Politechnika Łódzka
Instytut Fizyki

Laboratorium elektroniki

Ćwiczenie E59

Programowanie mikrokontrolerów PIC16 i urządzeń peryferyjnych w języku C

Wersja 1.0 (29 stycznia 2016)

Spis treści:

Przed zapoznaniem się z instrukcją i przystąpieniem do wykonywania ćwiczenia należy opanować następujący materiał teoretyczny:

1. Podstawy języka C [1].
2. Najważniejsze specyficzne właściwości języka C zaimplementowanego w kompilatorze HI-TECH C® dla mikrokontrolerów PIC10/12/16 [2,3].
3. Podstawy architektury i programowania mikrokontrolerów Microchip PIC16 z rodziny Midrange [4-10].
4. Ogólna orientacja w składnikach zestawu ZL4PIC do uruchamiania mikrokontrolerów [11,12].

1. Cel ćwiczenia

Celem ćwiczenia jest nabycie praktycznych umiejętności programowania mikrokontrolerów z rodziny PIC16 w języku C. W ćwiczeniu położono nacisk na zastosowania mikrokontrolerów w aplikacjach kontrolno-pomiarowych, obejmujących pomiar napięcia przy użyciu przetwornika analogowo-cyfrowego A/D, precyzyjny pomiar czasu przy użyciu sprzętowych zegarów, prezentację wyników pomiarów na wyświetlaczu numerycznym LED lub alfanumerycznym LCD oraz sygnalizację akustyczną. W ćwiczeniu wykorzystano wyłącznie darmowe powszechnie dostępne oprogramowanie.

2. Zagrożenia

Rodzaj	Brak	Małe	Średnie	Duże
zagrożenie elektryczne		+		
zagrożenie optyczne	+			
zagrożenie mechaniczne (w tym akustyczne, hałas)	+			
zagrożenie polem elektro-magnetycznym (poza widmem optycznym)	+			
zagrożenie biologiczne	+			
zagrożenie radioaktywne (jonizujące)	+			
zagrożenie chemiczne	+			
zagrożenie termiczne (w tym wybuch i pożar)	+			

Urządzenia wykorzystywane w tym ćwiczeniu zasilane są bezpiecznym napięciem 9V otrzymywanym z zasilaczy podłączonych do sieci 230V.

3. Wprowadzenie teoretyczne

3.1. Przegląd 8-bitowych mikrokontrolerów PIC

Mikrokontrolery PIC są zaprojektowane według zmodyfikowanej architektury Harvard RISC. Architektura Harvard charakteryzuje się rozdzieleniem pamięci programu i pamięci danych, przy czym długość słowa maszynowego jest dobierana niezależnie w obu rodzajach pamięci. W mikrokontrolerach należących do rodzin PIC10, PIC12, PIC16 i PIC18 jednostka arytmetyczno-logiczna i słowo w pamięci danych są zawsze 8-bitowe. Słowo w pamięci programu może mieć rozmiary:

- 12-bitów (Baseline Architecture) – w układach scalonych PIC10 i PIC12 (oraz wybranych PIC16),
- 14-bitów (Midrange and Enhanced Midrange Architecture) – w układach PIC16 (oraz wybranych PIC12),
- 16-bitów (PIC18 Architecture, dawniej High Performance Architecture) – w układach PIC18.

Najliczniej reprezentowane są układy PIC16, które będą podstawą zajęć w Laboratorium Elektroniki. W pracowni dostępne są mikrokontrolery PIC16F84A, PIC16F819 i PIC16F877A. Mikrokontrolery PIC16F84A reprezentują jedną z najprostszych i najstarszych konstrukcji, natomiast mikrokontrolery PIC16F877A wyróżniają się w grupie Midrange szczególnie bogatym wyposażeniem. Mikrokontrolery PIC16F819 reprezentują pośredni zakres oferowanych możliwości.

Tabela 1. Porównanie mikrokontrolerów dostępnych w Laboratorium Elektroniki.

Wyposażenie	Mikrokontroler		
	PIC16F84A	PIC16F819	PIC16F877A
Pamięć programu (słowa 14-bitowe)	1024	2048	8192
Pamięć danych SRAM (bajty)	68	256	368
Pamięć danych EEPROM (bajty)	64	256	256
Wejścia/wyjścia cyfrowe	13	do 16	do 33
Oscylator taktujący zewn./wewn.	tak / –	tak / tak	tak / –
Przetwornik 10 bitowy A/D	–	1	1
Zegary 8/16 bitowe	1 / 0	2 / 1	2 / 1
Komparatory analogowe	–	2	2
Układy CCP i PWM	–	1	1
Interfejsy transmisji szeregowej	–	SPI, I^2C (Slave)	USART (w tym RS232), SPI, I^2C (Master/Slave)
8-bitowy port równoległy	–	–	PSP

3.2. Przegląd właściwości kompilatora języka C dla mikrokontrolerów Midrange

3.2.1. Dostępne oprogramowanie

Do programowania wybranych mikrokontrolerów PIC z rodziny Midrange zostanie wykorzystane zintegrowane środowisko MPLAB IDE udostępnione bezpłatnie przez firmę Microchip na stronie internetowej www.microchip.com. Środowisko to zawiera następujące składniki:

- MPLAB Editor - edytor tekstu umożliwiający wygodną edycję plików źródłowych i nagłówkowych,
- MPASM – makroasembler obsługujący wszystkie rodziny mikrokontrolerów firmy Microchip,
- MPLINK – program łączący moduły tworzone za pomocą asemblera i kompilatorów języka C w jeden plik z danymi dla programatora, emulatora lub symulatora,
- MPLAB Project Manager – menadżer projektów ułatwiający pracę z projektami zawierającymi wiele plików,
- MPLAB SIM – programowy symulator mikrokontrolerów,
- obsługę zewnętrznych urządzeń programatorów i emulatorów sprzętowych, m.in. programatora PICSTART Plus dostępnego w Laboratorium Elektroniki,
- system pomocy i pliki z dokumentacją.

Dla mikrokontrolerów PIC z rodzin Baseline i Midrange producent nie dostarcza własnego kompilatora języka C, jednakże instalator pakietu MPLAB IDE umożliwia zainstalowanie kompilatorów innych firm. W wersji MPLAB IDE 8.66 (wersja używana w niniejszej instrukcji) dostępne są m.in. kompilatory języka C:

- HI-TECH C® for PIC10/12/16 MCUs,
- CCS C Compiler for PIC12/14/16/18.

UWAGA: pliki instalacyjne wymienionych kompilatorów mogą być nieobecne w niektórych późniejszych wersjach instalatora pakietu MPLAB IDE.

Ze względu na znaczące różnice między tymi kompilatorami ograniczymy się tylko do kompilatora HI-TECH C® for PIC10/12/16 MCUs, który jest w znacznym stopniu zgodny ze standardem ANSI C. W niniejszej instrukcji założono znajomość podstaw języka programowania C i omówione zostaną tylko specyficzne cechy kompilatora i jego bibliotek standardowych. Kompilator HI-TECH C® for PIC10/12/16 MCUs posiada swoje własne warunki licencji odrębne od pakietu MPLAB IDE. Kompilator ten uaktywniony w wersji darmowej (Lite Mode) nie narzuca żadnych ograniczeń czasu użytkowania ani rozmiarów programu, natomiast nie udostępnia niektórych metod optymalizacji kodu programu. Najwyższa wydajność jest oferowana tylko w wersji PRO, którą można uaktywnić za darmo na 45 dniowy okres próbny albo bez ograniczeń w wersji komercyjnej. Wydajność oferowana w trybie „Lite Mode" jest jednak w zupełności wystarczająca do realizacji ćwiczenia.

Źródłowe pliki dla kompilatora mają rozszerzenie `.c`. Proces kompilacji przebiega w kilku etapach, z którymi związane jest tworzenie w katalogu projektu szeregu plików zawierających pośrednie rezultaty (pliki z rozszerzeniami `.pre`, `.p1`, `.as`, `.obj`). Znajomość tych plików nie jest konieczna w niniejszym ćwiczeniu. Końcowy rezultat jest zapisywany do pliku z rozszerzeniem `.hex`, który zawiera kompletny zapis kodu programu, danych konfiguracyjnych oraz danych przeznaczonych do zapisania w pamięci nieulotnej mikrokontrolera. Pliki `.hex` można otworzyć w środowisku MPLAB IDE i załadować do pamięci mikrokontrolera przy wykorzystaniu programatora bez konieczności ponownego używania kompilatora. Ponadto tworzone są także pliki z rozszerzeniem `.cof`, które przechowują dane dla debugera/symulatora.

3.2.2. Podstawowe typy danych

Kompilator HI-TECH C® zapewnie szeroki wybór podstawowych typów danych o różnych rozmiarach w pamięci (*Tabela 2*). Jednostka arytmetyczno-logiczna w mikrokontrolerach z rodziny Midrange jest 8-bitowa, zatem wybór typu danych o możliwie najwęższym zakresie zapewnia najwyższą wydajność i najmniejszy rozmiar kodu. Operacje na liczbach zmiennoprzecinkowych są szczególnie złożone i czasochłonne, dlatego należy ich używać tylko w koniecznych przypadkach. W wielu programach możliwe jest poprawienie

wydajności poprzez wykorzystanie liczb całkowitych w charakterze liczb stałopozycyjnych zastępujących liczby zmiennoprzecinkowe.

Tabela 2. Podstawowe typy danych w kompilatorze HI-TECH C® for PIC10/12/16 MCUs.

Typ	Rozmiar	Zakres / precyzja
bit	1	0 albo +1
signed char	8	–128 … +127
unsigned char	8	0 … +255
signed short	16	–32 768 … +32 767
unsigned short	16	0 … +65 535
signed int	16	–32 768 … +32 767
unsigned int	16	0 … +65 535
signed short long	24	–8 388 608 … +8 388 607
unsigned short long	24	0 … +16 777 215
signed long	32	–2 147 483 648 … +2 147 483 647
unsigned long	32	0 … +4 294 967 295
float	24 albo 32*	około 5 albo 8 cyfr dziesiętnych
double	24 albo 32*	około 5 albo 8 cyfr dziesiętnych

* Format liczb zmiennoprzecinkowych 32-bitowych jest w pełni zgodny z typem danych o pojedynczej precyzji zdefiniowanym w normie IEEE 754. Formaty obu typów zmiennoprzecinkowych zostały podane w materiałach do wykładu „Temat 2. Kodowanie liczb”. Wybór rozmiarów typów float i double jest dokonywany przez programistę np. w oknie dialogowym "Build Options For Project", zakładka "Global". W pakiecie MPLAB IDE okno to otwiera się po wybraniu z menu pozycji Project/Build options .../Project (dostępne gdy otwarty jest jakiś projekt).

W mikrokontrolerach z rodziny Midrange pamięć danych (GPR – *general purpose registers*) jest podzielona na banki o rozmiarze 128B, przy czym cześć każdego banku jest zajęta przez rejestry specjalne (SFR – *special function registers*). Rozmiary typów złożonych (tablice i struktury) nie mogą przekraczać rozmiaru pamięci danych w ramach jednego banku pamięci. Kompilator języka C dokonuje automatycznej alokacji zmiennych w pamięci i generuje niezbędny kod przełączający banki pamięci. Dane w strukturach i tablicach są umieszczane pod kolejnymi wolnymi adresami z wyrównaniem do granicy bajtów (z wyjątkiem pól bitowych).

W strukturach można zdefiniować pola bitowe, np.:

```
struct {
  unsigned int lo : 1;
  unsigned int dummy : 6;
  unsigned int hi : 1;
} foo;
```

Chociaż użyto typu `unsigned int`, który typowo ma rozmiar 16 bitów, to jednak w przypadku pól bitowych ilość bitów zajętych w pamięci odpowiada liczbie bitów zadeklarowanych. Kolejne pola bitowe są umieszczane pod kolejnymi wolnymi bitami, rozpoczynając od bitu nr 0 (bitu najmniej znaczącego) w pierwszym bajcie struktury. Zmienne zdefiniowanego powyżej typu `foo` zajmują w pamięci 1 bajt.

3.2.3. Dostęp do rejestrów specjalnych mikrokontrolera

Wszystkie rejestry specjalne (SFR) leżące w przestrzeni adresowej danych (mapy pamięci podano w Aneksie A) zdefiniowano w plikach nagłówkowych dostarczanych razem z kompilatorem jako zmienne z modyfikatorem „volatile” (tzn. ulotny) i adresem statycznym podanym po znaku „@”. Dodatkowo dla rejestrów SFR zawierających bity o specyficznym

znaczeniu, poszczególne bity zdefiniowano także jako jednobitowe zmienne. Przykładowo w pliku nagłówkowym pic16f84a.h (dla układu PIC16F84A) zdefiniowano m.in.

```
volatile  unsigned char TRISA  @ 0x85;
volatile  bit           TRISA2 @ (unsigned)&TRISA*8+2;
```

Ustawienie np. drugiego bitu w rejestrze TRISA bez modyfikacji pozostałych bitów jest możliwe zarówno poprzez operację na całym rejestrze 8-bitowym

```
TRISA |= 0x04;
```

jak i poprzez odwołanie do jednego wybranego bitu

```
TRISA2 = 1;
```

3.2.4. Dostęp do nieulotnej pamięci danych EEPROM poprzez zmienne w języku C

Liczne mikrokontrolery z rodziny Midrange wyposażone są w nieulotną pamięć danych typu EEPROM. Pamięć ta nie jest dostępna bezpośrednio w przestrzeni adresowej danych. Początkowy stan komórek tej pamięci można umieścić w pliku .hex wykorzystując makrodefinicję `__EEPROM_DATA()`

```
#include <htc.h>
_ _EEPROM_DATA(0, 1, 2, 3, 4, 5, 6, 7);
```

Kolejne argumenty wywołania makrodefinicji definiują kolejne bajty w pamięci EEPROM. Wielokrotne wywołania makrodefinicji dotyczą następujących po sobie komórek w pamięci. Dane określone w ten sposób zostaną zapisane do pamięci EEPROM tylko podczas programowania mikrokontrolera. Do zapisu oraz odczytu jednego bajtu pamięci EEPROM podczas wykonywania programu przeznaczone są procedury `eeprom_write` oraz `eeprom_read`, dostępne w bibliotece standardowej języka C, np.:

```
#include <htc.h>
void eetest(void) {
   unsigned char value = 1;
   unsigned char address = 0;

   // zapisz "value" do EEPROM pod zadany adres
   eeprom_write(address, value);
   // odczytaj z EEPROM pod zadanym adresem
   value = eeprom_read(address);
}
```

Funkcja `eeprom_write` rozpoczyna proces zapisu i nie oczekuje na zakończenie, które nastąpi po kilku milisekundach. Kolejne wywołania funkcji `eeprom_write` lub `eeprom_read` nie prowadzą jednak do błędu, gdyż funkcje te zawsze oczekują na zakończenie zapisu przed wykonaniem kolejnej żądanej operacji. Rozmiar pamięci EEPROM dla wybranego aktualnie mikrokontrolera jest określony przez predefiniowaną stałą `_EEPROMSIZE`. Należy pamiętać, że czas dostępu do danych w pamięci EEPROM jest o kilka rzędów wielkości dłuższy niż w przypadku zmiennych w ulotnej pamięci danych typu SRAM.

3.2.5. Pamięć konfiguracyjna mikrokontrolera

Rejestr konfiguracyjny mikrokontrolera znajduje się pod adresem 2007h w przestrzeni adresowej programu i można go ustawić tylko na etapie programowania mikrokontrolera.

Rejestr ten kontroluje między innymi typ zegara taktującego pracę mikrokontrolera, opóźnienie startu po załączeniu zasilania, aktywność układu *Watchdog Timer*, sposób programowania pamięci Flash RAM (niskonapięciowe albo wysokonapięciowe), ochronę zawartości pamięci programu i pamięci danych EEPROM, oraz funkcje niektórych wyprowadzeń mikrokontrolera. Błędne ustawienie wartości tego rejestru jest częstą przyczyną niepowodzeń podczas prób uruchomienia programu. Rejestr konfiguracyjny omówiono wcześniej szczegółowo w instrukcji do ćwiczenia ASM w rozdziale „Pamięć konfiguracji w mikrokontrolerach PIC z rodziny Midrange".

Programista przygotowujący program w języku C w postaci projektu zarządzanego przez pakiet MPLAB IDE może ustawić rejestr konfiguracyjny poprzez okno dialogowe Configuration Bits dostępne po wybraniu z menu Configure pozycji Configuration Bits... .

Alternatywnie wartość rejestru konfiguracyjnego można określić w tekście źródłowym programu wykorzystując makrodefinicję `__CONFIG(x)` zdefiniowaną w pliku nagłówkowym htc.h. Wartość argumentu `x` można podać bezpośrednio lub skonstruować ją ze stałych symbolicznych zdefiniowanych w pliku nagłówkowym i połączonych operatorem bitowego iloczynu logicznego &, np. dla mikrokontrolera PCI16F819:

```
#include<htc.h>
_ _CONFIG(CP_OFF & DEBUG_OFF & LVP_OFF & BOREN_ON & MCLRE_ON & PWRTE_ON & WDTE_OFF & FOSC_XT);
```

Użyte powyżej stałe symboliczne są zdefiniowane w plikach nagłówkowych dedykowanych do poszczególnych modeli mikrokontrolerów. Przykładowo z układem PIC16F819 związany jest plik nagłówkowy pic16f819.h. Wstawienie pliku `htc.h` powoduje wstawienie także pliku właściwego dla mikrokontrolera wybranego w projekcie.

3.2.6. Ograniczenia wynikające z budowy stosu w mikrokontrolerach z rodziny Midrange

Stos sprzętowy w mikrokontrolerach z rodziny Midrange może być użyty tylko do przechowywania adresów powrotów z funkcji i ma pojemność 8 adresów. Wykorzystanie przerwań dodatkowo zmniejsza głębokość stosu dla jawnych wywołań funkcji. Architektura mikrokontrolerów Midrange nie zawiera żadnych rozwiązań umożliwiających wykrywanie przepełnienia stosu sprzętowego.

Argumenty i zmienne lokalne funkcji są umieszczane na stosie obsługiwanym w sposób programowy a rozmiar stosu jest ograniczony tylko ilością wolnych komórek pamięci we wszystkich bankach. Sposób zarządzania stosem danych zaimplementowany w kompilatorze HI-TECH C® for PIC10/12/16 MCUs powoduje, że wszystkie funkcje są tylko jednowejściowe (nie nadają się do wywołań rekurencyjnych).

Przerwania sprzętowe mogą wydarzyć się w dowolnym momencie. Aby zapewnić poprawne działanie funkcji wywoływanych zarówno pomiędzy przerwaniami jak i podczas obsługi przerwań, kompilator generuje dwa warianty kodu takich funkcji.

3.2.7. Biblioteka standardowa języka C

W bibliotece standardowej języka C dostarczanej z kompilatorem HI-TECH C® for PIC10/12/16 MCUs zaimplementowano liczne funkcje objęte standardem ANSI C. Funkcje pominięte w tej bibliotece dotyczą głównie operacji wejścia/wyjścia, które są znane z pełnych systemów komputerowych, natomiast nie mają jednoznacznych odpowiedników w przypadku programowania mikrokontrolerów. Wśród niezaimplementowanych właściwości są m.in. wejście standardowe stdin oraz wyjście standardowe stdout. Funkcje z biblioteki standardowej, które odczytują wyniki z wejścia standardowego, np. funkcja gets, wywołują

funkcję getch albo getche pobierającą jeden znak z wejścia standardowego. Jeżeli wejście standardowe jest wykorzystywane, to obowiązek dostarczenia definicji funkcji getch i/lub getche w źródłach kompilowanego programu spoczywa na programiście. Funkcje te muszą mieć nagłówek:

```
char getch(void);
char getche(void);
```

Funkcja getch odczytuje 1 bajt z wejścia standardowego, natomiast getche dodatkowo zapisuje echo wprowadzanych znaków na wyjście standardowe.

Funkcje z biblioteki standardowej, które zapisują wyniki do wyjścia standardowego, np. funkcja printf, wywołują funkcję putch zapisującą jeden znak na wyjście standardowe. Funkcję putch definiuje programista a jej nagłówek musi mieć postać:

```
void putch(char c);
```

Funkcja printf nie jest umieszczona w bibliotece standardowych funkcji języka C. Funkcja ta jest generowana ze specjalnego pliku źródłowego, przy czym zestaw opcji włączanych warunkowo do kompilacji zależy od analizy łańcucha znaków opisującego formatowanie, który podawany jest jako pierwszy argument wywołania funkcji. Im więcej różnych wymaganych właściwości zostanie wykrytych, tym większy będzie rozmiar wygenerowanego kodu funkcji printf. Proces wyboru składników funkcji printf włączonych do kompilacji nie podlega bezpośredniej kontroli użytkownika. Jeżeli łańcuch znaków opisujących formatowanie nie jest dany bezpośrednio podczas kompilacji (jest przekazywany przez wskaźnik), wówczas kompilator może co najwyżej przeanalizować dalsze argumenty funkcji printf i można spodziewać się znacznej nadmiarowości wygenerowanego kodu. W szczególnie skomplikowanych przypadkach kod samej funkcji printf może okazać się większy od rozmiaru całej pamięci programu dostępnej w mikrokontrolerach PIC16F84A oraz PIC16F819 (odpowiednio 1K oraz 2K słów 14-bitowych).

Funkcje ze standardowych bibliotek języka C oraz procedury arytmetyczne są automatycznie przyłączane przez linker do kodu programu na podstawie analizy ich użycia w programie. Wykorzystanie tylko wybranych funkcji z danej biblioteki nie powoduje przyłączenia żadnych innych funkcji z tej biblioteki – zostaną uwzględnione tylko składniki jawnie występujące w tekście źródłowym programu. Opis funkcji bibliotecznych jest dostępny w podręczniku do kompilatora. Gdy w środowisku MPLAB IDE otwarty jest projekt wykorzystujący kompilator HI-TECH C®, wówczas plik PDF z podręcznikiem użytkownika można otworzyć przez wybranie z menu Project/HI-TECH C Manual.

3.2.8. Szablon minimalnego programu dla kompilatora HI TECH C®

W każdym programie w języku C konieczna jest funkcja `main`. Funkcja ta nie przyjmuje żadnych argumentów a rezultat zwracany przez funkcję nie będzie miał znaczenia i można zadeklarować `void`. Ponadto zazwyczaj wymagane jest także użycie pliku nagłówkowego `htc.h` który nie należy do standardu ANSI C lecz wynika ze specyficznych cech kompilatora i obsługiwanych mikrokontrolerów. Kompletny program, który nie wykonuje żadnej akcji, wygląda zatem następująco:

```
#include <htc.h>

void main(void)
{
}
```

4. Wybrane zagadnienia z architektury mikrokontrolerów z rodziny Midrange

4.1. Wybór częstotliwości wewnętrznego oscylatora RC w mikrokontrolerach PIC16

Zestaw uruchomieniowy ZL4PIC posiada rezonator kwarcowy 4 MHz przeznaczony do wykorzystania w generatorze taktującym pracę mikrokontrolera. W przypadku użycia mikrokontrolera PIC16F819 dostępny jest także wewnętrzny oscylator RC o częstotliwości 8 MHz, jednakże po włączeniu zasilania częstotliwość ta jest dzielona przez 256. W wielu zastosowaniach ta najniższa możliwa częstotliwość jest niewystarczająca i należy zmienić stopień podziału przez zapisanie odpowiedniej wartości do rejestru OSCCON. Zmiana stopnia podziału może zostać dokonana w dowolnym momencie wykonywania programu. W przypadku użycia oscylatora wykorzystującego zewnętrzny rezonator kwarcowy nie ma możliwości podziału jego częstotliwości. Wybór typu oscylatora może być dokonany tylko na etapie programowania mikrokontrolera poprzez ustawienie wartości rejestru konfiguracyjnego (rozdział „3.5. Pamięć konfiguracyjna mikrokontrolera”).

Pozostałe mikrokontrolery dostępne w Laboratorium Elektroniki (tzn. PIC16F84A oraz PIC16F877A) nie posiadają wewnętrznego oscylatora RC ani rejestru OSCCON.

Rejestr OSCCON, adres 8Fh (*Oscillator Control Register*)

U-0	R/W-0	R/W-0	R/W-0	U-0	R-0	U-0	U-0
—	IRCF2	IRCF1	IRCF0	—	IOFS	—	—
bit 7							bit 0

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n” = wartość po włączeniu zasilania (POR):	„1” = bit ustawiony,	„0” = bit wyzerowany.

Opisy bitów:

bit 7 **Niezaimplementowane**: przy odczycie wartość „0”.

bit 6-4 **IRCF<2:0>**: wybór częstotliwości wewnętrznego oscylatora RC
111 = 8 MHz (bezpośredni sygnał z zegara 8 MHz)
110 = 4 MHz (podzielona częstotliwość z zegara 8 MHz)
101 = 2 MHz (———————— , , ————————)
100 = 1 MHz (———————— , , ————————)
011 = 500 kHz (———————— , , ————————)
010 = 250 kHz (———————— , , ————————)
001 = 125 kHz (———————— , , ————————)
000 = 31,25 kHz (———————— , , ————————)

bit 3 **Niezaimplementowane**: przy odczycie wartość „0”.

bit 2 **IOFS**: bit stabilności częstotliwości wewnętrznego oscylatora
1 = częstotliwość jest stabilna,
0 = częstotliwość nie jest stabilna.

bit 1-0 **Niezaimplementowane**: przy odczycie wartość „0”.

Plik nagłówkowy pic16f819.h dostarczany z kompilatorem HI-TECH C® definiuje zarówno jednobajtową zmienną OSCCON, jak i jednobitowe zmienne IOFS, IRCF0, IRCF1, IRCF2 umożliwiające dostęp do poszczególnych bitów rejestru.

4.2. Konfigurowanie wejść analogowych i wykorzystanie przetwornika A/D

Niektóre mikrokontrolery z rodziny Midrange wyposażone są w przetworniki analogowo-cyfrowe (A/D) o rozdzielczości 8, 10 albo 12 bitów. Wśród mikrokontrolerów dostępnych w Laboratorium Elektroniki modele PIC16F819 oraz PIC16F877A wyposażone są w jeden 10-bitowy przetwornik współpracujący z analogowym selektorem sygnału z jednego z wejść ANx układu scalonego. W zestawie uruchomieniowym ZL4PIC znaczenie ma jednak tylko wejście analogowe AN0, gdyż jedynie to wejście można połączyć z suwakiem potencjometru R2 pracującego jako dzielnik napięcia zasilającego. Zakres napięć rozróżnianych przez przetwornik jest określony przez dolne oraz górne napięcia odniesienia, które można podać z zewnętrznych źródeł poprzez wejścia odpowiednio AN2 i AN3 albo przyłączyć wewnątrz mikrokontrolera do masy V_{SS} oraz dodatniego napięcia zasilania V_{DD}, które w zestawie uruchomieniowym ZL4PIC jest stabilizowane na poziomie +5V. Zarządzanie analogowym selektorem i przetwornikiem A/D odbywa się poprzez rejestry specjalne ADCON0 oraz ADCON1 (opisane na następnej stronie), natomiast wynik konwersji odczytuje się z rejestrów ADRESH oraz ADRESL. Wszystkie wymienione rejestry są dostępne bezpośrednio w przestrzeni adresowej danych.

Napięcie, które będzie podlegało konwersji A/D jest zapamiętywane w kondensatorze wewnątrz mikrokontrolera. Przed rozpoczęciem konwersji wymagane jest odczekanie czasu akwizycji potrzebnego na naładowanie kondensatora. Czas ten zależy od rezystancji R_S źródła mierzonego napięcia. Dla zestawu ZL4PIC możemy założyć (z zapasem) $R_S = 10\ k\Omega$ i wówczas wymagany czas akwizycji wynosi około 20 µs. Kondensator jest odłączany na czas konwersji i przyłączany z powrotem z opóźnieniem $2\ T_{AD}$ po zakończeniu konwersji (wyzerowaniu bitu GO/$\overline{\text{DONE}}$), gdzie T_{AD} jest okresem zegara taktującego pracę przetwornika A/D.

W celu wykonania konwersji w przetworniku A/D należy wykonać następujące kroki:

1). Skonfigurować moduł przetwornika A/D:
   - skonfigurować odpowiednie nóżki układu scalonego jako wejścia analogowe (rejestr ADCON1),
   - wybrać formatu 10-bitowego wyniku konwersji (rejestr ADCON1),
   - wybrać wejście sygnału dla przetwornika A/D (rejestr ADCON0),
   - wybrać zegar taktujący pracę przetwornika A/D (rejestr ADCON0),
   - włączyć modułu przetwornika A/D (rejestr ADCON0).
2). Odczekać wymagany czas akwizycji.
3). Uruchomić konwersję przez ustawienie bitu GO/$\overline{\text{DONE}}$ (rejestr ADCON0).
4). Odczekać na zakończenie konwersji do momentu, gdy bit GO/$\overline{\text{DONE}}$ zmieni wartość na 0. Układ przetwornika A/D może generować przerwanie w momencie zakończenia cyklu przetwarzania, jednakże w bieżącym ćwiczeniu ograniczymy się do omówienia konwersji tylko przy wyłączonych przerwaniach przetwornika A/D (bit ADIE w rejestrze PIE1 pozostawiony w domyślnym stanie początkowym 0).
5). Odczytać wynik konwersji z pary rejestrów ADRESH:ADRESL (pod adresami 1Eh oraz 9Eh). Format wyniku zależy od stanu bitu ADFM w rejestrze ADCON1.

ADFM = 1

7							2	1	0	7							0
0	0	0	0	0	0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0		
rej. ADRESH								rej. ADRESL									

ADFM = 0

7							2	1	0	7							0
b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	0	0		
rej. ADRESH								rej. ADRESL									

6). Przed rozpoczęciem nowego cyklu konwersji A/D odczekać co najmniej okres czasu 2 T_{AD} (opóźnienie przyłączenia kondensatora po konwersji). Wykonać skok do pkt. 1 albo 2 w zależności od potrzeb.

Aby uzyskać najdokładniejszy wynik konwersji A/D, okres T_{AD} sygnału zegarowego taktującego moduł przetwornika musi być tak krótki jako to możliwe, jednakże nie krótszy niż 1,6 μs oraz nie dłuższy niż 6,4 μs. Dla rezonatora kwarcowego 4 MHz, który znajduje się w zestawie ZL4PIC, optymalny okres wynosi więc $T_{AD} = 8\ T_{OSC} = 2{,}0$ μs, gdzie okres zegara $T_{OSC} = 1/4\text{MHz} = 0{,}25$ μs. Okres taki otrzymamy zapisując bit ADCS2 := 0 (w rejestrze ADCON1) oraz bity ADCS<1:0> := 01 (w rejestrze ADCON0).

Uwaga 1: dla mikrokontrolerów pracujących z zegarem szybszym niż 1 MHz nie jest zalecane używanie wewnętrznego oscylatora RC dedykowanego tylko do użytku z przetwornikiem A/D. Znacznie mniejszy poziom zakłóceń osiąga się wykorzystując ogólny zegar systemowy. Alternatywnie możliwe jest użycie wewnętrznego oscylatora RC w połączeniu z wprowadzaniem jądra mikrokontrolera w stan uśpienia na czas wykonywania konwersji A/D, co jednak wykracza poza program niniejszego ćwiczenia.

Uwaga 2: w mikrokontrolerach PIC16F819 dostępny jest także wewnętrzny oscylator RC 8 MHz współpracujący z programowanym dzielnikiem (opisany w rozdziale 4.1), który można wykorzystać do taktowania całego mikrokontrolera. Ten oscylator RC nie jest tożsamy z oscylatorem RC współpracującym tylko z przetwornikiem A/D.

Rejestry konfiguracyjne przetwornika A/D i analogowego selektora mają następujący format:

Rejestr ADCON0, adres 1Fh (*Analog to Digital Control Register 0*)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n” = wartość po włączeniu zasilania (POR):	„1” = bit ustawiony,	„0” = bit wyzerowany.

Opisy bitów:

bit 7-6 **ADCS<1:0>**: wybór zegara dla przetwornika A/D

Jeżeli ADCS2 = 0 (bit z rejestru ADCON1):

00 = $F_{OSC}/2$,
01 = $F_{OSC}/8$,
10 = $F_{OSC}/32$,
11 = F_{RC} (wewnętrzny oscylator RC tylko dla przetwornika A/D).

Jeżeli ADCS2 = 1:

00 = $F_{OSC}/4$,
01 = $F_{OSC}/16$,
10 = $F_{OSC}/64$,
11 = F_{RC} (wewnętrzny oscylator RC tylko dla przetwornika A/D).

bit 5-3 **CHS<2:0>**: wybór wejścia analogowego dla przetwornika A/D
000 = kanał 0 (AN0),
001 = kanał 1 (AN1),
010 = kanał 2 (AN2),
011 = kanał 3 (AN3),
100 = kanał 4 (AN4)
101 = kanał 5 (AN5 – niedostępne w PIC16F819),
110 = kanał 6 (AN6 – niedostępne w PIC16F819),
111 = kanał 7 (AN7 – niedostępne w PIC16F819).

bit 2 **GO/$\overline{\text{DONE}}$**: bit stanu konwersji A/D
Jeżeli ADON = 1:
1 = trwa konwersja A/D (ustawienie tego bitu rozpoczyna konwersję),
0 = konwersja A/D nie jest wykonywana (ten bit jest automatycznie zerowany w momencie zakończenia konwersji).

bit 1 **Niezaimplementowane**: przy odczycie wartość „0”.

bit 0 **ADON**: bit załączenia przetwornika A/D
1 = moduł przetwornika A/D jest włączony,
0 = moduł przetwornika A/D jest wyłączony i nie pobiera prądu zasilania.

Plik nagłówkowe dostarczane z kompilatorem HI-TECH C® definiują jednobitową zmienną GO_nDONE (oraz synonimy GO_DONE, nDONE) umożliwiającą dostęp do bitu GO/$\overline{\text{DONE}}$. Pozostałe bity rejestru ADCON0 są dostępne poprzez zmienne o nazwach jak w ramce formatu rejestru.

Rejestr ADCON1, adres 9Fh (*Analog to Digital Control Register 1*)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Opisy bitów:
bit 7 **ADFM**: wybór formatu wyniku z przetwornika A/D
1 = wyrównanie do prawej; 6 najbardziej znaczących bitów rejestru ADRESH jest odczytywanych jako 0,
0 = wyrównanie do lewej; 6 najmniej znaczących bitów rejestru ADRESL jest odczytywanych jako 0.

bit 6 **ADCS2**: opisany wcześniej razem z bitami ADCS<1:0> z rej. ADCON0

bit 5-4 **Niezaimplementowane**: przy odczycie wartość „0”.

bit 3-0 **PCFG<3:0>**: konfiguracja wejść przetwornika A/D i napięć odniesienia.

PCFG	AN7(*)	AN6(*)	AN5(*)	AN4	AN3	AN2	AN1	AN0	V_{REF+}	V_{REF-}
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0001	A	A	A	A	V_{REF+}	A	A	A	AN3	V_{SS}
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0011	D	D	D	A	V_{REF+}	A	A	A	AN3	V_{SS}
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0101	D	D	D	D	V_{REF+}	D	A	A	AN3	V_{SS}
011x	D	D	D	D	D	D	D	D	–	–
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1010	D	D	A	A	V_{REF+}	A	A	A	AN3	V_{SS}
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	AN3	AN2

Oznaczenia w tabeli: (*) = wejścia niedostępne w mikrokontrolerze PIC16F819,
A = wejście analogowe,
D = wejście/wyjście cyfrowe,
V_{REF-} = dolne napięcie odniesienia dla przetwornika A/D,
V_{REF+} = górne napięcie odniesienia dla przetwornika A/D,
V_{DD} = dodatnie napięcie zasilania,
V_{SS} = masa.

4.3. Wykorzystanie portu A jako cyfrowego portu wejścia/wyjścia

Niektóre linie portu A mogą pełnić różne funkcje związane z dodatkowymi urządzeniami peryferyjnymi. Jeżeli te dodatkowe funkcje są nieaktywne, wówczas linie portu A tworzą dwukierunkowy port cyfrowy zgodny z poziomami napięć układów TTL. Liczba dwustanowych linii dostępnych w porcie A zależy od modelu mikrokontrolera.

Kierunek transmisji danych określają bity w rejestrze TRISA. Ustawienie w tym rejestrze bitu w stan 1 konfiguruje związaną z nim linię portu jako wejście, tzn. sterownik wyjścia linii przyjmuje stan wysokiej impedancji. Po włączeniu zasilania oraz wszelkich rodzajach resetu wszystkie aktywne bity rejestru TRISA ustawiane są w stan 1. Wyzerowanie bitu w TRISA konfiguruje linię jako wyjście, którego stan logiczny jest określony przez odpowiedni bit rejestru PORTA. Niezależnie od wartości zapisanej do rejestru TRISA, bieżący stan linii można zawsze oczytać z rejestru PORTA.

Rejestry kontrolujące stan portu A w mikrokontrolerach PIC16F84A

Adres	Nazwa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Wartość po POR	po innym resecie
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Rejestry kontrolujące stan portu A w mikrokontrolerach PIC16F819

Adres	Nazwa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Wartość po POR	po innym resecie
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxx0 0000	uuu0 0000
85h	TRISA	TRISA7	TRISA6	TRISA5(1)	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111

Rejestry kontrolujące stan portu A w mikrokontrolerach PIC16F877A

Adres	Nazwa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Wartość po POR	po innym resecie
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111

Oznaczenia:
x = wartość nieznana, u = wartość bez zmiany,
- = bit dotyczący linii, która nie została zaimplementowana w porcie A, odczytywany jako 0,
(1) linia RA5 w PIC16F819 jest tylko wejściem, bit 5 w TRISA jest odczytywany zawsze jako 1.
POR – reset po włączeniu zasilania (*Power On Reset*) oraz po spadku napięcia zasilającego,
inny reset – reset przez wyzerowanie linii $\overline{\text{MCLR}}$ lub przepełnienie licznika WDT (*Watchdog Timer*).

Sterowniki wyjść portu A są typu „push-pull”, tzn. zawierają zarówno tranzystory podnoszące napięcie na wyjściu do poziomu napięcia zasilania jak i tranzystory zwierające wyjście z masą. Wyjścia te można zredukować metodami programowymi do wyjść typu „otwarty dren”. W tym celu należy wyzerować odpowiednie bity w rejestrze PORTA, a następnie przełączać stan wyjść (między wysoką impedancją a zwarciem do masy) przez zapisywanie 1 albo 0 do odpowiednich bitów rejestru TRISA. W mikrokontrolerach PIC16F84A i PIC16F877A (nie dotyczy PIC16F819) sterownik wyjścia linii RA4 jest wyjątkowo typu otwarty dren, tak więc ustawienie 4-go bitu w rejestrze PORTA w stan 1 przełącza wyjście w stan wysokiej impedancji niezależnie od stanu rejestru TRISA.

Uwaga 1: odczyt rejestru PORTA zwraca zawsze bieżący stan linii, nie zaś wartość zapamiętaną podczas ostatniego zapisu do rejestru. Wartość odczytana z PORTA zaraz po zapisie może nie zgadzać się z wartością zapisaną, nawet gdy na wyjściu nie ma zwarcia. Czas przechodzenia wyjścia w zadany stan może być dłuższy od jednego cyklu zegarowego MCU i zależy od pojemności obwodów przyłączonych do wyjścia na zewnątrz mikrokontrolera.

Uwaga 2: operacja zapisu do PORTA przebiega w cyklu odczyt-modyfikacja-zapis. Odczyt bieżącego stanu i zapis stanu po modyfikacji do rejestrów zatrzaskowych dotyczy wszystkich linii portu, nawet podczas wykonywania instrukcji operujących na pojedynczych bitach (tzn. instrukcji bcf i bsf). Jedyną pewną metodą modyfikacji wybranych pojedynczych bitów w PORTA jest wykonanie operacji na buforze w pamięci a następnie przepisanie całego bajtu z bufora do rejestru PORTA.

4.4. Wykorzystanie portu B jako cyfrowego portu wejścia/wyjścia

W mikrokontrolerach PIC16F84A, PIC16F819 oraz PIC16F877A port B złożony jest z ośmiu linii. Jeżeli dodatkowe funkcje nie są aktywne, wówczas wszystkie linie portu B można wykorzystać jako uniwersalne wejścia/wyjścia zgodne z poziomami napięć TTL. Praca tych linii jest kontrolowana przez rejestry PORTB oraz TRISB. Sposób użycia rejestrów jest identyczny jak w przypadku analogicznych rejestrów portu A (patrz poprzedni rozdział), z tym że w 8-bitowych rejestrach portu B aktywne są wszystkie bity.

Rejestry kontrolujące stan portu B w mikrokontrolerach PIC16F84A, PIC16F819, PIC16F877A

Adres	Nazwa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Wartość po POR	po innym resecie
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111

Oznaczenia:
x = wartość nieznana, u = wartość bez zmiany,
POR – reset po włączeniu zasilania (*Power On Reset*) oraz po spadku napięcia zasilającego,
inny reset – reset przez wyzerowanie linii $\overline{\text{MCLR}}$ lub przepełnienie licznika WDT (*Watchdog Timer*).

W mikrokontrolerach PIC16F819 niektóre linie portu B mogą być użyte razem ze specjalistycznymi układami cyfrowymi, takimi jak kontroler magistral szeregowych SPI i I^2C oraz generator PWM (wytwarza impulsy o programowanym wypełnieniu) jednakże układy te nie są aktywne po włączeniu zasilania i wykraczają poza zakres niniejszego ćwiczenia.

Unikalną cechą portu B jest możliwość programowego włączenia rezystorów podciągających napięcie do poziomu napięcia zasilania na liniach skonfigurowanych jako wejścia (1 w odpowiednich bitach rejestru TRISB). Po włączeniu zasilania rezystory są nieaktywne i można je załączyć przez wyzerowanie bitu $\overline{\text{RBPU}}$ w rejestrze OPTION_REG (w kompilatorze HI-TECH C® bit ten jest dostępny poprzez zmienną nRBPU). Bit ten kontroluje załączenie rezystorów na wszystkich wejściach portu B, natomiast na wyjściach (0 w odpowiednich bitach rejestru TRISB) rezystory pozostają odłączone. Rezystory te są konieczne do poprawnego odczytu stanu przycisków podłączonych w zestawie ZL4PIC do wejść mikrokontrolera RB0, RB1, RB2 i RB3. W przypadku gdy rezystory nie są załączone, stan logiczny wejść przy rozwartych przyciskach jest niestabilny.

Rejestr OPTION_REG, adres 81h

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Dotyczy układów PIC16F84A, PIC16F819 oraz PIC16F877A

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n” = wartość po włączeniu zasilania (POR):	„1” = bit ustawiony,	„0” = bit wyzerowany.

Opisy bitów:

bit 7 **$\overline{\text{RBPU}}$**: bit sterujący rezystorami podciągających napięcie na wyjściach portu B do +5V
1 = podciąganie napięcia w porcie B wyłączone,
0 = podciąganie napięcia w porcie B włączone

W bieżącym rozdziale ograniczymy się tylko do podania informacji istotnych podczas operacji na porcie B. Pełny opis bitów rejestru OPTION_REG przedstawiono dalej w rozdziale „4.6.2. Zegar 0”.

Uwaga: w pakiecie MPLAB IDE domyślnie ustawione jest niskonapięciowe programowanie MCU (ang. *Low Voltage Program Enabled*), które wymaga użycia dodatkowo linii RB3 i po zakończeniu programowania linia ta nie będzie aktywna jako uniwersalne wejście/wyjście. W zestawie uruchomieniowym ZL4PIC aktywność linii RB3 jest wymagana do odczytu stanu jednego z przycisków. Zalecane jest wybranie programowania podwyższonym napięciem +12V (ang. *HV - High Voltage*), np. w oknie Configuration Bits, Field LVP. Pozostałe linie używane przez programator (RB6 i RB7 z portu B oraz RA5/$\overline{\text{MCLR}}$ z portu A) po zakończeniu programowania są aktywne. Uwaga ta nie dotyczy mikrokontrolera PIC16F84A, który nie obsługuje programowania niskonapięciowego.

4.5. Przerwania sprzętowe

W mikrokontrolerach o architekturze Midrange zaimplementowano jednopoziomowy system przerwań, tzn. wszystkie typy przerwań obsługiwane są przez tylko jedną procedurę. W niniejszym ćwiczeniu przerwania sprzętowe mogą być przydatne przede wszystkim do okresowego powtarzania czynności z okresem odmierzanym precyzyjnie przez zegary sprzętowe (omówione dalej w rozdziale „4.6. Pomiar czasu wewnątrz mikrokontrolera”). Ponadto przycisk SP1 w zestawie ZL4PIC jest połączony z linią mikrokontrolera RB0 i zmiana stanu tej linii skonfigurowanej jako wejście cyfrowe może generować przerwania.

Z każdą przyczyną przerwania związane są następujące dwa bity w rejestrach specjalnych mikrokontrolera:

- xxxIE (ang. *interrupt enable*) – bit zezwolenia na generowanie przerwania z danej przyczyny; przerwanie jest aktywne gdy bit ten jest równy 1,
- xxxIF (ang. *interrupt flag*) – znacznik ustawiany sprzętowo w stan 1 po każdym zgłoszeniu przerwania; zerowanie znacznika musi odbywać się w programowo.

Powyżej „xxx" symbolizuje różne oznaczenia poszczególnych przyczyn przerwań. Ponadto bit GIE (ang. *global interrupt enable*) w stanie 0 blokuje wszelkie przerwania, natomiast w stanie 1 dopuszcza przerwania nie zamaskowane przez pozostałe bity …IE.

Wystąpienie przyczyny przerwania powoduje następujące zdarzenia:

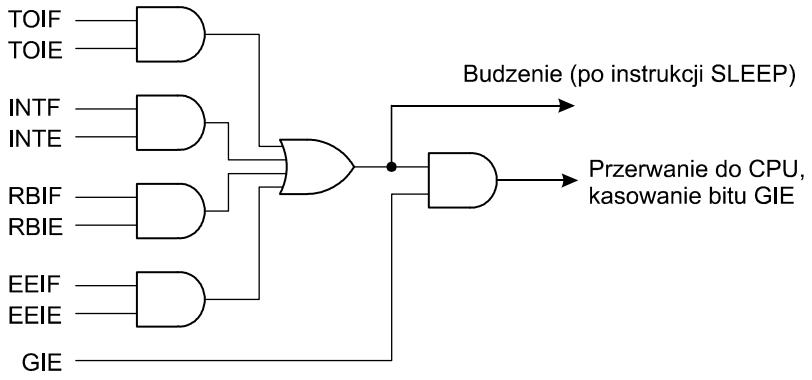
1. Ustawienie w stan 1 odpowiedniego znacznika …IF.
2. Jeżeli GIE = 1 oraz odpowiedni bit xxxIE = 1, to procesor odkłada na stos aktualną wartość licznika rozkazów PC, blokuje układ przerwań przez wyzerowanie GIE i wykonuje skok do procedury pod adresem 0004h.
3. Podczas powrotu z procedury obsługi przerwań przywracana jest poprzednia wartość PC, zaś przerwania są odblokowywane przez ustawienie bitu GIE.

W kompilatorze HI-TECH C® for PIC10/12/16 MCUs procedurę obsługi przerwania deklaruje się następująco:

```
void interrupt IntHandler(void) {
  if (TMR2IE && TMR2IF) {
    TMR2IF = 0;
    …
  }
  if (INTE && INTF) {
    INTF = 0;
    …
  }
  …
}
```

Wewnątrz powyższej procedury przedstawiono szablon typowego sposobu rozpoznawania przyczyny przerwania przez badanie stanu logicznego wyrażeń typu (xxxIE & xxxIF). Znaczniki xxxIF są ustawiane sprzętowo, natomiast ich zerowanie musi odbywać się jawienie w programie. Sprzętowe ustawianie znaczników xxxIF jest aktywne niezależnie od stanu bitów zezwolenia na przerwanie. W przypadku gdy generowanie przerwań jest zablokowane, znaczniki xxxIF można okresowo testować i zerować w głównej pętli programu.

W najprostszych mikrokontrolerach z rodziny Midrange, takich jak PIC16F84A, wszystkie bity związane z obsługą przerwań znajdują się w dwóch rejestrach INTCON oraz EECON1 (leżących w przestrzeni adresowej danych), zaś układ logiczny zgłaszający wystąpienie przerwania ma strukturę przedstawioną na rys. 1.



Rys. 1. Schemat logiki układu przerwań w mikrokontrolerze PIC16F84A.

Rejestr INTCON, adres 0Bh i 8Bh

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

Dotyczy układów PIC16F84A

Rejestr EECON1, adres 88h

U-0	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
–	–	–	EEIF	WRERR	WREN	WR	RD
bit 7							bit 0

Dotyczy układów PIC16F84A

Oznaczenia:			
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,	
S = dozwolony programowy zapis tylko stanu 1, zerowanie odbywa się sprzętowo.			
„- n" = wartość po włączeniu zasilania (POR):	„1" = bit ustawiony,	„0" = bit wyzerowany,	„x" = nieznana wartość.

Opisy bitów w rejestrze INTCON:

bit 7 **GIE**: bit globalnego zezwolenia na przerwania,

bit 6 **EEIE**: bit zezwolenia na przerwanie po zakończeniu zapisu do pamięci EEPROM,

bit 5 **T0IE**: bit zezwolenia na przerwanie po przepełnieniu licznika TMR0,

bit 4 **INTE**: bit zezwolenia na przerwanie po zmianie stanu na wejściu INT/RB0. Rodzaj zbocza wywołującego przerwanie (rosnące albo opadające) zależy od stanu bitu INTEDG w rej. OPTION_REG,

bit 3 **RBIE**: bit zezwolenia na przerwanie po zmianie stanu na wejściach RB4…RB7,

bit 2 **T0IF**: znacznik ustawiany po przepełnieniu licznika TMR0,

bit 1 **INTF**: znacznik ustawiany po zmianie stanu na wejściu INT/RB0,

bit 0 **RBIF**: znacznik ustawiany po zmianie stanu dowolnego z wejść RB4…RB7.

Opisy bitów w rejestrze EECON1:

bit 4 **EEIF**: znacznik ustawiany sprzętowo po zakończeniu zapisu do pamięci EEPROM (musi być zerowany programowo),

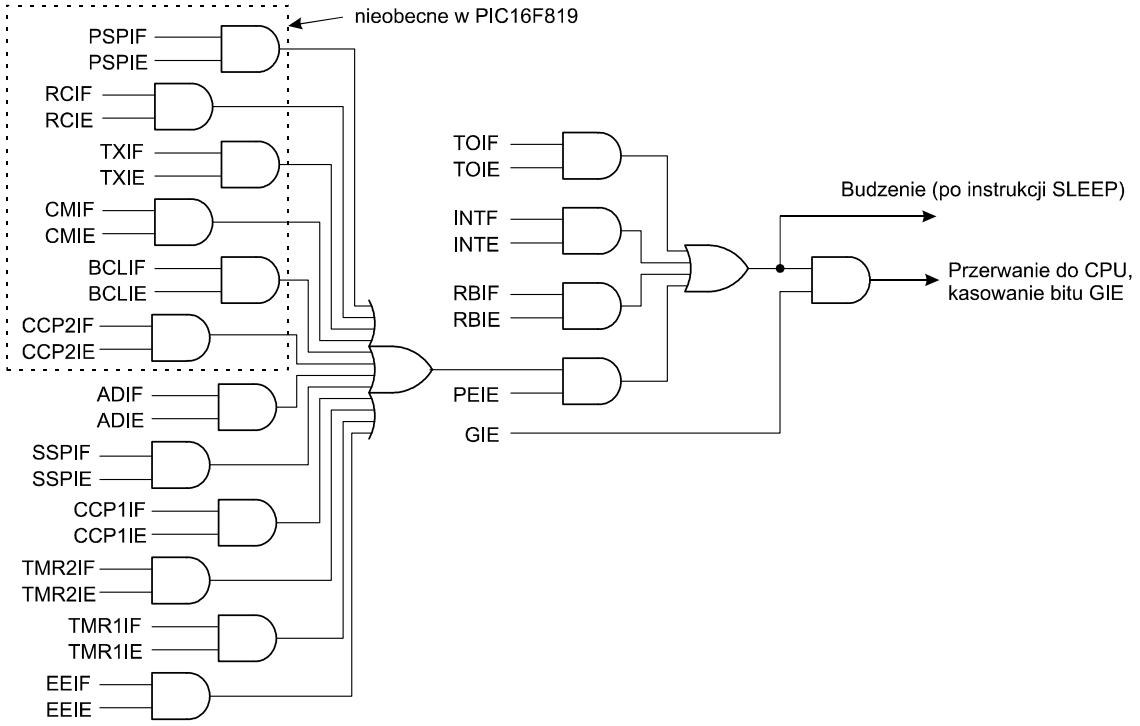
Pozostałe bity z rejestru EECON1 pominiemy jako nieistotne w niniejszym ćwiczeniu.

W nowszych konstrukcjach mikrokontrolerów wprowadzono szereg specjalistycznych urządzeń peryferyjnych, które także mogą zgłaszać przerwania. Typowy schemat układu logicznego zgłaszania przerwań przedstawiono na rys. 2. Oprócz globalnego maskowania przerwań bitem GIE, obsługa przerwań większości układów peryferyjnych może być zamaskowana także poprzez wyzerowanie bitu PEIE, który w rejestrze INTCON zastąpił bit EEIE występujący w starszych mikrokontrolerach.

Rejestr INTCON, adres 0Bh i 8Bh

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

Dotyczy układów PIC16F819 oraz PIC16F877A



Rys. 2. Schemat logiki układu przerwań w mikrokontrolerach PIC16F877A i PIC16F819.

Rejestr EECON1 występujący w mikrokontrolerach PIC16F84A został zlikwidowany w nowszych konstrukcjach. Bity zezwoleń dla większości przerwań zostały umieszczone w nowych rejestrach specjalnych PIE1 i PIE2, natomiast znaczniki zgłoszeń przerwań w nowych rejestrach PIR1 i PIR2. Ilość wykorzystanych bitów i format rejestrów zależy od wyposażenia mikrokontrolera w układy peryferyjne.

W układach scalonych PIC16F877A format rejestrów związanych z zezwoleniem na przerwania jest następujący:

Rejestr PIE1, adres 8Ch

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Rejestr PIR1, adres 0Ch

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Rejestr PIE2, adres 8Dh

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
–	CMIE	–	EEIE	BCLIE	–	–	CCP2IE
bit 7							bit 0

Rejestr PIR2, adres 0Dh

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
–	CMIF	–	EEIF	BCLIF	–	–	CCP2IF
bit 7							bit 0

Dotyczy układów PIC16F877A. W układach PIC16F819 nieobecne są bity oznaczone na rys. 2

Opisy bitów w rej. PIE1:
bit 7 **PSPIE**: bit zezwolenia na przerwanie po zapisie/odczycie przez port równoległy,
bit 6 **ADIE**: bit zezwolenia na przerwanie po zakończeniu konwersji w przetworniku A/D,
bit 5 **RCIE**: bit zezwolenia na przerwanie po odebraniu danych przez moduł transmisji szeregowej USART,
bit 4 **TXIE**: bit zezwolenia na przerwanie po wysłaniu danych przez moduł transmisji szeregowej USART,
bit 3 **SSPIE**: bit zezwolenia na przerwanie modułu SPI (ang. *Synchronous Serial Port*),
bit 2 **CCP1IE**: bit zezwolenia na przerwanie pierwszego modułu CCP (ang. *Capture, Compare, PWM*),
bit 1 **TMR2IE**: bit zezwolenia na przerwanie zegara 2 (ang. *Timer 2*),
bit 0 **TMR1IE**: bit zezwolenia na przerwanie zegara 1 (ang. *Timer 1*).

Opisy bitów w rej. PIE2:
bit 7 **Niezaimplementowane**: przy odczycie wartość „0”,
bit 6 **CMIE**: bit zezwolenia na przerwanie generowane przez komparatory analogowe,
bit 5 **Niezaimplementowane**: przy odczycie wartość „0”,
bit 4 **EEIE**: bit zezwolenia na przerwanie po zakończeniu zapisu do pamięci EEPROM,
bit 3 **BCLIE**: bit zezwolenia na przerwanie przy kolizji magistrali,
bit 2-1 **Niezaimplementowane**: przy odczycie wartość „0”,
bit 0 **CCP2IE**: bit zezwolenia na przerwanie drugiego modułu CCP (ang. *Capture, Compare, PWM*).

Opisy bitów znaczników xxxIF w rejestrach PIR1 i PIR2 odpowiadają opisanym powyżej bitom zezwoleń xxxIE. W przypadku układów scalonych PIC16F819 niektóre bity w rejestrach PIE1, PIE2, PIR1 oraz PIR2 pozostają niezaimplementowane i odczytywane są zawsze jako zero. Bity niezaimplementowane w PIC16F819 zaznaczono na schemacie na rys 2.

4.6. Pomiar czasu wewnątrz mikrokontrolera

4.6.1. Opóźnienie o zadanej długości

W bibliotece standardowej języka C dostępna jest funkcja oczekująca przez czas potrzebny na wykonanie zadanej liczby instrukcji prostych (tzn. wykonywanych przez 4 cykle zegarowe; nie dotyczy instrukcji skoków)

```
void _delay(unsigned long n);
```

Ponadto w bibliotece standardowej zdefiniowano makrodefinicje, które ułatwiają wprowadzanie opóźnienia trwającego przez zadany czas:

```
_ _delay_us(x)      // opóźnienie o zadaną liczbę mikrosekund
_ _delay_ms(x)      // opóźnienie o zadaną liczbę milisekund
```

Przed użyciem tych makrodefinicji trzeba zdefiniować wcześniej stałą _XTAL_FREQ określającą szybkość pracy oscylatora taktującego mikrokontroler w [Hz]. W przypadku wykorzystywania rezonatora kwarcowego dostępnego w zestawie ZL4PIC stała ta może przyjąć tylko jedną wartość:

```
#define _XTAL_FREQ  4000000L
```

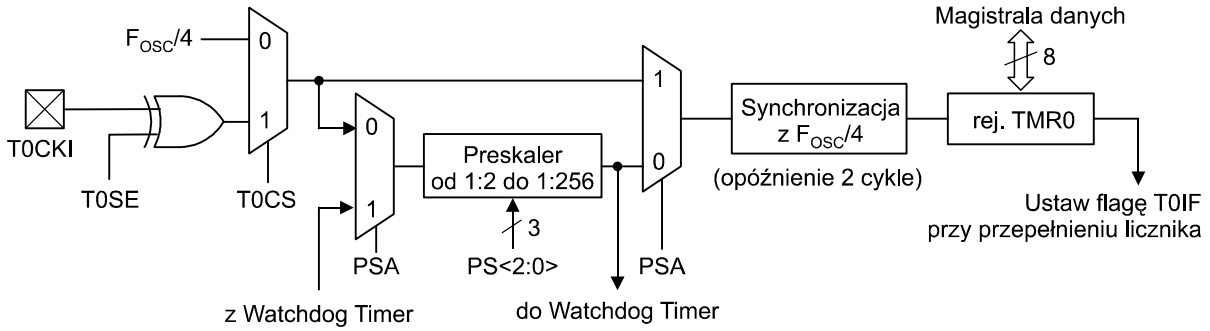
Funkcja `_delay` oraz odwołujące się do niej makrodefinicje `__delay_us` i `__delay_ms` wykorzystają pętlę programową o liczbie powtórzeń zależnej od wymaganego czasu oczekiwania. W przypadku gdy aktywna jest obsługa przerwań sprzętowych, rzeczywiste opóźnienie może być zmienne, jednak nie krótsze niż podana wartość. Do precyzyjnego odmierzania czasu należy wykorzystać opisane dalej zegary sprzętowe.

4.6.2. Zegar 0

Zegar 0 (w oryginalnej dokumentacji „Timer0”) może być wykorzystany zarówno do zliczania impulsów z zegara sterującego cyklem wykonywania instrukcji w mikrokontrolerze jak również z zewnętrznego źródła podłączonego do wejścia TOCKI (inna funkcja tej linii do cyfrowe wej./wyj. RA4). Ten drugi przypadek pominiemy w niniejszym ćwiczeniu z powodu braku odpowiedniego źródła impulsów w zestawie uruchomieniowym ZL4PIC. Wyborem źródła sygnału dla zegara 0 steruje bit T0CS z rejestru OPTION_REG. Logiczną strukturę modułu zegara 0 przedstawiono na rys. 3. W przypadku wykorzystania zegara pracującego z rezonatorem kwarcowym 4 MHz dostępnym z zestawie ZL4PIC, częstotliwość sygnału taktującego zegar 0 wynosi 4 MHz/4 = 1 MHz, gdzie podział przez 4 związany jest z długością jednego cyklu przetwarzania instrukcji (dekodowanie, pobranie danych, przetwarzanie, zapisanie wyniku). W mikrokontrolerach PIC16F819 dostępny jest ponadto wewnętrzny oscylator RC o częstotliwości 8 MHz z dodatkowym podzielnikiem (opisany w rozdziale 4.1), który może zastąpić zegar wykorzystujący zewnętrzny rezonator kwarcowy.

Moduł zegara 0 zawiera 8-bitowy licznik TMR0, który jest dostępny w przestrzeni adresowej danych pod adresem 01h i może być zarówno zapisywany jak i odczytywany. Długość cyklu zegara 0 można wydłużyć przez załączenie wstępnego podzielnika, zwanego preskalerem, o maksymalnym podziale 1:256. Preskaler jest współużytkowany przez zegar 0 oraz układ „Watchdog” (pominięty w tym ćwiczeniu) i w danej chwili może być użyty tylko w jednym z tych układów. Powiązanie preskalera jest kontrolowane przez bit PSA, natomiast stopień podziału przez bity PS<2:0> z rejestru OPTION_REG. Licznik preskalera nie jest bezpośrednio dostępny. Jeżeli preskaler jest włączony w układ zegara 0, to zapis do rejestru TMR0 powoduje także wyzerowanie preskalera.

Zegar 0 podczas przepełnienia licznika TMR0 z FFh na 00h ustawia znacznik T0IF w rejestrze INTCON i może wywoływać przerwania sprzętowe (opisane w rozdziale 4.5).



Rys. 3. Schemat blokowy modułu zegara 0.

Rejestr OPTION_REG, adres 81h

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Dotyczy układów PIC16F84A, PIC16F819 oraz PIC16F877A

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n" = wartość po włączeniu zasilania (POR):	„1" = bit ustawiony,	„0" = bit wyzerowany.

Opisy bitów:

bit 7 **$\overline{\text{RBPU}}$**: bit sterujący rezystorami podciągających napięcie na wyjściach portu B do +5V (w kompilatorze HI-TECH C® bit ten jest dostępny poprzez zmienną nRBPU)
1 = podciąganie napięcia w porcie B wyłączone,
0 = podciąganie napięcia w porcie B włączone.

bit 6 **INTEDG**: bit wyboru zbocza wywołującego przerwanie
1 = przerwanie wyzwalane zboczem rosnącym na wejściu RB0/INT,
0 = przerwanie wyzwalane zboczem opadającym na wejściu RB0/INT.

bit 5 **T0CS**: wybór źródła sygnału zegarowego dla licznika TMR0
1 = przejścia na wejściu RA4/TOCKI,
0 = wewnętrzny zegar cyklu instrukcji (CLKO).

bit 4 **T0SE**: bit wyboru zbocza inkrementującego licznik TMR0
1 = inkrementacja po zboczu opadającym na wejściu RA4/TOCKI,
0 = inkrementacja po zboczu rosnącym na wejściu RA4/TOCKI.

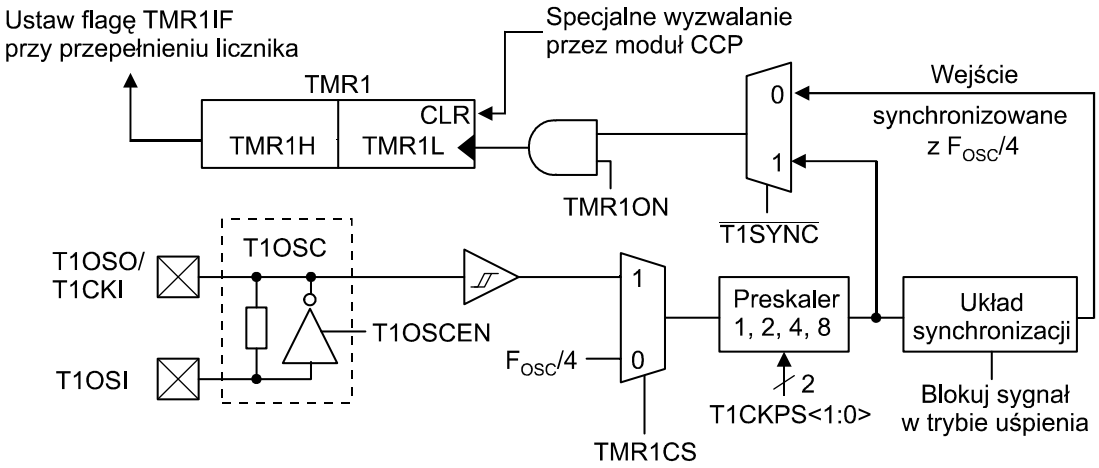
bit 3 **PSA**: bit powiązania preskalera
1 = preskaler przypisany do WDT (*Watchdog Timer*),
0 = preskaler przypisany do zegara 0.

bit 2-0 **PS<2:0>**: bity wyboru podziału częstotliwości w preskalerze:

wartości bitów	podział dla TMR0	podział dla WDT
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

4.6.3. Zegar 1

Zegar 1 (w oryginalnej dokumentacji „Timer1”) jest układem 16-bitowego licznika/zegara, który może być wykorzystany zarówno do zliczania impulsów z zegara sterującego cyklem wykonywania instrukcji w mikrokontrolerze jak również z zewnętrznego źródła podłączonego do wejścia T1CKI (inna funkcja tej linii to cyfrowe wej./wyj. RB6). Ponadto istnieje możliwość podłączenia zewnętrznego rezonatora kwarcowego pomiędzy linie T1OSO/T1CKI oraz T1OSI, który w połączeniu z obwodami wewnątrz mikrokontrolera utworzy dodatkowy oscylator dedykowany dla zegara 1 i niezależny od oscylatora cyklu instrukcji. Załączeniem tego oscylatora steruje bit T1OSCEN, natomiast wyborem źródła sygnału dla zegara 1 steruje bit TMR1CS z rejestru T1CON. Ze względu na brak odpowiedniego wyposażenia zestawu uruchomieniowego ZL4PIC w niniejszym ćwiczeniu ograniczymy się do wykorzystania zegara 1 tylko z oscylatorem cyklu instrukcji. Logiczną strukturę modułu zegara 1 przedstawiono na rys. 4.



Rys. 4. Schemat blokowy modułu zegara 1.

16-bitowy licznik zegara 1 jest dostępny w przestrzeni adresowej danych w postaci dwóch 8-bitowych rejestrów: mniej znaczący bajt licznika w rejestrze TMR1L pod adresem 0Eh oraz bardziej znaczący bajt TMR1H pod adresem 0Fh. Zapis i odczyt 16-bitowego uruchomionego licznika przy wykorzystaniu 8-bitowych operacji dostępnych w mikrokontrolerze wymaga specjalnych środków ostrożności. W przypadku zapisu licznika, najpierw należy wyzerować rejestr TMR1L w celu zabezpieczenia się przed przepełnieniem tej części licznika i przeniesieniem do rejestru TMR1H pomiędzy zapisami do rejestrów:

```
void SetTimer1(unsigned int n1) {
  ; wejście z zablokowanymi przerwaniami
  TMR1L = 0;
  TMR1H = (unsigned char)(n1 >> 8);
  TMR1L = (unsigned char)n1;
}
```

W przypadku odczytu licznika włączonego zegara, rejestr TMR1H należy odczytać dwukrotnie: przed oraz po odczycie rejestru TMR1L. Odczyt dwóch różnych wartości z rejestru TMR1H świadczy o wystąpieniu przepełnienia rejestru TMR1L i przeniesienia do TMR1H – w takim przypadku serie odczytów należy powtórzyć, np:

```
unsigned int GetTimer1(void) {
  unsigned char HByte;
  unsigned char LByte;
  do{
    HByte = TMR1H;
    LByte = TMR1L;
  }while(TMR1H != HByte);
  return ((unsigned int)HByte) << 8) | LByte;
}
```

Długość cyklu zegara 1 można wydłużyć przez załączenie wstępnego podzielnika, zwanego preskalerem, o maksymalnym podziale 1:8. Stopień podziału w preskalerze określają bity T1CKPS<1:0> z rejestru T1CON. Licznik preskalera nie jest bezpośrednio dostępny. Licznik ten jest zerowany podczas zapisów do rejestrów TMR1L oraz TMR1H.

Po włączeniu zasilania zegar 1 jest wyłączony i można go włączyć przez ustawienie bitu TMR1ON w rejestrze T1CON.

Zegar 1 podczas przepełnienia 16-bitowego licznika TMR1H:TMR1L z FFFFh na 0000h ustawia znacznik TMR1IF w rejestrze PIR1 i może wywoływać przerwania sprzętowe (przerwania opisano w rozdziale 4.5).

Rejestr T1CON, adres 12h

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
–	–	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Dotyczy układów PIC16F819 oraz PIC16F877A

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n” = wartość po włączeniu zasilania (POR):	„1” = bit ustawiony,	„0” = bit wyzerowany.

Opisy bitów:

bit 7-6 **Niezaimplementowane**: przy odczycie wartość „0”

bit 5-4 **T1CKPS<1:0>**: stopień podziału w preskalerze
11 = podział 1:8,
10 = podział 1:4,
01 = podział 1:2,
00 = podział 1:1.

bit 3 **T1OSCEN**: bit włączenia oscylatora zegara 1
1 = oscylator jest włączony,
0 = oscylator jest wyłączony.

bit 2 $\overline{\text{\textbf{T1SYNC}}}$: bit synchronizacji zewnętrznego zegara (w kompilatorze HI-TECH C® bit ten jest dostępny poprzez zmienną nT1SYNC)
Jeżeli TMR1CS = 1:
1 = nie synchronizuj sygnału z zewnętrznego zegara,
0 = synchronizuj sygnał z zewnętrznego zegara z zegarem cyklu instrukcji.
Jeżeli TMR1CS = 0:
Ten bit jest ignorowany; zegar 1 używa wewnętrznego oscylatora.

bit 1 **TMR1CS**: bit wyboru źródła sygnału dla zegara 1
1 = zewnętrzny oscylator wykorzystujący wyprowadzenie T1OSO/T1CKI (aktywne zbocze rosnące),
0 = wewnętrzny oscylator ($F_{OSC}/4$).

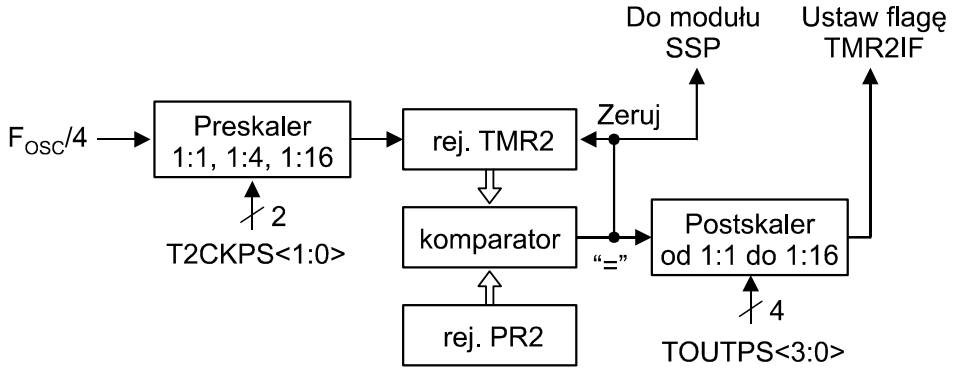
bit 0 **TMR1ON**: bit włączenia zegara 1
1 = zegar 1 włączony,
0 = zegar 1 zatrzymany.

4.6.4. Zegar 2

Zegar 2 (w oryginalnej dokumentacji „Timer2") może być wykorzystany wyłącznie do odmierzania czasu na podstawie impulsów z zegara sterującego cyklem wykonywania instrukcji w mikrokontrolerze. Logiczną strukturę modułu zegara 2 przedstawiono na rys. 5. Moduł zegara 2 zawiera 8-bitowy licznik TMR2 dostępny w przestrzeni adresowej danych pod adresem 11h. Licznik ten jest automatycznie zerowany po osiągnięciu wartości przechowywanej w rejestrze PR2 pod adresem 92h, tak więc w celu zaprogramowania licznika o okresie N cykli należy wpisać do rejestru PR2 wartość N-1. Długość cyklu zegara 2 można wydłużyć przez załączenie wstępnego podzielnika, zwanego preskalerem, który udostępnia podział 1:1, 1:4 oraz 1:16 wybierany przy pomocy bitów T2CKPS<1:0> z rejestru T2CON. Ponadto dostępny jest drugi podzielnik, zwany postskalerem, który dzieli impulsy zerowania licznika TMR2 przez dowolną wartość z zakresu od 1:1 do 1:16 wybieraną przy pomocy bitów TOUTPSM<3:0> z rejestru T2CON. Liczniki preskalera i postskalera nie są bezpośrednio dostępne. Liczniki te są zerowane podczas zapisu do rejestrów TMR2 oraz T2CON.

Przepełnienie licznika postskalera powoduje ustawienie znacznika TMR2IF w rejestrze PIR1 a ponadto może zostać wywołane przerwanie sprzętowe (opisane w rozdziale 4.5).

Po włączeniu zasilania zegar 2 jest wyłączony i można go włączyć przez ustawienie bitu TMR2ON w rejestrze T2CON.



Rys. 5. Schemat blokowy modułu zegara 2.

Rejestr T2CON, adres 12h

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
–	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Dotyczy układów PIC16F819 oraz PIC16F877A

Oznaczenia:		
R = bit do odczytu,	W = bit do zapisu,	U = bit niezaimplementowany, odczytywany jako 0,
„- n" = wartość po włączeniu zasilania (POR):	„1" = bit ustawiony,	„0" = bit wyzerowany.

Opisy bitów:

bit 7 **Niezaimplementowane**: przy odczycie wartość „0"

bit 6-3 **TOUTPS<3:0>**: stopień podziału w postskalerze
0000 = podział 1:1,
0001 = podział 1:2,
0010 = podział 1:3,
…
1111 = podział 1:16.

bit 2 **TMR2ON**: bit załączenia zegara 2
1 = zegar jest włączony,
0 = zegar 2 jest wyłączony.

bit 1-0 **T2CKPS<1:0>**: stopień podziału w preskalerze
00 = podział 1:1,
01 = podział 1:4,
1x = podział 1:16.

4.7. Programowanie wyświetlacza alfanumerycznego LCD

Wyświetlacz alfanumeryczny LCD typu WC1602A zastosowany w zestawie uruchomieniowym ZL4PIC jest zgodny z wyświetlaczem HD44780 firmy Hitachi, który stał się nieformalnym standardem naśladowanym przez wielu innych producentów. Wyświetlacz ten komunikuje się z mikrokontrolerem za pomocą ośmiu linii danych oznaczonych D0…D7 oraz trzech sygnałów sterujących $R/\overline{W}$, RS i E. Magistrala danych może pracować w dwóch trybach:

- 8-bitowym – cały bajt danych lub cały kod instrukcji przesyłane są w jednym cyklu,
- 4-bitowym – 8-bitowe dane i kody instrukcji przesyłane są po połowie - najpierw bardziej znacząca połówka bajtu a następnie połówka mniej znacząca. W tym trybie wykorzystywane są tylko linie danych D4…D7. Brak podłączenia linii wyświetlacza D0…D3 w zestawie ZL4PIC powoduje, że tryb 4-bitowy jest jedynym możliwym. Sposób podłączenia wszystkich linii wyświetlacza w zestawie ZL4PIC opisano w tabeli 3 oraz przedstawiono na schemacie na rys. C.1 w Aneksie C.

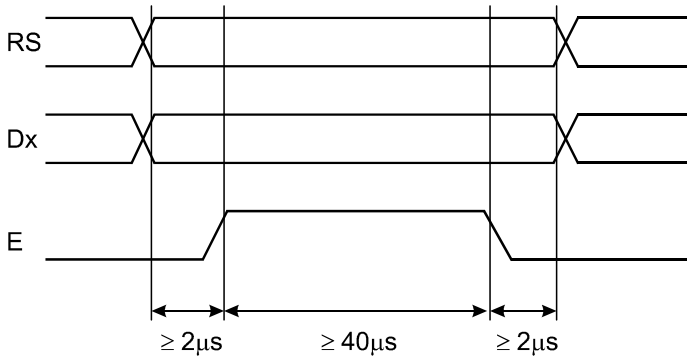
Tabela 3. Funkcje wyprowadzeń wyświetlacza alfanumerycznego i ich podłączenie w zestawie ZL4PIC.

Wyprowadzenie	Symbol	Aktywny	Funkcja	Podłączenie w ZL4PIC
1	V_{SS}	0 V	Minus zasilania	V_{SS}
2	V_{DD}	5,0 V	Plus zasilania	V_{DD}
3	V0	–	Regulacja kontrastu	potenc. R25
4	RS	H/L	Wybór rejestru	RB2
5	R/W	H/L	H: odczyt / L: zapis	V_{SS} (odczyt niemożliwy)
6	E	H	Sygnał zezwolenia	RB3
7	DB0	H/L	Linia danych	–
8	DB1	H/L	Linia danych	–
9	DB2	H/L	Linia danych	–
10	DB3	H/L	Linia danych	–
11	DB4	H/L	Linia danych	RB4
12	DB5	H/L	Linia danych	RB5
13	DB6	H/L	Linia danych	RB6
14	DB7	H/L	Linia danych	RB7
15	A	(+5,0V)	Zasilanie podświetlania LED (+)	–
16	K	(–)	Zasilanie podświetlania LED (–)	–

Znaczenie sygnałów sterujących wymianą danych z wyświetlaczem jest następujące:

- Sygnał $R/\overline{W}$ określa kierunek przesyłania danych po magistrali D0…D7, jednakże w zestawie uruchomieniowym ZL4PIC złącze wyświetlacza związane z sygnałem $R/\overline{W}$ jest trwale zwarte z masą i możliwy jest wyłącznie zapis do wyświetlacza.

- Sygnał RS służy do wyboru jednego z dwóch rejestrów wewnętrznych sterownika LCD:
  RS = 0 wybrany rejestr instrukcji IR (ang. *instruction register*),
  RS = 1 wybrany rejestr danych DR (ang. *data register*).
- Sygnał E służy do uaktywnienia wymiany danych pomiędzy mikrokontrolerem i wyświetlaczem gdy E = 1. Bezpiecznym rozwiązaniem jest przyjęcie, że zmiany wszystkich pozostałych linii są dokonywane tylko wtedy gdy wymiana danych jest nieaktywna podczas E = 0. Linia E powinna być utrzymywana w stanie E = 1 co najmniej przez 40 µs (rys. 6). Pomiędzy ustabilizowaniem stanu linii RS i linii danych a załączeniem linii E powinno upłynąć co najmniej 2 µs. Poszczególne modele wyświetlaczy mogą różnić się szczegółowymi wymaganiami dotyczącymi protokołu transmisji i wymaganych minimalnych opóźnień, jednakże wykorzystanie maksymalnej katalogowej szybkości transmisji może prowadzić do niezgodności programu z innymi podobnymi wyświetlaczami.



Rys. 6. Przebiegi czasowe podczas bezpiecznej wersji zapisu do wyświetlacza alfanumerycznego.

Wyświetlacz alfanumeryczny posiada trzy rozdzielne obszary wewnętrznej pamięci:

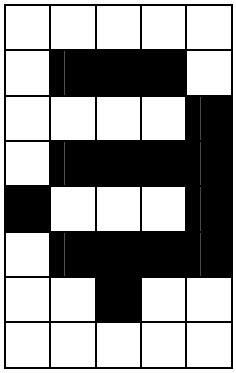
- CG ROM (ang. *Character Generator ROM*) – pamięć stała generatora znaków, w której znajdują się wzorce wyświetlanych znaków o wymiarach 5×8 pikseli. Znaki o kodach 32…125 pokrywają się ze standardem ASCII. Pełną tabelę wzorów znaków podano w karcie katalogowej wyświetlacza [12].
- DD RAM (ang. *Display Data RAM*) – pamięć kodów wyświetlanych znaków o pojemności 80 bajtów. W przypadku wyświetlacza WC1602A, który posiada matrycę złożoną tylko z 2 linii po 16 znaków po inicjalizacji wyświetlacza widoczne są komórki pamięci oznaczone poniżej pogrubioną ramką:

Numer znaku w linii	1	2	3	…	…	14	15	16	17	18	19	…	…	38	39	40
Adres znaku w linii 1	00h	01h	02h	…	…	0Dh	0Eh	0Fh	10h	11h	12h	…	…	26h	27h	28h
Adres znaku w linii 2	40h	41h	42h	…	…	4Dh	4Eh	4Fh	50h	51h	52h	…	…	66h	67h	68h

Okno widoczności może być przesuwane wzdłuż 40-znakowych linii w pamięci instrukcją *Cursor or Display Shift* (patrz Tabela 4). Okno wykraczające poza graniczne adresy linii ulega zawinięciu na przeciwny kraniec linii.

- CG RAM (ang. *Character Generator RAM*) – pamięć o pojemności 64 bajtów używana do definiowania własnych ośmiu znaków (rys. 7). Definicja n-tego znaku zaczyna się od adresu 8n. Kolejnym znakom zdefiniowanym w CG RAM odpowiadają kolejne kody od 0 do 7 w pamięci DD RAM. Kody 8…15 wpisane do DD RAM dają taki sam efekt jak kody mniejsze o 8.

Zapis danych do pamięci CG RAM albo DD RAM trzeba poprzedzić instrukcją która określi typ pamięci oraz adres, od którego rozpocznie się zapis (patrz funkcje „Set CGRAM Address" oraz „Set DDRAM Address" w tabeli 4). Po zapisaniu 1 bajtu danych bieżący adres w pamięci może być automatycznie zwiększany albo zmniejszany w zależności od stanu bitu I/D przesłanego komendą „Entry Mode Set".



Matryca znaku	dane w CG RAM	adres w CG RAM	kod znaku w DD RAM
	0000 0000	08h	
	0000 1110	09h	
	0000 0001	0Ah	
	0000 1111	0Bh	01h
	0001 0001	0Ch	
	0000 1111	0Dh	
	0000 0100	0Eh	
	0000 0000	0Fh	

Rys. 7. Definiowanie wzorców własnych znaków w pamięci CG RAM.

Tabela 4. Tabela instrukcji wyświetlacza alfanumerycznego LCD typu WC1602A.

Instrukcja	Kod instrukcji										Opis	Czas wykonania
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Zapisanie ekranu spacjami i ustawienie kursora na początek (adres 0)	1,53 ms
Return Home	0	0	0	0	0	0	0	0	1	–	Ustaw kursor na początek ekranu (adres 0)	1,53 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Ustawienie kierunku przesuwania kursora i włączenie przesuwania zawartości ekranu	39 μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Włączenie/wyłączenie wyświetlacza (D), kursora (C) i migania kursora (B).	39 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	–	–	Przesunięcie zawartości wyświetlacza albo kursora o 1 znak bez zmiany w pamięci DDRAM.	39 μs
Function Set	0	0	0	0	1	DL	N	F	–	–	Ustaw szerokość słowa danych (8/4-bity), liczbę linii wyświetlacza (2/1), oraz typ fontu (5x11/5x8 punktów).	39 μs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Ustaw bieżący adres w pamięci CGRAM do zapisu lub odczytu.	39 μs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Ustaw bieżący adres w pamięci DDRAM do zapisu lub odczytu.	39 μs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Odczytaj znacznik zajętości wyświetlacza wewnętrzną operacją oraz bieżący adres w DDRAM.	0 μs
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Zapis danej do pamięci DDRAM albo CGRAM	43 μs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Odczyt danej z pamięci DDRAM albo CGRAM	43 μs

Oznaczenia:
AC6…AC0 – adres w pamięci CGRAM lub DDRAM,
B = 1: miganie kursora włączone, B = 0: miganie wyłączone,
BF = 1: moduł zajęty, BF = 0: wysłane dane zostały przyjęte,
C = 1: kursor widoczny, C = 0: kursor niewidoczny,
D7…D0 – dane zapisywane/odczytywane do/z pamięci DDRAM albo CGRAM,
D = 1: wyświetlanie włączone, D = 0: wyświetlanie wyłączone,
DL = 1: 8-bitowa linia danych, DL = 0: 4-bitowa linia danych,
F = 1: znaki 5×11 punktów (niedostępne w wyświetlaczu WC1602A), F = 0: znaki 5×8 punktów,
I/D = 1: inkrementacja adresu pamięci po zapisie do rej. DR, I/D = 0: dekrementacja adresu,
N = 1: tryb 2-liniowy, N = 0: tryb 1-liniowy,
R/L = 1: przesuwanie wyświetlacza/kursora w prawo, R/L = 0: przesuwanie wyświetlacza/kursora w lewo,
S/C = 1: przesuwanie zawartości wyświetlacza, S/C = 0: przesuwanie kursora,
SH = 1: przesuwanie zawartości całego ekranu włączone, SH = 0: przesuwanie wyłączone.

Wyświetlacz po włączeniu zasilania wymaga inicjalizacji oraz ustawienia do pracy z 4-bitową linią danych. Ciąg instrukcji niezbędnych do wykonania na tym etapie i wymagane odstępy czasu różnią się w zależności od źródła danych, jednakże istnieje zawsze pewna część wspólna. Poniżej przedstawiono przykłady kilu praktycznych funkcji napisanych w języku C i przeznaczonych do obsługi wyświetlacza alfanumerycznego LCD. Funkcja inicjalizacji została napisana z pewnym nadmiarem w porównaniu do minimum koniecznego według karty katalogowej wyświetlacza WC1602A, jednakże należy mieć na uwadze możliwość wymiany wyświetlacza.

```
#include<htc.h>
// Przetestowano z układami: PIC16F84A, PIC16F819, PIC16F877A
// Rejestr konfiguracyjny ustawiany w oknie "Configuration bits"

#define _XTAL_FREQ    4000000L
#define LCD_PORT      PORTB
#define LCD_EN        RB3
#define LCD_RS        RB2
#define LCD_RS_MASK   4

void lcd_write_nibble(unsigned char c) {
  // Przesłanie do wyświetlacza 4-bitów danych oraz ustawienie linii sterujących:
  // bity 4...7 - dane
  // bit 2 - stan linii RS

  LCD_PORT = c;
  __delay_us(2);
  LCD_EN = 1;
  __delay_us(40);
  LCD_EN = 0;
}// lcd_write_nibble

void lcd_write_cmd(unsigned char c) {
  // Przesłanie do wyświetlacza 8-bitów instrukcji w dwóch częściach
  LCD_RS = 0;
  LCD_EN = 0;
  // RS=0, E=0, zapisz górne 4 bity komendy
  lcd_write_nibble(c & 0xF0);

  // RS=0, E=0, zapisz dolne 4 bity komendy
  lcd_write_nibble(c << 4);
  if (c <= 3) __delay_ms(2); else __delay_us(40);
}// lcd_write_cmd

void lcd_init(void) {  // Inicjalizacja wyświetlacza
  unsigned char licz;

  __delay_ms(45);      // czekaj na stabilność zasilania
  TRISB &= 0b00000011; // ustaw linie RB7...RB2 jako wyjścia
  LCD_RS = 0;
  LCD_EN = 0;
  // zapisz górne 4 bity komendy "Function Set", interfejs 8-bitowy
  LCD_PORT = 0x30;
  __delay_us(2);

  // Wyślij trzykrotnie komendę "Function Set", interfejs 8-bitowy
  for(licz=1; licz<=3; licz++) {
    LCD_EN = 1;
    __delay_us(40);
    LCD_EN = 0;
    __delay_ms(5);
  }

  // RS=0, E=0, interfejs 4-bitowy
  lcd_write_nibble(0x20);
  __delay_us(40);
```

```
  lcd_write_cmd(0x28);  // Function Set: interfejs 4-bitowy, N=1, F=0
  lcd_write_cmd(0x08);  // Display OFF
  lcd_write_cmd(0x01);  // Clear Display
  lcd_write_cmd(0x06);  // Entry Mode Set I/D=1, SH=0
  lcd_write_cmd(0x0C);  // Display ON, D=1, C=0, B=0
}// lcd_init

void DisplayClear(void) {
  // Wyczyszczenie ekranu LCD i ustawienie kursora w lewym górnym rogu
  lcd_write_cmd(0x01);
}// DisplayClear

void gotoxy(unsigned char x, unsigned char y) {
  // Ustawienie współrzędnych pisania w pamięci DDRAM na wyświetlaczu LCD.
  // (1,1) - lewy, górny róg.
  lcd_write_cmd(0x80 | ((0x40*(y - 1) + x - 1) & 0x7F));
}// gotoxy

void lcd_write_data(unsigned char c) {
  // Zapisz kod znaku do pamięci DDRAM
  LCD_RS = 1;
  LCD_EN = 0;
  // RS=1, E=0, zapisz górne 4 bity danych
  lcd_write_nibble((c & 0xF0) | LCD_RS_MASK);

  // RS=1, E=0, zapisz dolne 4 bity danych
  lcd_write_nibble((c << 4) | LCD_RS_MASK);
  __delay_us(43);
}// lcd_data_data
```

4.8. Programowanie wyświetlacza numerycznego LED

4.8.1. Wprowadzenie

Zestaw uruchomieniowy ZL4PIC zawiera 4 jednocyfrowe wyświetlacze numeryczne LED ze wspólną katodą. Układ połączeń wyświetlaczy z mikrokontrolerem jest widoczny na schemacie zestawu ZL4PIC przedstawionym na rys. C.1 w załączniku C. Każdy z wyświetlaczy zbudowany jest z 7 segmentów przeznaczonych do wyświetlania wybranej cyfry oraz z ósmego segmentu, którym jest kropka dziesiętna. Wszystkie wyjścia portu B w mikrokontrolerze, tzn. RB0, RB1, …, RB7 wykorzystano do wyboru jednego z segmentów wyświetlacza. Ze względu na znaczą liczbę wszystkich wyprowadzeń czterech wyświetlaczy, połączono anody analogicznych segmentów ze wszystkich czterech wyświetlaczy. Wybór aktualnie świecącego wyświetlacza dokonywany jest przy pomocy wyjść RA0, RA1, RA2 i RA3 mikrokontrolera, przy czym wyjście RA3 związane jest z wyświetlaniem cyfry najmniej znaczącej, natomiast wyjście RA0 z wyświetlaniem cyfry najbardziej znaczącej. Tranzystory T1…T4 pełnią rolę wzmacniaczy prądowych a jednocześnie inwerterów zwierających katody wybranych wyświetlaczy do masy, gdy stan odpowiedniej linii RAx = 1.

Wyświetlanie różnych wzorów znaków jednocześnie na dwóch lub więcej wyświetlaczach wymaga okresowego uaktywniania kolejnych wyświetlaczy. Jeżeli częstotliwość powtarzania załączeń wynosi co najmniej 80 Hz, to wyświetlacze będą stwarzały wrażenie ciągłego i stabilnego świecenia. Zestaw ZL4PIC nie posiada żadnego sterownika wyświetlacza numerycznego LED, zatem okresowe przełączanie aktywnych segmentów wyświetlacza musi być realizowane przez mikrokontroler. Zadanie to zazwyczaj realizowane jest w procedurze obsługi przerwań generowanych przez jeden spośród sprzętowych zegarów wbudowanych w mikrokontroler. Wykorzystanie przerwań upraszcza realizację innych zadań wewnątrz głównej pętli programu.

Uwaga: w zestawie ZL4PIC linie mikrokontrolera RB0, …, RB3 wykorzystywane są także do odczytu stanu przycisków S1, …, S4. Sterowanie chociaż jednym wyświetlaczem LED uniemożliwia wykorzystanie tych przycisków. Jeżeli przyciski są konieczne, to należy rozważyć użycie wyświetlacza LCD, który pozostawia możliwość odczytu linii RB0 i RB1 połączonych z przyciskami S1 i S2. W przypadku wykorzystania mikrokontrolera PIC16F819 możliwe jest wykorzystanie przycisku S5 jako kolejnego wejścia cyfrowego, natomiast w pozostałych mikrokontrolerach przycisk ten może pełnić tylko funkcję resetu.

4.8.2. Przykładowy program odmierzający czas i sterujący wyświetlaczami LED

Program, którego tekst źródłowy podano na następnej stronie, wyświetla na wyświetlaczach numerycznych LED 4-cyfrową liczbę sekund, które upłynęły od włączenia zasilania lub od resetu. Do precyzyjnego odmierzania czasu wykorzystano sprzętowy zegar 2, który częstotliwość 1 MHz z zegara systemowego dzieli przez 2500 (zegar zliczający modulo 250 i dodatkowo postskaler 1:10). Procedura obsługi przerwania jest więc wywoływana z częstotliwością 1MHz/2500 = 400 Hz i każde wywołanie powoduje uaktywnienie wyświetlania kolejnej jednej z czterech cyfr na wyświetlaczu. Pełny cykl wyświetlania 4-cyfrowej liczby powtarzany jest z częstotliwością 100 Hz. Raz na 400 wywołań procedury obsługi przerwań zwiększany jest licznik sekund w zmiennej `LiczSek`.

```
#include<htc.h>
// Program dla mikrokontrolera PIC16F819
__CONFIG(CP_OFF & DEBUG_OFF & LVP_OFF & BOREN_ON & MCLRE_ON & PWRTE_ON & WDTE_OFF &
FOSC_XT);

unsigned int LiczInt = 0;
unsigned int LiczSek = 0;
unsigned char DigitNum = 0;

// Definicja segmentów wyświetlacza LED do wyświetlania cyfr
// 0,1,2,3,4,5,6,7,8,9
const unsigned char LedSeg[10]={
  0b00111111,  0b00000110,
  0b01011011,  0b01001111,
  0b01100110,  0b01101101,
  0b01111101,  0b00000111,
  0b01111111,  0b01101111
};

void interrupt IntHandler(void) {
  if (TMR2IE && TMR2IF) {
    TMR2IF = 0;
    PORTA = 0;
    switch(DigitNum) {
      case 0:
        PORTB = LedSeg[LiczSek % 10];     // zapal LED na linii RA0
        PORTA = 0b1000; break;
      case 1:
        PORTB = LedSeg[(LiczSek / 10) % 10];     // zapal LED na linii RA1
        PORTA = 0b0100; break;
      case 2:
        PORTB = LedSeg[(LiczSek / 100) % 10];     // zapal LED na linii RA2
        PORTA = 0b0010; break;
      case 3:
        PORTB = LedSeg[(LiczSek / 1000) % 10];     // zapal LED na linii RA3
        PORTA = 0b0001; break;
     }
     DigitNum++;
     DigitNum &= 0b00000011;

     if (++LiczInt >= 400) {
       LiczInt = 0;
       if (++LiczSek > 9999) LiczSek = 0;
     }
  }
}// IntHandler

void main(void) {
  ADCON1 = 0x06;            // ustaw linie RA0...RA5 jako cyfrowe

  PORTA = PORTB = 0;        // początkowo wszystkie wyjścia w stanie 0
  TRISA = 0b11110000;       // linie RA0...RA3 jako wyjścia
  TRISB = 0b00000000;       // wszystkie linie portu B wyjściowe

  T2CON = 0b01001000;       // postskaler 1:10, preskaler 1:1, zegar 2 wyłączony
  TMR2 = 0;                 // zeruj licznik zegara 2
  PR2 = 249;                // ustaw okres zegara 2 na (249+1)

  TMR2IF = 0;               // wytrzyj flagę zgłoszenia przerwania zegara 2
  TMR2IE = 1;               // odblokuj przerwania zegara 2
  PEIE = 1;                 // odblokuj przerwania układów peryferyjnych
  TMR2ON = 1;               // włącz zegar 2
  ei();                     // odblokuj wszystkie przerwania

  do{                       // wszystkie czynności wykonywane są
  }while(1);                // w procedurze obsługi przerwania
}// main
```

4.9. Programowanie brzęczyka piezoelektrycznego

Brzęczyk piezoelektryczny dostępny w zestawie uruchomieniowym ZL4PIC można połączyć jedynie z linią mikrokontrolera RA2, co wyklucza użycie sprzętowych mechanizmów generowania okresowych przebiegów. Wytwarzanie ciągłego dźwięku o ustalonej częstotliwości tonu podstawowego sprowadza się do okresowego przełączania linii RA2 skonfigurowanej jako cyfrowe wyjście pomiędzy stanami 0 i 1.

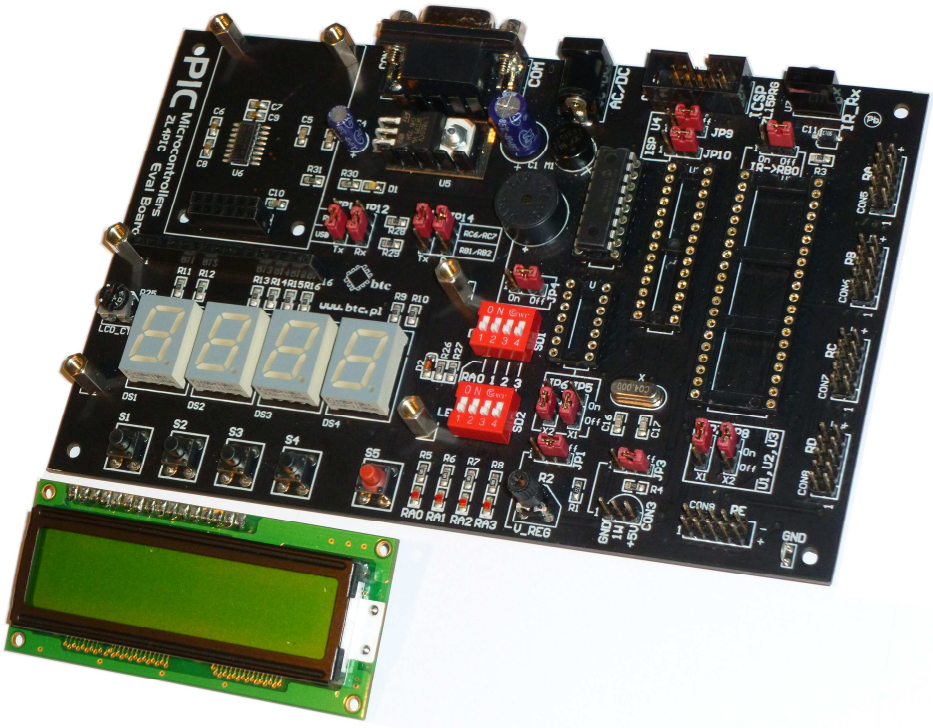
Dla częstotliwości rzędu kilku kHz i większych wykorzystanie procedury obsługi przerwań sprzętowych do generowania sygnału sterującego brzęczykiem może okazać się zadaniem stawiającym zbyt wysokie wymagania przed mikrokontrolerem. W takiej sytuacji konieczne jest umieszczenie obsługi brzęczyka w głównej pętli programu. Synchronizację brzęczyka z wybranym zegarem sprzętowym można uzyskać przy niskich narzutach np. przez oczekiwanie na ustawienie flagi zgłoszenia przerwania xxxIF związanej z odpowiednim zegarem przy zablokowanym mechanizmie wywoływania funkcji obsługi przerwań (system przerwań sprzętowych omówiono w rozdziale 4.5).

5. Dostępna aparatura

5.1. Zestaw uruchomieniowy

Zestaw ZL4PIC umożliwia uruchamianie programów na mikrokontrolerach z rodziny PIC12, PIC16 i niektórych PIC18 w obudowach DIP8, DIP14, DIP18, DIP28 i DIP40. Wygląd zestawu przedstawiono na rys. 3, natomiast schemat elektryczny na rys. F.1 w aneksie F. Zestaw wyposażono m.in. w następujące peryferia:

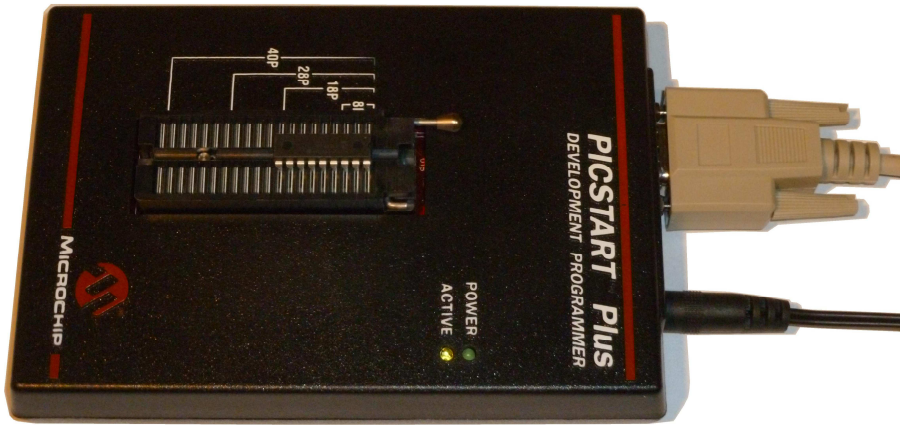
- cztery numeryczne wyświetlacze siedmiosegmentowe,
- wyświetlacz LCD 2×16 znaków, który można zamontować ponad wyświetlaczem numerycznym (oba typy wyświetlaczy nie mogą być jednocześnie używane),
- cztery przyciski ogólnego przeznaczenia S1,…,S4 podłączone do wejść mikrokontrolera RA0…RA3.
- przycisk resetu mikrokontrolera S5 dołączony do wejścia $\overline{\text{MCLR}}$, w przypadku niektórych mikrokontrolerów (np. PIC16F819) może być użyty także jako dodatkowy przycisk funkcyjny,
- cztery diody świecące, które w zależności od potrzeb mogą być przyłączone przełącznikiem SD2 do wejść mikrokontrolera RB0…RB3,
- przetwornik piezoceramiczny do sygnalizacji akustycznej,
- potencjometr V_REG do regulacji napięcia na wejściu przetwornika A/C dostępnego w niektórych mikrokontrolerach (np. PIC16F819, PIC16F877A),
- port RS232 umożliwiający komunikację z np. komputerem,
- odbiornik promieniowania podczerwonego, np. do odbioru sygnałów z pilota zdalnego sterowania,
- złącze magistrali 1Wire.



Rys. 3. Zestaw uruchomieniowy ZL4PIC ze zdjętym wyświetlaczem LCD.

5.2. Programator mikrokontrolerów

Przygotowany samodzielnie program przed uruchomieniem go w zestawie ZL4PIC należy zapisać do pamięci mikrokontrolera przy użyciu programatora PICSTART Plus. Programator ten jest przeznaczony do współpracy z pakietem MPLAB zainstalowanym na komputerze z systemem Windows. Komunikacja programatora z komputerem odbywa się przez złącze RS232. W przypadku braku takiego złącza w komputerze programator należy podłączyć za pośrednictwem konwertera RS232 ↔ USB.



Rys. 4. Programator PICSTART Plus z mikrokontrolerem zaciśniętym w podstawce.

5.3. Zasilacze

Na stanowisku doświadczalny wykorzystywane są dwa zasilacze:

1. Zasilacz 9V DC HiTRON model HES10-09007-0-7 dedykowany wyłącznie do pracy z programatorem PICSTART Plus.
2. Uniwersalny zasilacz dogniazdkowy z wielopozycyjnym przełącznikiem napięcia DC. Przed rozpoczęciem pracy z zastawem ZL4PIC zalecane jest wybranie w zasilaczu napięcia 9V. Napicie to jest obniżane do 5V przez stabilizator wbudowany w zestaw ZL4PIC.

UWAGA: wtyki wymienionych zasilaczy są zgodne, jednakże nie należy podłączać zasilacza HiTRON w sposób niezgodny z jego przeznaczeniem.

5.4. Komputer

W pracowni udostępniono komputery z systemem Windows i zainstalowanym pakietem MPLAB 8.66 i kompilatorem HI-TECH C® for PIC10/12/16 MCUs.

6. Przebieg doświadczenia

W pierwszej kolejności należy przygotować do pracy zestaw złożony z komputera klasy PC i programatora PICSTART Plus połączonego z komputerem przez port RS232. Następnie należy przygotować w środowisku MPLAB IDE program dla jednego wybranego mikrokontrolera PIC z rodziny Midrange napisany w języku C. Program ten ma stanowić rozwiązanie problemu postawionego w treści wybranego zadania. Po pomyślnym skompilowaniu programu należy go zapisać w pamięci mikrokontrolera przy wykorzystaniu programatora PICSTART Plus. Następnie zaprogramowany mikrokontroler przenosi się do zestawu uruchomieniowego ZL4PIC z odpowiednio skonfigurowanymi zworkami i przełącznikami w celu przetestowania funkcji programu.

6.1. Kolejność czynności

1. Podłączyć programator mikrokontrolerów PICSTART Plus do komputera. Podczas tej czynności komputer powinien być wyłączony. Podłączyć zasilanie programatora napięciem stałym 9V – zalecane jest używanie w tym celu wyłącznie dedykowanego zasilacza HiTRON model HES10-09007-0-7.
2. Przygotować drugi zasilacz do pracy z zestawem uruchomienionym ZL4PIC. W tym celu wybrać przełącznikiem w zasilaczu napięcie 9V. Założyć odpowiednią końcówkę na przewód wyjściowy zasilacza, która pasuje do gniazda zasilania w zestawie ZL4PIC, przy czym biegunowość napięcia wyjściowego nie ma znaczenia. Nie podłączać jeszcze napięcia zasilającego zestaw ZL4PIC.
3. Uruchomić komputer i po załadowaniu systemu uruchomić zintegrowane środowisko MPLAB IDE.
4. Przed nawiązaniem łączności z programatorem należy wskazać jego typ w pakiecie MPLAB IDE. W tym celu wybieramy z menu pozycję `Programmer/Select Programmer` i dalej na liście model dostępny w pracowni `PICSTART Plus`. Następnie aktywujemy programator wybierając z menu `Programmer/Enable Programmer`. Jeżeli ta operacja nie powiodła się, to należy sprawdzić: zasilanie programatora (w programatorze powinna świecić się zielona dioda „POWER”), połączenie z komputerem przez port RS232 oraz numer portu szeregowego ustawiony w pakiecie MPLAB IDE w oknie dialogowym `Programmer` otwieranym przez wybranie z menu pozycji `Programmer/Settings`.
5. W porozumieniu z prowadzącym zajęcia wybrać zadanie, które będzie realizował mikrokontroler w zestawie uruchomieniowym ZL4PIC. Przykładowe zadania zebrano w następnym rozdziale. Wykonawcy ćwiczeń mogą zaproponować własne zadania.
6. Wybrać odpowiedni typ mikrokontrolera do zaprogramowania i zgłosić decyzję obsłudze pracowni. W pracowni powinny być dostępne przynajmniej mikrokontrolery typu PIC16F84A-04, PIC16F819 oraz PIC16F877A.
7. Użycie kompilatora języka C w środowisku MPLAB IDE wymaga utworzenia projektu, nawet jeżeli w skład projektu wejdzie tylko jeden plik źródłowy. Nowy projekt można stworzyć wykonując następujące kroki:
   a. Wybrać z menu rozwijalnego `Project/Project Wizard...`.
   b. Następnie w oknie dialogowym wybrać typ mikrokontrolera.

c. Kolejne okno dialogowe służy do wyboru używanego kompilatora – na liście `Active Toolsuite` wybrać `HI-TECH Universal ToolSuite` i kliknąć przycisk `Dalej`.
d. W następnym oknie dialogowym podajemy nazwę projektu i ścieżkę dostępu do katalogu, w którym projekt ma być utworzony.
e. Kolejne okno dialogowe daje możliwość dołączenia do projektu istniejących już plików. Ponieważ w tym ćwiczeniu tworzymy projekt od podstaw prawdopodobnie możliwość ta nie zostanie wykorzystana i trzeba tylko kliknąć przycisk `Dalej`.
f. Jeżeli wcześniejsze kroki przebiegły pomyślnie pojawi się okno z informacją o utworzeniu nowego projektu. Kliknąć przycisk `Zakończ`.
g. W ostatnim kroku kreatora projektu pojawia się okno wyboru pliku, w którym zostanie zapisana konfiguracja obszaru roboczego (ang. *workspace*) w pakiecie MPLAB IDE. Zalecane jest potwierdzenie zaproponowanej domyślnej nazwy pliku i ścieżki do katalogu.

8. W nowo utworzonym projekcie nie ma jeszcze żadnych domyślnych plików źródłowych (patrz zakładka `Source Files`). Aby otworzyć okno edytora źródłowego tekstu programu wybrać z menu rozwijanego `File/New`. Tekst z edytora można zapisać do pliku np. przez wybranie z menu pozycji `File/Save As…` . Utworzony w ten sposób plik nie jest dołączany automatycznie do otwartego projektu. W celu dodania nowego pliku można np. wybrać z menu rozwijalnego `Project/Add Files To Project...`. Następnie w oknie dialogowym wybrać nazwę pliku do utworzenia. Aby typ pliku został prawidłowo zaklasyfikowany w projekcie należy upewnić się, że plik źródłowy posiada rozszerzenie `.c`. Ponadto zalecane jest tworzenie plików źródłowych w katalogu projektu.
9. W celu skontrolowania i ustawienia opcji projektu wybrać z menu rozwijanego `Project/Build Options.../Project`. W przypadku dokonania zmian w opcjach zapisać zmiany do pliku projektu wybierając z menu rozwijanego `Project/Save Project`.
10. Ustawić bity rejestru konfiguracyjnego mikrokontrolera. Jeżeli bity konfiguracyjne nie są ustawiane w tekście źródłowym programu, to można je ustawić w oknie dialogowym otwieranym po wybraniu z menu `Configure/Configuration Bits...`. W przypadku dokonania zmian zapisać je do pliku projektu wybierając z menu rozwijanego `Project/Save Project`. Nie należy bezkrytycznie akceptować domyślnych ustawień przyjętych w pakiecie MPLAB IDE, gdyż mogą one nie być odpowiednie do uruchomienia programu.
11. Przygotować plik źródłowy programu. Oprócz niniejszej instrukcji pomocny może okazać się podręcznik opisujący kompilator HI-TECH C® for PIC10/12/16 razem z funkcjami bibliotecznymi, który można otworzyć wybierając z menu rozwijanego `Project/HI-TECH C Manual`.
12. W celu skompilowania programu wybrać z menu rozwijanego `Project/Built`. Poprawić wszystkie błędy zgłaszane przez kompilator, gdyż w przeciwnym razie nie zostanie wygenerowany żaden kod programu do zapisania w mikrokontrolerze.
13. Zapisać kod programu w pamięci mikrokontrolera. W tym celu należy włożyć mikrokontroler do podstawki w programatorze i zacisnąć go dźwigienką. Nie ma potrzeby deaktywacji programatora przed włożeniem lub wyjęciem mikrokontrolera do/z podstawki programatora, gdyż napięcia w podstawce są załączane tylko na czas transmisji danych.

UWAGA: mikrokontroler należy odpowiednio zorientować względem podstawki, tzn. nóżka mikrokontrolera o numerze `1` oznaczona kropką wytłoczoną w rogu jego obudowy musi trafić do złącza w podstawce opisanego cyfrą `1`.

Następnie wybrać z menu `Programmer/Program`. Jeżeli nie zostały zgłoszone żadne błędy, to mikrokontroler z zapisanym programem jest gotowy do użycia. Wyjąć mikrokontroler z programatora.

14. Włożyć mikrokontroler do podstawki z zestawie uruchomieniowym ZL4PIC zwracając uwagę na odpowiednią orientację układu scalonego względem podstawki: półokrągłe nacięcie na brzegu mikrokontrolera powinno znaleźć się po stronie analogicznego nacięcia w podstawce.

    UWAGA: pamiętać o odłączeniu zasilacza zestawu ZL4PIC przed każdym włożeniem lub wyjęciem mikrokontrolera.
15. Przed załączeniem zasilania skonfigurować odpowiednio zworki JP1…JP14 oraz mikroprzełączniki w sekcjach SD1 i SD2 w zestawie ZL4PIC w zależności od wykorzystywanych składników zestawu. Zworki i przełączniki przedstawiono na schemacie zamieszczonym w Aneksie C.
16. Podłączyć zasilanie zestawu ZL4PIC. Zalecane jest podłączenie najpierw samego zasilacza do gniazda sieci 230V~ a następnie włożenie wtyczki pod napięciem 9V do gniazda oznaczonego AC/DC w zestawie ZL4PIC. W przypadku wykonania czynności w odwrotnej kolejności niektóre egzemplarze mikrokontrolerów mogą nie uruchomić się z powodu zbyt długiego czasu narastania napięcia zasilającego.
17. Przetestować działanie mikrokontrolera w zestawie ZL4PIC. Jeżeli program nie działa zgodnie z założeniami, wyciągnąć wnioski i wykonać poprawki w programie. Rozważyć konsultację z obsługą pracowni.

    UWAGA: przytrzymywanie przycisków S1…S4 powoduje zwieranie do masy linii mikrokontrolera RB0…RB4 i nie ma możliwości odłączenia tych przycisków. Zwieranie do masy tych linii, które zostały skonfigurowane jako cyfrowe wyjścia do sterowania wyświetlaczem LCD albo LED grozi uszkodzeniem mikrokontrolera.
18. Po osiągnięciu zgodności działania programu z złożeniami wykonać krótką demonstrację przed obsługą laboratorium.
19. Zakończenie pracy.
    a. Zdeaktywować programator wybierając z menu w pakiecie MPLAB IDE `Programmer/Disable Programmer`.
    b. Zamknąć pogram MPLAB IDE.
    c. Odłączyć przewody zasilania programatora i zestawu ZL4PIC oraz wyłączyć oba zasilacze z kontaktów sieci 230V~.
    d. Wykonać kopię plików związanych z projektem na własnym nośniku danych.
    e. Jeżeli obsługa pracowni zaleci uprzątnięcie stanowisk to należy wyłączyć komputer, a dopiero w drugiej kolejności odłączyć przewód z portów RS232 w komputerze i programatorze. Następnie zapakować urządzenia i przewody do odpowiednich pudełek.

6.2. Propozycje zadań realizowanych przez mikrokontroler

Należy zaprogramować mikrokontroler do realizacji wybranego zadania oraz przetestować jego działanie w zestawie uruchomieniowym ZL4PIC. Wykonawcy mogą zaproponować własne zadania.

1. Układ zastępujący kostkę do losowania liczb całkowitych z zakresu od 1 do 6. Po każdym przyciśnięciu przycisku S5 wylosowana liczba powinna wyświetlić się na jednym wybranym segmencie wyświetlacza numerycznego LED. Liczbę o wysokim stopniu losowości można uzyskać np. przez pomiar czasu z wysoką rozdzielczością i zapamiętanie licznika czasu w chwili przyciśnięcia przycisku. Następnie wynik należy zredukować do wymaganego zakresu przy wykorzystaniu działania (N mod 6) + 1, gdzie N jest licznikiem zwiększającym swoja wartość proporcjonalnie do upływu czasu. **Uwaga:** zadanie to można zrealizować jedynie na mikrokontrolerze PIC16F819. W przypadku pozostałych mikrokontrolerów dostępnych w Laboratorium Elektroniki przycisk S5 może pełnić jedynie rolę resetu, natomiast użycie przycisków S1…S4 jest wykluczone z powodu wykorzystania linii RB0…RB3 mikrokontrolera do obsługi wyświetlacza.
2. Układ, który zlicza przyciśnięcia przycisku S1 w cyklu modulo 16, zapamiętuje stan licznika w nieulotnej pamięci EEPROM oraz podaje wynik przy wykorzystaniu czterech diod DEL dostępnych w zestawie ZL4PIC. Przyciśnięcie przycisku S2 powinno zerować stan licznika. Podczas testowania należy kilka razy odłączyć i ponownie załączyć zasilanie i sprawdzić, czy stan licznika jest poprawnie zapamiętywany.
3. Układ, który mierzy przy wykorzystaniu przetwornika A/D napięcie na wejściu AN0 (w zestawie ZL4PIC możliwość połączenia z dzielnikiem napięcia zrealizowanym na potencjometrze R2). Wynik wyświetlany jest na wyświetlaczu alfanumerycznym LCD jako liczba w systemie dziesiętnym w jednostkach [mV] z rozdzielczością 1 mV. Dokładna kalibracja takiego woltomierza wykracza poza zakres ćwiczenia. Założyć, że minimalna wartość możliwa do odczytania z pary rejestrów ADRESH:ADRESL odpowiada napięciu 0V, natomiast maksymalna wartość 1111111111 (w systemie binarnym, przy wyrównaniu wyniku do prawej strony rejestrów) odpowiada napięciu +5V.
4. Układ, który mierzy przy wykorzystaniu przetwornika A/D napięcie na wejściu AN0 i sygnalizuje przy wykorzystaniu brzęczyka piezoelektrycznego stan logiczny odpowiadający temu napięciu w standardzie napięć dla układów TTL. Jeżeli napięcie odpowiada stanowi logicznemu niskiemu (0…0,8V), to brzęczyk powinien wydawać dźwięk o częstotliwości wyraźnie niższej niż w przypadku wykrycia stanu wysokiego (2,0…5,0V). Jeżeli napięcie nie odpowiada żadnemu określonemu stanowi logicznemu, to brzęczyk jest wyłączony.
5. Układ, który mierzy przy wykorzystaniu przetwornika A/D napięcie na wejściu AN0. Wynik wyświetlany jest na wyświetlaczu numerycznym LED jako liczba w systemie dziesiętnym w jednostkach [V] z rozdzielczością 0,01 V. Ponieważ pierwszy od lewej segment wyświetlacza LED wykorzystuje linię mikrokontrolera RA0/AN0, należy z niego zrezygnować alby możliwe było odczytanie poziomu napięcia z dzielnika zrealizowanego na potencjometrze R2, który można przyłączyć tylko do wejścia mikrokontrolera RA0/AN0.
6. Układ stopera, który wyświetla zmierzony czas na wyświetlaczu alfanumerycznym LCD z rozdzielczością 0,01 s. W celu uniknięcia problemów spowodowanych wielokrotnymi

załączeniami styków podczas jednego przyciśnięcia przycisku, zalecane jest rozdzielenie funkcji „start”, „stop” oraz „zerowanie” pomiędzy trzy różne przyciski.

7. Układ, który cyklicznie przewija w jednej linii wyświetlacza alfanumerycznego LED tekst dłuższy od liczby kolumn widocznych na wyświetlaczu, np.: „Instytut Fizyki PL, ul. Wolczanska 219”. Okres przewijania dostosować doświadczalnie tak, aby możliwy był wygodny odczyt. Definiowanie polskich liter dla wyświetlacza nie jest wymagane.
8. Układ, który symuluje sygnalizację dźwiękową w budziku. Sygnalizacja powinna być dwustopniowa: jeżeli początkowy łagodniejszy sygnał nie zostanie wyłączony np. w ciągu 30 sekund, to sygnał staje się bardziej natarczywy. W celu uniknięcia problemów spowodowanych drganiem styków, można przyjąć, że dwa różne przyciski odpowiadają za uruchomienie oraz wyłączenie sygnału. Zmianę natężenia dźwięku można uzyskać np. przez zmianę współczynnika wypełnienia impulsów sterujących brzęczykiem.

7. Wskazówki do raportu

Raport powinien zawierać:

1. Stronę tytułową (wg wzoru).
2. Sformułowanie celu ćwiczenia.
3. Wykaz użytej aparatury. W szczególności należy pamiętać o podaniu symbolu katalogowego wybranego mikrokontrolera.
4. Treść zadania, które powinien realizować zaprogramowany mikrokontroler w zestawie ZL4PIC.
5. Jeżeli treść zadania nie jest jednoznaczna, podać przyjęte dodatkowe założenia projektowe. Jeżeli wykorzystano jakieś przyciski, to należy podać przydzielone im funkcje.
6. Konfigurację wybranych zworek i przełączników w zestawie ZL4PIC o krytycznym znaczeniu dla działania programu.
7. Tekst źródłowy programu.
8. Dyskusję uzyskanych wyników. W szczególności rozważyć:
   a). Czy uzyskane wyniki są zgodne z założeniami. W przypadku wystąpienia rozbieżności opisać środki podjęte w celu ich usunięcia, znalezione błędy i uzyskany ostatecznie rezultat.
   b). Czy wystąpiły jakieś problemy w wykorzystywanym sprzęcie, oprogramowaniu lub w dokumentacji?
   c). Czy z perspektywy czasu można dostrzec jakieś możliwości uproszczenia lub poprawy algorytmu działania?

W raporcie ocenie podlegać będzie obecność, poprawność i jakość wszystkich wymienionych powyżej składników. Wstęp teoretyczny nie jest wymagany i w przypadku jego zamieszczenia w raporcie nie wpłynie na ocenę.

8. Literatura

[1] B.W. Kernighan, D.M. Ritchie, *Język C*, WNT, Warszawa 1988.

[2] *Getting Started with the HI-TECH C Compiler for PIC10/12/16 MCUs*, Microchip Technology Inc. 2010, plik PDF dostępny w zainstalowanym kompilatorze HI-TECH C® jako „Quckstart guide" w menu systemu Windows w grupie „HI-TECH Software".

[3] *HI-TECH C® for PIC10/12/16 User's Guide*, Data Sheet DS51865B, Microchip Technology Inc. 2010, plik PDF dostępny na stronie www.microchip.com oraz w zainstalowanym kompilatorze HI-TECH C® jako „User manual" w menu systemu Windows w grupie „HI-TECH Software".

[4] S. Pietraszek, *Mikroprocesory jednoukładowe PIC*, Helion, Gliwice 2002.

[5] T. Jabłoński, *Mikrokontrolery PIC16F8x w praktyce*, Wydawnictwo BTC, Warszawa 2002.

[6] T. Jabłoński, K. Pławsiuk, *Programowanie mikrokontrolerów PIC w języku C*, Wydawnictwo BTC, Warszawa 2005.

[7] *PICmicro™ Mid-Range MCU Family Reference Manual*, Data Sheet DS33023A, Microchip Technology Inc. 1997, dostępne na stronie www.microchip.com.

[8] *PIC16F84A*, Data Sheet DS35007C, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[9] *PIC16F818/819*, Data Sheet DS39598F, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[10] *PIC16F87XA*, Data Sheet DS39582C, Microchip Technology Inc. 2001-2013, dostępne na stronie www.microchip.com.

[11] *ZL4PIC Uniwersalny zestaw uruchomieniowy dla mikrokontrolerów PIC*, Wydawnictwo BTC 2005, dostępne na stronie www.fizyka.p.lodz.pl/pl/dla-studentow/tc/.

[12] Karta katalogowa wyświetlacza *WC1602A*, Wincom Tech. Co. Ltd., dostępna na stronie www.fizyka.p.lodz.pl/pl/dla-studentow/tc/.

Aneksy

A. Mapy pamięci wybranych mikrokontrolerów z rodziny PIC16

	Adres		Adres
INDF	00h	INDF	80h
TMR0	01h	OPTION_REG	81h
PCL	02h	PCL	82h
STATUS	03h	STATUS	83h
FSR	04h	FSR	84h
PORTA	05h	TRISA	85h
PORTB	06h	TRISB	86h
	07h		87h
EEDATA	08h	EECON1	88h
EEADR	09h	EECON2	89h
PCLATH	0Ah	PCLATH	8Ah
INTCON	0Bh	INTCON	8Bh
GPR Rejestry uniwersalne, 68 bajtów SRAM	0Ch 4Fh	Dostęp do SRAM w banku 0	8Ch CFh
	50h		D0h
	7Fh		FFh
Bank 0		Bank 1	

Oznaczenia:

Niezaimplementowane komórki pamięci, odczytywane jako 0.

Tabela A.1. Mapa pamięci mikrokontrolera PIC16F84A.

	Adres		Adres		Adres		Adres
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
	07h		87h		107h		187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Rezerwacja (1)	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Rezerwacja (1)	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
	18h		98h				
	19h		99h				
	1Ah		9Ah				
	1Bh		9Bh				
	1Ch		9Ch				
	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
GPR Rejestry uniwersalne, 96 bajtów SRAM	20h 7Fh	GPR Rejestry uniwersalne, 80 bajtów SRAM	A0h EFh	GPR Rejestry uniwersalne, 80 bajtów SRAM	120h 16Fh	Dostęp do 20h-7Fh	1A0h 1FFh
		Dostęp do 70h-7Fh	F0h FFh	Dostęp do 70h-7Fh	170h 17Fh		
Bank 0		Bank 1		Bank 2		Bank 3	

Oznaczenia:

Niezaimplementowane komórki pamięci, odczytywane jako 0.

(1): rejestry zarezerwowane; nie zapisywać.

Tabela A.2. Mapa pamięci mikrokontrolera PIC16F819.

	Adres		Adres		Adres		Adres
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD	08h	TRISD	88h		108h		188h
PORTE	09h	TRISE	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Rezerwacja (1)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Rezerwacja (1)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h	GPR Rejestry uniwersalne, 16 bajtów SRAM	116h	GPR Rejestry uniwersalne, 16 bajtów SAM	196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
GPR Rejestry uniwersalne, 96 bajtów SRAM	20h	GPR Rejestry uniwersalne, 80 bajtów SRAM	A0h	GPR Rejestry uniwersalne, 80 bajtów SRAM	120h	GPR Rejestry uniwersalne, 80 bajtów SRAM	1A0h
			EFh		16Fh		1EFh
		Dostęp do 70h-7Fh	F0h	Dostęp do 70h-7Fh	170h	Dostęp do 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Oznaczenia:

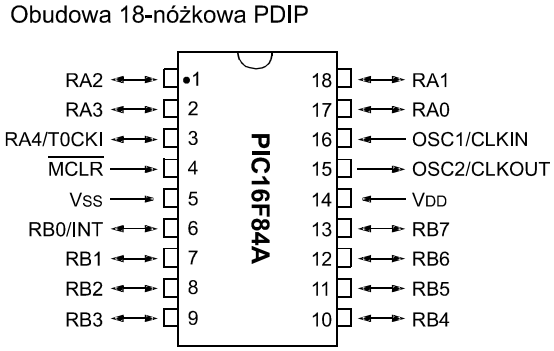
Niezaimplementowane komórki pamięci, odczytywane jako 0.

(1): rejestry zarezerwowane; nie zapisywać.

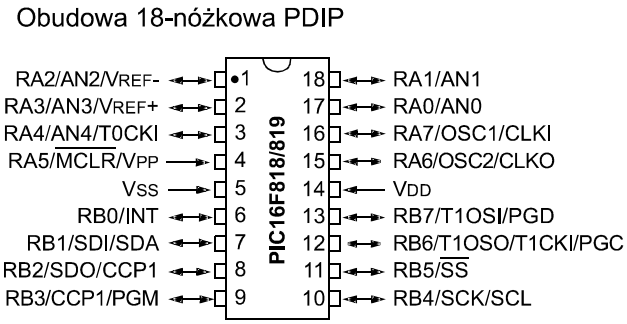
Tabela A.3. Mapa pamięci mikrokontrolera PIC16F877A.

B. Rozkład wyprowadzeń mikrokontrolerów

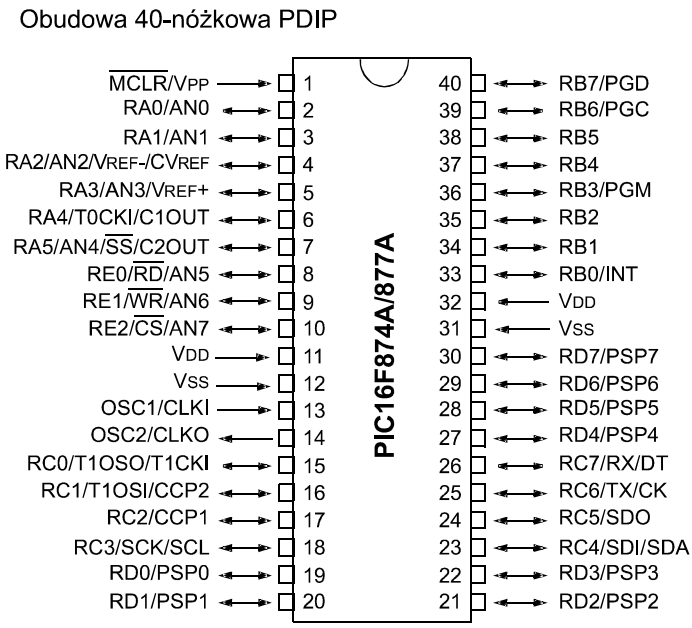
Mikrokontrolery PIC16 są produkowane w różnych obudowach, które nawet dla mikrokontrolerów tego samego typu różnią się liczbą i rozkładem wyprowadzeń. Mikrokontrolery dostępne w Laboratorium Elektroniki przeznaczone do pracy w zestawach uruchomieniowych ZL4PIC mają obudowy typu PDIP.



Rys. B.1. Rozkład wyprowadzeń mikrokontrolera PIC16F84A w obudowie PDIP.



Rys. B.2. Rozkład wyprowadzeń mikrokontrolerów PIC16F818 i 819 w obudowie PDIP.



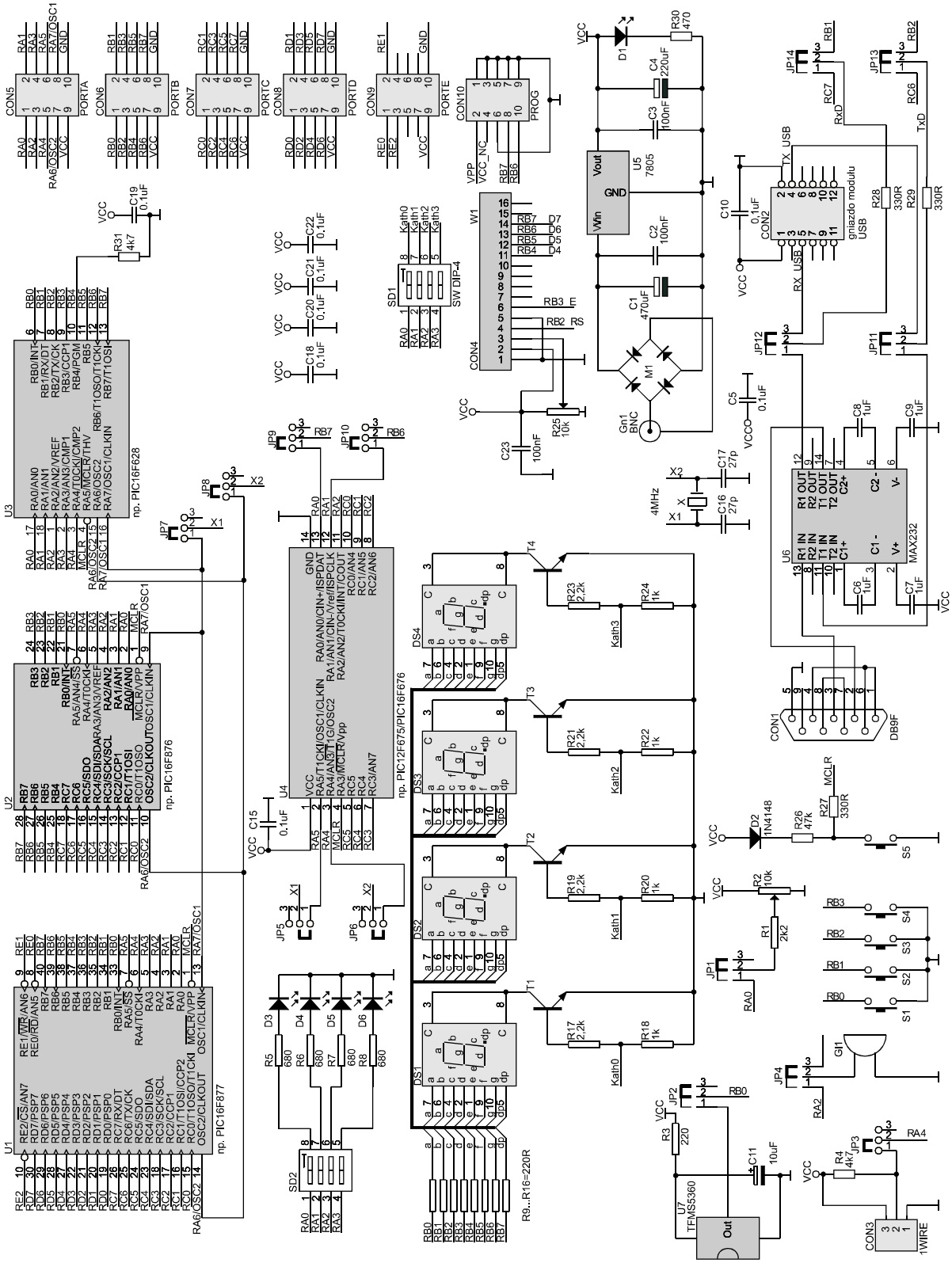
Rys. B.3. Rozkład wyprowadzeń mikrokontrolera PIC16F877A w obudowie PDIP.

Opis wybranych funkcji linii mikrokontrolerów:

AN0…AN7	–	wejścia analogowe,
CLKI(N)	–	wejście sygnału z zewnętrznego zegara,
CLKO(UT)	–	wyjście sygnału zegarowego z wewnętrznego oscylatora RC,
$\overline{\text{MCLR}}$	–	reset mikrokontrolera wyzwalany stanem niskim na tym wejściu,
PGM	–	linia zarezerwowana dla programatora gdy używane jest programowanie niskim napięciem,
RA0…RA7	–	linie portu A skonfigurowane jako cyfrowe wej./wyj. ogólnego przeznaczenia,
RB0…RB7	–	linie portu B skonfigurowane jako cyfrowe wej./wyj. ogólnego przeznaczenia,
RC0…RC7	–	linie portu C skonfigurowane jako cyfrowe wej./wyj. ogólnego przeznaczenia,
RD0…RD7	–	linie portu D skonfigurowane jako cyfrowe wej./wyj. ogólnego przeznaczenia,
RE0…RE2	–	linie portu E skonfigurowane jako cyfrowe wej./wyj. ogólnego przeznaczenia,
OSC1, OSC2	–	miejsca podłączenia rezonatora kwarcowego,
V_{DD}	–	dodatnie napięcie zasilania,
V_{PP}	–	miejsce przyłączenia napięcia +12...14 V z programatora w trybie wysokonapięciowym,
V_{SS}	–	masa.

C. Zestaw uruchomieniowy ZL4PIC

Zestaw uruchomieniowy ZL4PIC posiada odrębną instrukcję przygotowana przez producenta (BTC), która dostępna jest w dziale „Materiały pomocnicze” na stronie internetowej przedmiotu Technika Cyfrowa/zakładka „Laboratorium”. W niniejszej instrukcji ograniczmy się do podania schematu tego zestawu.



Rys. C.1. Schemat elektryczny zestawu ZL4PIC.