy *Temp*. Jeśli *J* nie jest jeszcze równe 2, następuje opuszczenie procedury obsługi przerwania i przy następnym przerwaniu zostaje znów skanowana klawiatura. I jeśli *J* będzie równe 2 oraz wartości komórek dwuelementowej tablicy *Temp* będą sobie równe, i będzie to oznaczać, że na pewno jest naciśnięty dany klawisz a nie jest to zakłócenie. Następnie zostaje obliczony kod klawisza na podstawie odczytanej wartości z kolumn i wartości zmiennej *I* która jest zmienna licznikowa pętli *For*. Czyli zostanie uwzględniony w kodzie klawisza numer skanowanego

wiersza klawiatury. W przypadku braku naciśnięcia przycisku zwracana jest wartość 15. Kod naciśniętego przycisku jest zapisywany w zmiennej *Przycisk*. Jak widać na schemacie przyciskom klawiatury zostały przypisane wartości od 1 do 32. Aby uzyskać tego typu kody naciśniętych przycisków, także wykorzystano do przekodowania tablice stałych *Tab_kod*. Ale przekodowywanie działa inaczej niż w poprzednim przykładzie. W tablicy zostały zamieszczone kody klawiszy zwracanych w przerwaniu Timer0. Za pośrednictwem funkcji *Lookdown* zwracana jest pozycja w tablicy odczytanego kodu klawisza, która będzie jego nowym kodem. Nowy kod przycisku zostaje zapisany do zmiennej *Kod_p* typu *integer*. Po przekodowaniu przy braku naciśniętego przycisku będzie zwracana wartość 33. Przykładowo po przyciśnięciu przycisku oznaczonego „2” zostanie w przerwaniu odebrany kod 11. W tablicy ma on pozycje 2 wiec zostanie do zmiennej *Kod_p* zapisana wartość 2, która jest nowym kodem naciśniętego przycisku. Przy takim odczycie danych z tablicy wystarczy w niej jedynie zapisać wartości kodów klawiszy na odpowiednich pozycjach, czyli będzie ona miała wartość zbliżoną do liczby przycisków klawiatury. Przekodowany kod naciśniętego jest wyświetlany na wyświetlaczu LCD. Bez problemów procedurę obsługi matrycy w przedstawionym programie można przystosować do obsługi większych lub mniejszy matryc klawiatur. Gdy w systemie mikroprocesorowym występują multipleksowane wyświetlacze LED, można obsługę klawiatur matrycowych zrealizować przy okazji ich obsługi. Podczas multipleksowania wyświetlaczy może się przy okazji odbywać odczyt klawiatury, której wiersze lub kolumny są załączane poprzez tranzystory załączające wyświetlacze. Można w ten sposób zredukować jeszcze bardziej liczbę linii mikrokontrolera potrzebnych do obsługi klawiatury matrycowej.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl

List. 3.

```
'Program obslugi klawiatury matrycowej 4x3 z
wykorzystaniem dedykowanych
'instrukcji getkbd()
'Na wyswietlaczu przedstawione zostaja kody
nacisnietych klawiszy,
'ktore zostaja wczesniej przekodowane na od-
powiadajace im kody
'tzn klawisz o znaku „1” bedzie posiadal kod
1 a nie przykladowo 13

$regfile = „m8def.dat”
                'informuje kompilator o pliku
dyrektyw mikrokontrolera
$crystal = 8000000
                  'informuje kompilator o
czestotliwosci oscylatora taktujacego mikro-
kontroler

Config Lcd = 16 * 2
              'konfiguracja typu wyswietlacza
LCD
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 =
Portc.2 , Db6 = Portc.1 , Db7 = Portc.0 , E
= Portc.4 , Rs = Portc.5       'konfiguracja
linii komunikacyjnych z wyswietlaczem
Config Kbd = Portd , Debounce = 20 , Delay =
100             'konfiguracja portu sterujace-
go klawiatura oraz potrzebnych opoznien

Dim Klawisz As Byte
              'zmienna do ktorej zapisywany
jest kod klawisza
Dim Konw_klaw As Byte
               'zmienna do ktorej zapisywany
jest przekodowany kod klawisza

Cursor Off
              'wylaczenie kursora
Cls
              'czysci LCD
Lcd „Przycisk: „
              'Wyswietlenie na LCD napisu

Do
               'poczatek nieskonczonej petli
Do-Loop
  Klawisz = Getkbd()
                'skanowanie klawiatury i
pobranie kodu klawisza

                'przy braku nacisnietego
klawisza zwracana jest wartosc 16
  Konw_klaw = Lookup(klawisz , Tab_klw)
'przekodowanie odczytanej wartosci nacisnie-
tego klawisza
  Locate 1 , 10
              'kursor na 10 kolumne wiersza
pierwszego
  Lcd Konw_klaw ; „  „
              'wyswietlenie przekonwertowa-
nego kodu nacisnietego klawisza
Loop
End
              'koniec programu

Tab_klw:
               'tablica konwersji kodow
klawiszy
Data 11 , 0 , 10 , 16 , 3 , 2 , 1 , 16 , 6 ,
5 , 4 , 16 , 9 , 8 , 7 , 16 , 16
```

List. 4.

```
'Program obslugi klawiatury matrycowej 8x4
(32 przyciski) w przerwaniu zglaszanym
'od przepelnienia Timer0 co 1/(8 MHz/1024/
150) = ok. 20 ms
'oczywiscie przy czestotliwosci oscylatora
8MHz
'Kod nacisnietego przycisku wyswietlany jest
na wyswietlaczu LCD
'Przy braku nacisnietego przycisku zwracana
jest wartosc 33,
'natomiast przyciski maja kody od 1 do 32

$regfile = „m8def.dat”
                'informuje kompilator o pliku
dyrektyw mikrokontrolera
$crystal = 8000000
                  'informuje kompilator o
czestotliwosci oscylatora taktujacego mikro-
kontroler
Config Portd = Output
'port D jako wyjsciowy
Config Portb = Input
'port B jako wejsciowy
Config Lcd = 16 * 2
               'konfigurowanie typu wyswie-
tlacza LCD
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 =
Portc.2 , Db6 = Portc.1 , Db7 = Portc.0 , E
= Portc.4 , Rs = Portc.5       'konfigurowa-
nie linii


               'mikrokontrolera, do ktorych
dolaczono LCD
Config Timer0 = Timer , Prescale = 1024
'konfigurowanie Timer0 jako timer z podzialem
preskalera przez 1024

On Timer0 Mult_kl
                  'konfigurowanie przerwania
od przepelnienia Timer0, skok do podprogramu
Mult_kl

Dim Temp(2) As Byte
               'dwu-elementowa tablica zmien-
neych pomocniczych Temp
Dim J As Byte
               'zmienna licznikowa
Dim Przycisk As Byte
                'zmienna, do ktorej bedzie
wpisywany kod nacisnietego przycisku (bez
konwersji)
Dim I As Byte
               'zmienna licznikowa
Dim Kod_p As Integer
'zmienna do ktorej bedzie wpisywany przekon-
wertowany kod przycisnietego przycisku
Dim M_odcz As Byte
                'zmienna pomocniczaprzechowu-
jaca odczytany stan linii portu B

Portb = 255
               'dolaczenie rezystorow podcia-
gajacych do portu B
```

List. 4. cd

```
Portd = 254
               'ustawienie stanu portu
wyjsciowego D na 11111110, czyli wyzerowanie
linii PD.0

Enable Interrupts
'odblokowanie globalnego systemu przerwan
Enable Timer0
'odblokowanie przerwania od przepelnienia
Timer0
Load Timer0 , 150
               'wpisanie do licznika wartosci
poczatkowej

Cursor Off
               'wylaczenie kursora na ekranie
LCD
Cls
               'czyszczenie ekranu LCD
Lcd „Przycisk: „
               'wyswietlenie napisu
Do
               'pętla glowna programu
 Kod_p = Lookdown(przycisk , Tab_kod , 65)
'pobranie z tablicy pozycji odczytanego kodu
klawisza (konwersja kodow

                'odpowiadajacych klawiasza)
- przetoworziny kod zapisywany jest do zmien-
nej kod_p

                 'przy braku nacisnietego
przycisku zwracana jest wartosc 33
 Locate 1 , 11
                'kursor do pierwszego wiersza
i na pozycje 11
 Lcd Kod_p ; „  „
                  'wyswietlenie na LCD
przekonwertowanego kodu przycisku oraz dwoch
dodatkowych spacji
 Waitms 100
               'opoznienie 100 ms
Loop
End
               'koniec programu

Mult_kl:
               'podprogram przerwania, w kto-
rym multipleksowana jest matryca przyciskow
  Load Timer0 , 150
               'wpisanie do licznika wartosci
poczatkowej
  Portd = 254
                'wartosc poczatkowa stanu
linii
portu D (wyzerowana tylko linia PD.0)
  For I = 0 To 7
               'petla wykonywana 8 razy
    M_odcz = Pinb And &B00001111
               'odczyt stanu przyciskow w da-
nym wierszu (odczyt tylko 4 mniej znaczacych
linii portu)
     If M_odcz <> 15 Then
               'jesli odczytana wartosc rozna
od 15 to
      Exit For
                'wczesniejsze opuszczenie
petli For
    End If
     Rotate Portd , Left , 1
                'zalaczenie kolejnego wiersza
skanowanej klawiatury
  Next I
                'zwiekszenie wartosci I o
jeden
  Incr J
               'zwiekszenie o jeden wartosci
J
  Temp(j) = M_odcz
               'przepisanie stanu przyciskow
do tablicy Temp z indeksem J
  If J = 2 Then
               'jest J rowne 2 to
    J = 0
               'wyzerowanie zmiennej J
    If Temp(1) = Temp(2) And Temp(1) <> 15
Then             'Jesli wartosci w tablicy
Temp sa rowne i wartosc w Temp(1)<>15 to
      Przycisk = 15 - Temp(1)
'odjecie od 15 wartosci odczytanego przycisku
i zapisanie nowego kodu do zmiennej Przycisk
      I = I * 10
                'pomnozenie wartosci I przez
10
      Przycisk = Przycisk + I
                'podanie wartosci I do nie-
przetworzonego kodu nacisnietego przycisku
    Else
               'w przeciwnym razie
      Przycisk = 15
                'zapisanie do zmiennej przy-
cisk wartosci 15
    End If
  End If
Return
'powrot z podprogramu przerwania

Tab_kod:
                'tablica konwersji kodow
klawiszy
Data 1 , 11 , 21 , 31 , 41 , 51 , 61 , 71 , 2
, 12 , 22 , 32 , 42 , 52 , 62 , 72
Data 4 , 14 , 24 , 34 , 44 , 54 , 64 , 74 , 8
, 18 , 28 , 38 , 48 , 58 , 68
Data 78 , 15
```