Ćwiczenie 3

Klawiatura i wyświetlacz LED

Program = Data structures + Algorithm

-- Niklaus Wirth

Warszawa, kwiecień 2022

1. Cel ćwiczenia

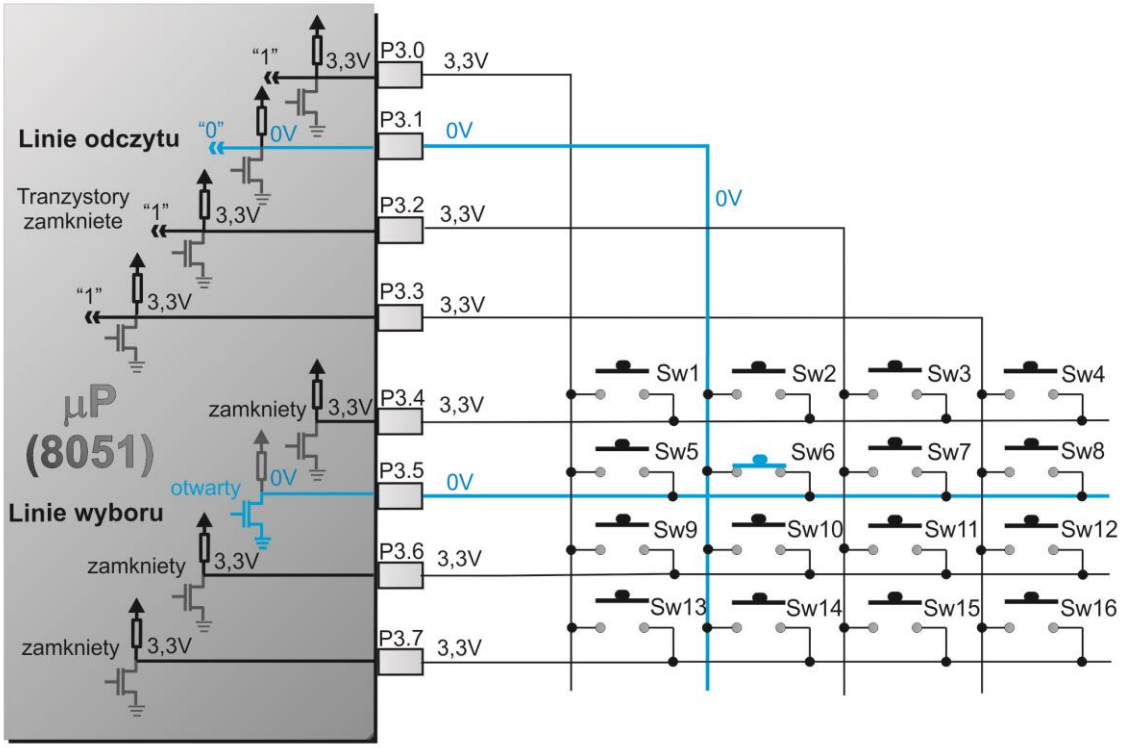
Celem ćwiczenia jest zapoznanie się studentów z programową obsługą elektromechanicznej 16-przyciskowej klawiatury w układzie matrycowym. Procedura ta powinna współgrać z obsługą oprogramowywanego w ramach poprzedniego ćwiczenia wyświetlacza LED. Dodatkowym zadaniem będzie realizacja funkcjonalności umożliwiającej wprowadzania do programu wielkości numerycznych w dziesiętnym systemie liczbowym.

W ramach ćwiczenia studenci będą mieli za zadanie oprogramowanie mikrokontrolera i wspomnianych peryferiów w taki sposób, aby pełniły one funkcje prostego urządzenia wejścia, przy pomocy którego będzie można wprowadzić zadaną przez użytkownika wartość liczbową, która zostanie wykorzystana podczas realizacji następnego ćwiczenia laboratoryjnego.

Językiem programowania wykorzystywanym w trakcie ćwiczenia będzie język C.

2. Opis klawiatury

Klawiatura umieszczona na płycie systemu uruchomieniowego EXTB-060/2 zawiera 16 przycisków i zrealizowano ją w układzie matrycowym - 4 kolumny x 4 rzędy. Schemat elektryczny podłączenia klawiatury do wyprowadzeń mikrokontrolera przedstawiono na Rys. 1.



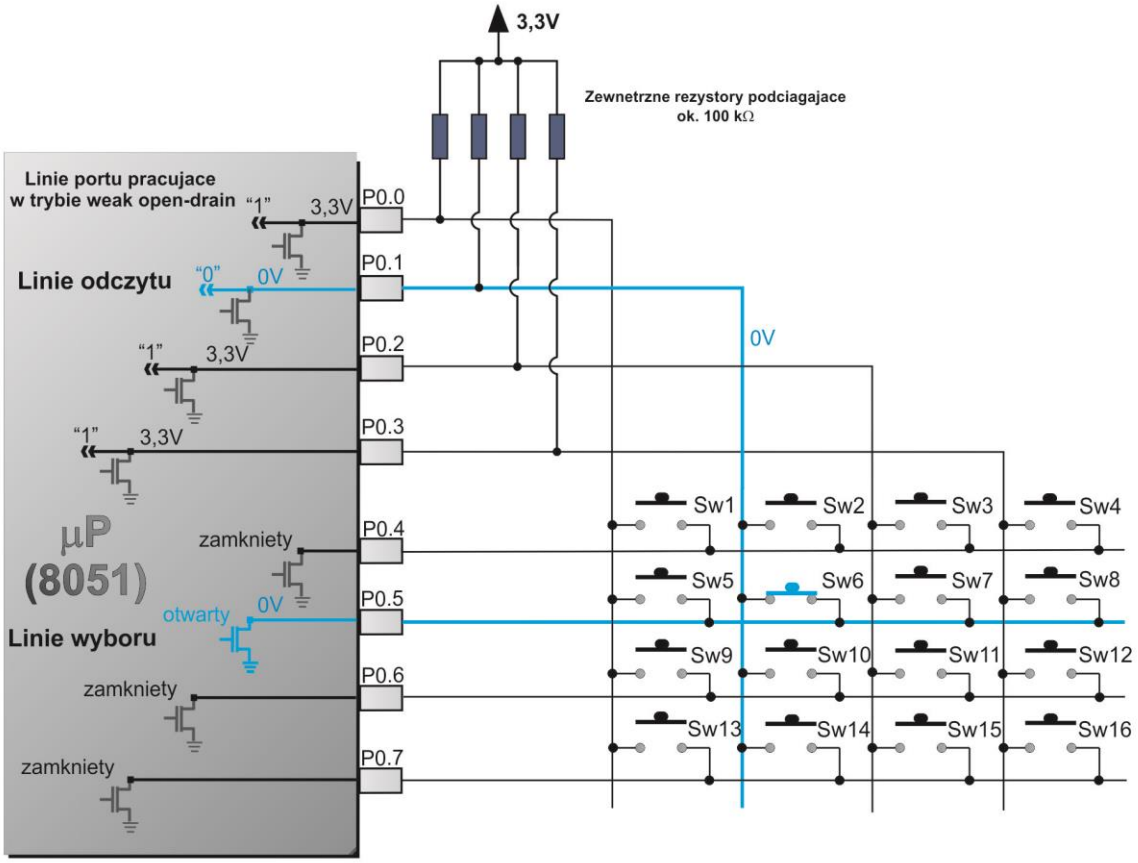
Rys. 1. Schemat podłączenia klawiatury matrycowej do portu mikrokontrolera.

Na powyższym rysunku prezentowany jest przypadek wciśnięcia przycisku „SW1”. W tej sytuacji stan „0” z wyjścia linii piątej portu trzeciego (P3.5) „propaguje” się do wejścia na linii pierwszej tego samego portu (P3.1). To które linie klawiatury będą pełniły rolę wejść, a które wyjść zależy od przyjętej konwencji. W omawianym przypadku wyjściami są linie

P3.4 - P3.7, a wejściami linie P3.0 – P3.3. Klawiatura jest symetryczna wobec czego można sobie bez problemu wyobrazić sytuację odwrotną, w której wyjściami będą młodsze linie portu P3, natomiast wejściami starsze linie tego samego portu.

Należy pamiętać o poprawnej konfiguracji stopnia wyjściowego poszczególnych linii portu wykorzystywanego do obsługi klawiatury. Linie wejściowe (P3.0 - P3.3) powinny być skonfigurowane do pracy w trybie ze słabą jedynką logiczną (*ang. weak pull-up*). Nie można ich konfigurować w trybie z silnej jedynki logicznej (*ang. push-pull*) gdyż mogłoby dojść do zwarcia zasilania z liniami wyjściowymi (P3.4 - P3.7) w sytuacji, w której wymuszają one zero. Zwarcie takie mogłoby doprowadzić do trwałego uszkodzenia procesora. Linie wyjściowe mogą pracować w trybie ze słabą jedynką logiczną, przy czym nie jest to obligatoryjne. Można bowiem skonfigurować je w tzw. trybie *open-drain*. Programowa obsługa klawiatury będzie jednak tożsama.

Innym możliwym sposobem konfiguracji linii portu obsługi klawiatury jest układ prezentowany na Rys. 2.



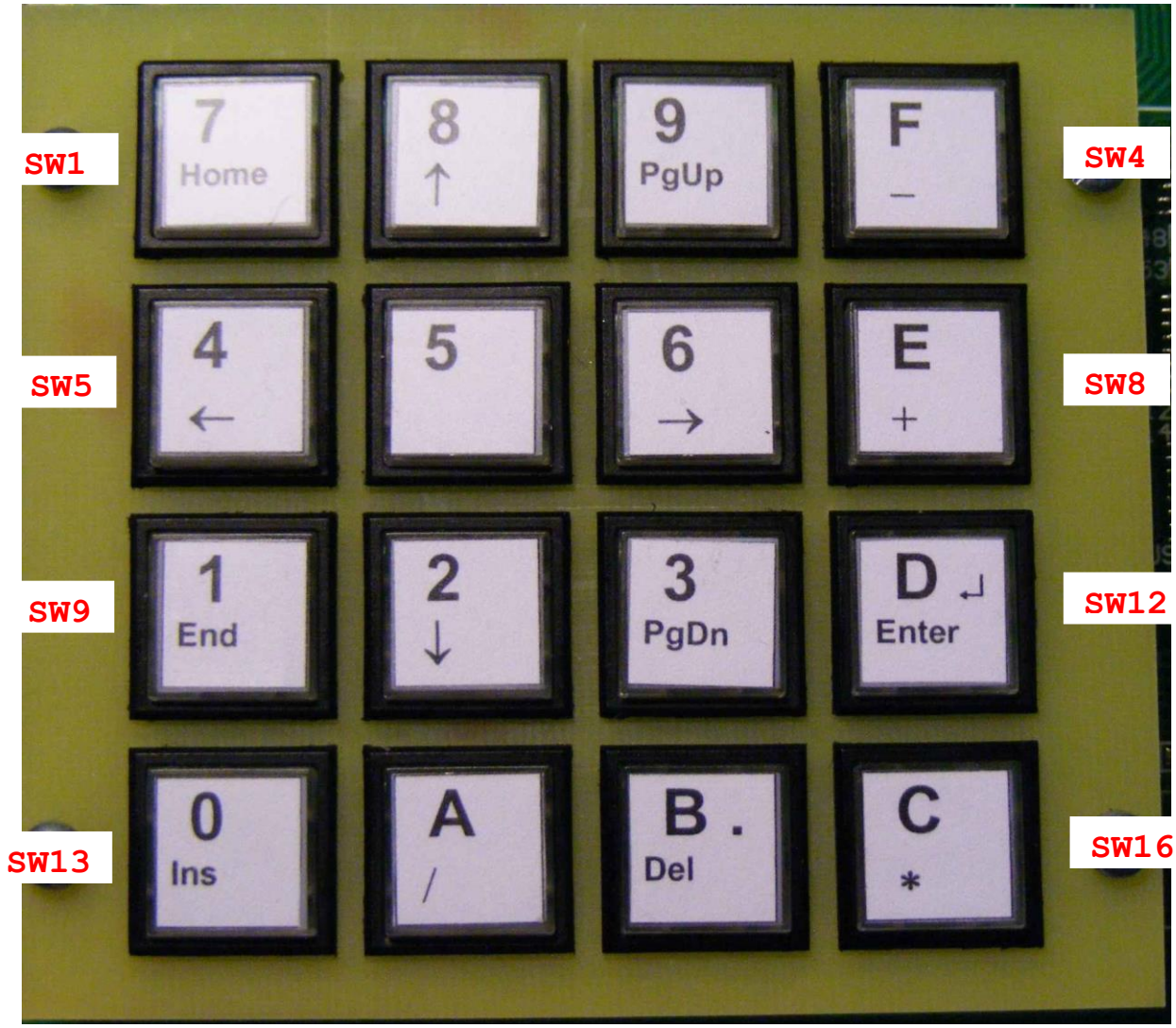
Rys. 2. Schemat podłączenia klawiatury matrycowej do portu mikrokontrolera pracującego w trybie *open-drain.*

W sytuacji prezentowanej na powyższym rysunku cały port do obsługi klawiatury jest skonfigurowany w trybie *open-drain*. Może to również dotyczyć przypadków wykorzystania mikrokontrolerów, w których są porty o nierekonfigurowanych stopniach wyjściowych, pracujących tylko w trybie *open-drain.* W takiej sytuacji do obsługi klawiatury konieczne jest podłączenie rezystorów podciągających do zasilania na liniach wejściowych wykorzystywanego portu.

W przypadku mikrokontrolera oprogramowywanego w ramach zajęć laboratoryjnych (Silabs C8051F060) jest możliwe dowolne przekonfigurowanie linii każdego z portów. Domyślnie pracują one jednak w trybie ze słabą jedynką logiczną (Rys. 1) i w takim trybie należy je pozostawić. Jest to tryb najbezpieczniejszy dla mikrokontrolera, ponieważ maksymalna wydajność prądowa linii ze słabą jedynką logiczną jest mocno ograniczona. Uniemożliwia to tym samym przypadkowe uszkodzenie mikrokontrolera w sytuacji zwarcia jedynki logicznej z logicznym zerem.

3. Programowa obsługa klawiatury

Na Rys. 3. widoczna jest klawiatura matrycowa wykorzystywana w ramach ćwiczenia wraz z deskryptorami SWx przypisanymi do poszczególnych klawiszy.

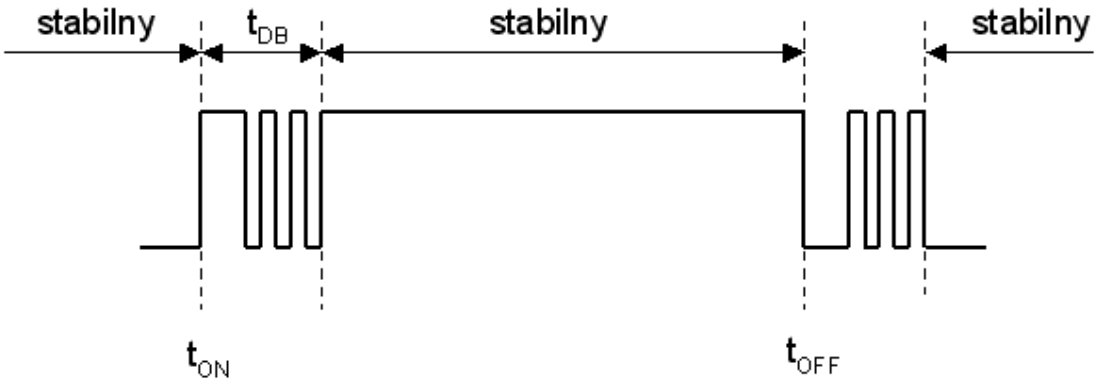


Rys. 2. Widok klawiatury.

Stan klawiatury odczytać można w procesie przemiatania zerem na starszej połówce portu P3 z jednoczesnym odczytem młodszej połówki tego samego portu. Linie portu P3 połączone są z kolumnami klawiatury (P3.0 - P3.3) i służą do odczytu stanu kolumny

klawiatury (linie odczytu - wejścia). Linie portu P3 połączone z wierszami klawiatury (P3.4 - P3.7) służą do wyboru odczytywanego wiersza (linie wyboru - wyjścia). Uaktywnienie wiersza następuje poprzez podanie zera logicznego na odpowiadającej mu linii portu P3 (P3.4 - P3.7). Jednocześnie na tych czterech liniach może się pojawić tylko jedno zero. W przeciwnym razie wystąpią błędy odczytu. Jeżeli klawisz aktywnego wiersza nie jest wciśnięty, to na odpowiadającej mu linii odczytu będzie stan wysoki. Jeżeli klawisz został wciśnięty, pojawi się stan niski.

Klawisze użyte w klawiaturze zestawionej na płycie EXTB-060/2 są elementami elektromechanicznymi o bardzo dobrej jakości, dlatego też trudno zaobserwować drżenie styków podczas zmiany stanu klawisza. W zwykłych klawiszach efekt ten można łatwo zauważyć (Rys. 4). Jest to oczywiście zjawisko niepożądane i do poprawnego działania klawiatur opartych o tego typu przyciski niezbędne są mechanizmy filtrujące (*ang. debouncing)* zrealizowane w postaci programowej lub z wykorzystaniem zewnętrznych obwodów analogowych. W przypadku obsługi klawiatury matrycowej na płycie EXTB-060/2 nie ma potrzeby implementacji zarówno w warstwie sprzętowej jak i programowej mechanizmów filtracji drgania styków.



Rys. 4. Ilustracja działania zwykłego klawisza.

Skanowanie klawiatury najwygodniej będzie zrealizować wykorzystując przerwanie czasowe licznika T0 lub T1. Wybrany licznik należy zaprogramować tak, aby zgłaszał przerwania z okresem wymaganym do poprawnego działania procedury przemiatania/skanowania ($T_{interrupt}$ ~ kilka ms). W kolejnych przerwaniach zgłoszonych przez licznik należy wystawiać „0” na kolejnych wierszach klawiatury z jednoczesnym odczytem wszystkich kolumn. Odczytana wartość inna niż 0x0F będzie oznaczała wciśnięcie klawisza. Kod wciśniętego klawisza będzie kombinacją półbajtu z ustawionym w danym momencie zerem na liniach wyjściowych oraz półbajtu odczytanego z linii wejściowych.

- ✓ Kod wybranego/wciśniętego klawisza to zestawienie półbajtu linii wyjścia z liniami wejścia.
- ✓ Kod wybranego/wciśniętego klawisza powinien jednoznacznie wskazywać jaki bajt sterujący powinien zostać wysłany na wyświetlacz LED (tj. umieszczony w odpowiedniej komórce pamięci, z której pobierane są dane do wyświetlacza LED).
- ✓ Zapamiętanie kodu klawisza poprzednio wybranego oraz obecnie wciśniętego umożliwia wykrycie długiego przytrzymania wciśniętego przycisku – jest to kluczowe przy wczytywaniu kolejnych wartości symbolizowanych przez przyciski.

W celu konfiguracji układu licznikowego Timer0 należy napisać funkcję inicjalizującą, która będzie ustawiała odpowiednie wartości rejestrów konfiguracyjnych TMOD, TCON, IE oraz TH0 i TL0. Można jednak skorzystać z gotowej funkcji Timer0_Init, której ciało znajduje się w pliku Template_3.c:

```
void Timer0_Init(void)
{
        SFRPAGE   = TIMER0_PAGE;      // Switch SFRPAGE to TMR0 PAGE
        TMOD     &= 0xF0;
        TMOD     |= 0x01;
        TMR0      = 65535-5000;       // Init value
        TCON     |= 0x10;
        IE       |= 0x02;             // Enable Timer0 interrupts
        SFRPAGE   = CONFIG_PAGE;
        return;
}
```

Powyższa funkcja korzysta z 16-bitowego przypisania (`TMR0 = 65535-5000`) do rejestrów licznikowych TH0 i TL0. Do tego celu należy na początku programu zdefiniować 16-bitową zmienną w przestrzeni SFR, której adres będzie zgodny z adresem rejestru TL0:

```
        sfr16 TMR0    = 0x8B;
```

Procedura obsługi przerwania od układu licznikowego Timer0 powinna być napisana w oparciu o poniższy szablon (proszę przepisać go do programu):

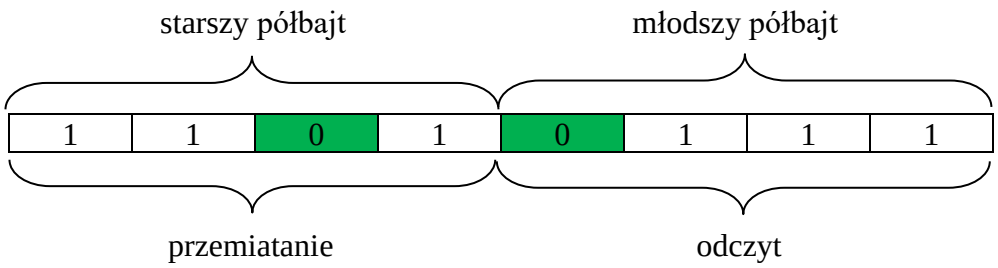
```
void TIMER0_ISR (void) interrupt 1
{
      unsigned char old_SFRPAGE;
      old_SFRPAGE = SFRPAGE;
      SFRPAGE     = TIMER0_PAGE;    // Switch SFRPAGE to CONFIG PAGE

      TMR0 = 65535 - 50000;

      /* User code */

      SFRPAGE = old_SFRPAGE;        // Restore SFR
      return;
}
```

Do detekcji stanu klawiatury (detekcji wciśniętego klawisza) wystarczy jeden bajt, który pełni jednocześnie rolę słowa sterującego (Rys. 5). Młodsza jego część przechowuje aktualny stan klawiszy (wciśnięty - 0, puszczony - 1), a starsza zawiera wektor wymuszający podany na wiersze klawiatury. W przytoczonej już wcześniej procedurze skanowania należy przemiatać zerem starszą część bajtu sterującego i wystawiać go na port P3, a następnie odczytywać stan tego portu. Wartość odczytanego portu to właśnie **kod klawisza**. Sytuacja, w której bajt odczytany zawiera same jedynki w swojej młodszej części oznacza, że żaden z 16 przycisków klawiatury nie został wciśnięty. Pojawienie się więcej niż jednego zera w młodszej części odczytywanego bajtu oznacza wciśnięcie więcej niż jednego przycisku. Taka sytuacja może nastąpić i powinna być przez program odpowiednio obsłużona (czytaj zasygnalizowana) lub „poprawnie" zignorowana.



Rys. 5. Słowo sterujące – kod wciśniętego klawisza.

- ✓ Młodsza część portu P3 (linie P3.0 - P3.3) to wejścia służące do odczytu stanu kolumn;
- ✓ Starsza część portu P3 (linie P3.4 - P3.7) to wyjścia służące do wyboru wiersza , **aktywne '0' logiczne**;
- ✓ Na liniach P3.4 - P3.7 może pojawić się w danym momencie tylko jedno '0' logiczne, które po wciśnięciu klawisza/y przechodzi do wejść P3.0 - P3.3.

W wyniku wciskania poszczególnych klawiszy ich symbole powinny być „przekazywane" do oprogramowanego poprzednio wyświetlacza LED. Oznacza to, że procedura obsługi klawiatury matrycowej powinna współgrać z procedurą obsługi wyświetlacza współdzieląc z nią zasoby pamięci w postaci odpowiednio zdefiniowanej zmiennej tablicowej, np.:

```
unsigned char led_display[5] = {0x7F, 0x00, 0x00, 0x00, 0x00};
```

lub:

```
unsigned char xdata led_display[5] = {0x7F, 0x00, 0x00, 0x00, 0x00};
```

Pierwsza ze wspomnianych procedur powinna wpisywać do tej tablicy dane w postaci symboli wybieranych klawiszy, a druga powinna te dane pobierać i wysterowywać nimi

wyświetlacz LED. Implementacja takiego mechanizmu umożliwi tym samym wyświetlanie symbolu klawisza na wyświetlaczu LED.

- ✓ Dane przekazywane do wyświetlacza LED powinny być przechowywane w tablicy zdefiniowanej w pamięci **RAM** lub **XRAM**. Nie może to być tablica w pamięci kodu ponieważ jej zawartość musi się zmieniać.
- ✓ Procedura obsługi multipleksowanego wyświetlacza LED powinna być w dużej mierze zgodna z procedurą napisaną w ramach ćwiczenia nr 2 i tym samym powinna być oparta o przerwanie układu licznikowego Timer3.

Jedna z procedur obsługi klawiatury, w najprostszym możliwym trybie pracy, powinna realizować następujące kroki/instrukcje:

a. przypisz wartość portu **P3** do zmiennej **port_value**,
b. wykonaj operację iloczynu logicznego zmiennej **port_value** z wartością **0x0F**
c. sprawdź czy otrzymana wartość jest różna od **0x0F**
   - ✓ jeżeli tak to prześlij kod klawisza do pamięci tablicowej wyświetlacza LED i wyjdź z procedury obsługi przerwania,
   - ✓ jeżeli nie to:
     - inkrementuj **keyboard_counter** (zakres zmienności 0 – 4),
     - przypisz do starszej części portu P3 zero przesunięte o wartość **keyboard_counter**,
     - ustaw **port_value** na wartość domyślną **0xFF**,
     - wyjdź z procedury obsługi przerwania.

Poniższy algorytm powinien zostać zaimplementowany w całości w procedurze obsługi układu licznika Timer0, której szablonowy listing przedstawiono na poprzednich stronach niniejszej instrukcji.

4. Zadania do wykonania

(5 pkt) Napisać program odczytujący stan całej klawiatury wykorzystując przerwanie od układu licznikowego T0. Symbol wciśniętego klawisza (zgodny z jego opisem) powinien być wyświetlany na dowolnej pozycji 7-segmentowego multipleksowanego wyświetlacza LED. Przytrzymanie przycisku powinno skutkować ciągłym wyświetlaniem jego symbolu (np. ‘1’ lub ‘A) – aż do momentu zwolnienia przycisku. Jednoczesne wciśnięcie więcej niż jednego klawisza nie powinno skutkować wystąpieniem nieoczekiwanych efektów. Nie musi być to również sygnalizowane dodatkowym komunikatem o błędzie, chociaż implementacja takiego mechanizmu będzie mile widziana. Zadanie powinno być realizowane jako twórcze rozwinięcie programu tworzonego podczas ćwiczenia nr 2. Można także skorzystać z pliku Template_3.c dostarczonego przez prowadzącego.

(2 pkt) Zmodyfikować uprzednio napisany program tak, aby do wyświetlania kodu wciśniętego klawisza nie było konieczne jego przytrzymanie. Po zwolnieniu przycisku kod powinien być nadal wyświetlany na wybranej pozycji wyświetlacza LED.

Wybranie/wciśnięcie nowego klawisza powinno zmodyfikować wyświetlany znak na zgodny z nowo wciśniętym przyciskiem.

(7 pkt) Zmodyfikować uprzedni program tak, aby umożliwiał wczytanie z klawiatury czterocyfrowej liczby dziesiętnej z zakresu 0 – 9999. Liczba powinna być sukcesywnie, wraz z wciskaniem kolejnych klawiszy, wyświetlana na wyświetlaczu LED począwszy od lewej (DISP0). Zatwierdzenie wprowadzonej liczby (wczytanie jej do pamięci/zmiennej) powinno być realizowane wraz z wciśnięciem przycisku „D/Enter” oraz sygnalizowane włączeniem diody D4. Po przekroczeniu ostatniej pozycji, wyświetlacz powinien być czyszczony oraz powinna być wprowadzana liczba kolejna. Klawisze oznaczone jako A, B, C, E, F nie powinny być brane pod uwagę w procesie wprowadzania wartości – powinny być ignorowane.