НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів та виробництв

Методичні вказівки для виконання лабораторної роботи №5 “Робота з модулем LCD на програмному симуляторі PIC Simulator IDE” з курсу “Проектування мікропроцесорних систем керування технологічними процесами”

Тернопіль 2017

Методичні вказівки для виконання лабораторної роботи №5 «Робота з модулем LCD на програмному симуляторі PIC Simulator IDE» з курсу «Проектування мікропроцесорних систем керування технологічними процесами».

Методичні вказівки розглянуті і схвалені кафедрою «Автоматизація технологічних процесів та виробництв», протокол № 4 від 21.11.2016 р.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Лабораторна робота №5
Робота з модулем LCD на програмному симуляторі PIC Simulator IDE

1. Робота з програмним симулятором PIC Simulator IDE

Запустивши на виконання PIC Simulator IDE, побачимо основне вікно цієї програми (рис. 1).

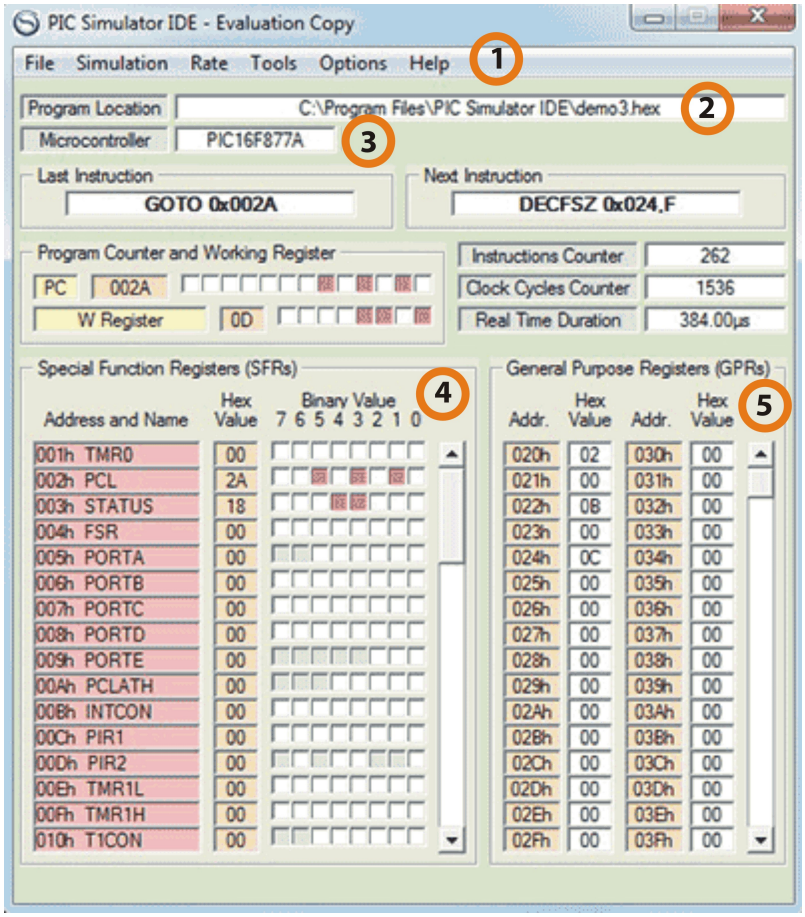

Рис. 1. Основне вікно програми PIC Simulator IDE

У верхній частині знаходяться різні меню, через які можна отримати доступ до різних основних і додаткових модулів програми (на рис. 1 позначено як «1»).

Далі, в рядку Program Location вказано шлях до обраної програми і її ім'я (на рис. 1 - «2»).

Нижче, в рядку Microcontrollers, відображається тип обраного мікроконтролера (на рис. 1 - «3»).

У нижній частині вікна є дві панелі (позначені як «4» і «5»). У них відображаються стан програми, вміст спеціальних і керуючих регістрів обраного МК.

Послідовність роботи з програмним симулятором наступний:

- запуск програми PIC Simulator IDE;
- вибір типу мікроконтролера, для якого написана програма;
- вибір частоти кварцового генератора (впливає тільки на відображувані програмою дані про час виконання програми або команди, але не на швидкість роботи програми, що налагоджуються в PIC Simulator IDE);
- завантаження програми у вигляді HEX-файлу або запуск вбудованого компілятора мови асемблер і написання в ньому потрібної програми;
- вибір потрібних модулів віртуальних пристроїв;
- вибір швидкості і режиму роботи програми симулятора;

● запуск процесу симуляції роботи програми на обраному МК.

Якщо потрібно скористатися для роботи з симулятором власною програмою або внести зміни у вже розроблену, необхідно створити або завантажити для цього файл асемблера, з якого після компіляції буде створений необхідний для роботи з симулятором hex-файл.

Для цього:

1. Натиснути Options | Assembler. Відкриється вікно компілятора Assembler – UNTITLED (рис. 2);

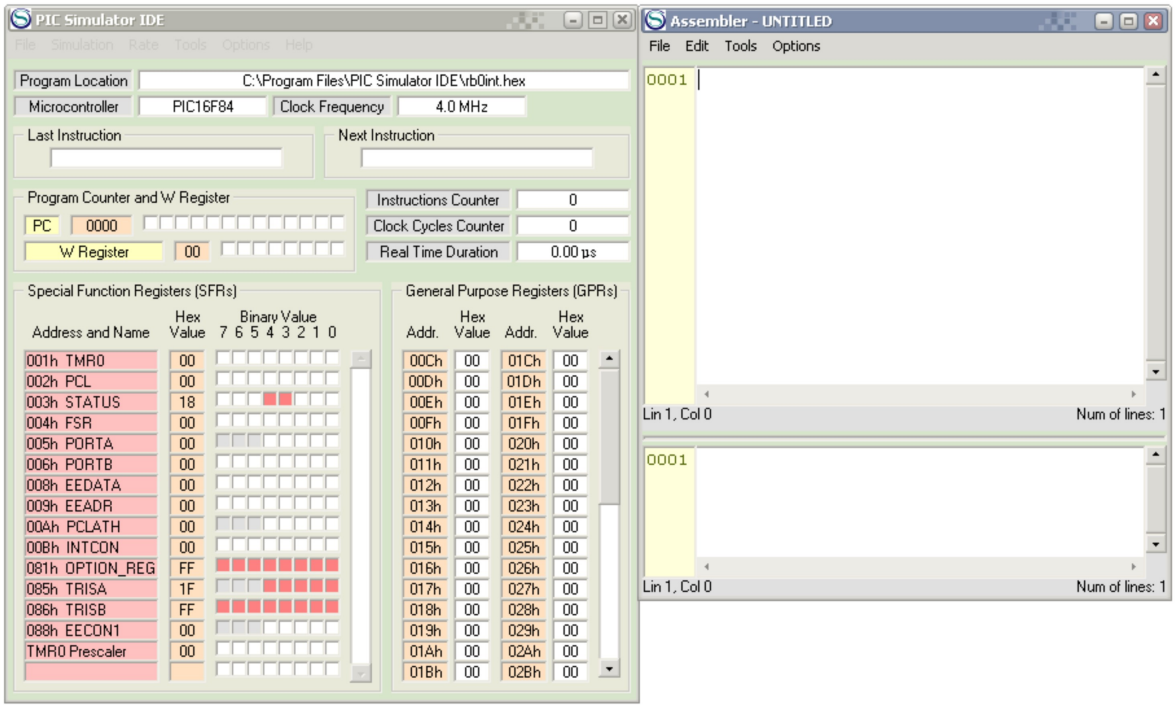

Рис. 2 Вікно симулятора з відкритим вікном Assembler

2. У вікні Assembler натисніть опцію File. Розкриється закладка (рис. 3), з якої для створення нового файлу потрібно натиснути New, а для завантаження вже створеного – OPEN.

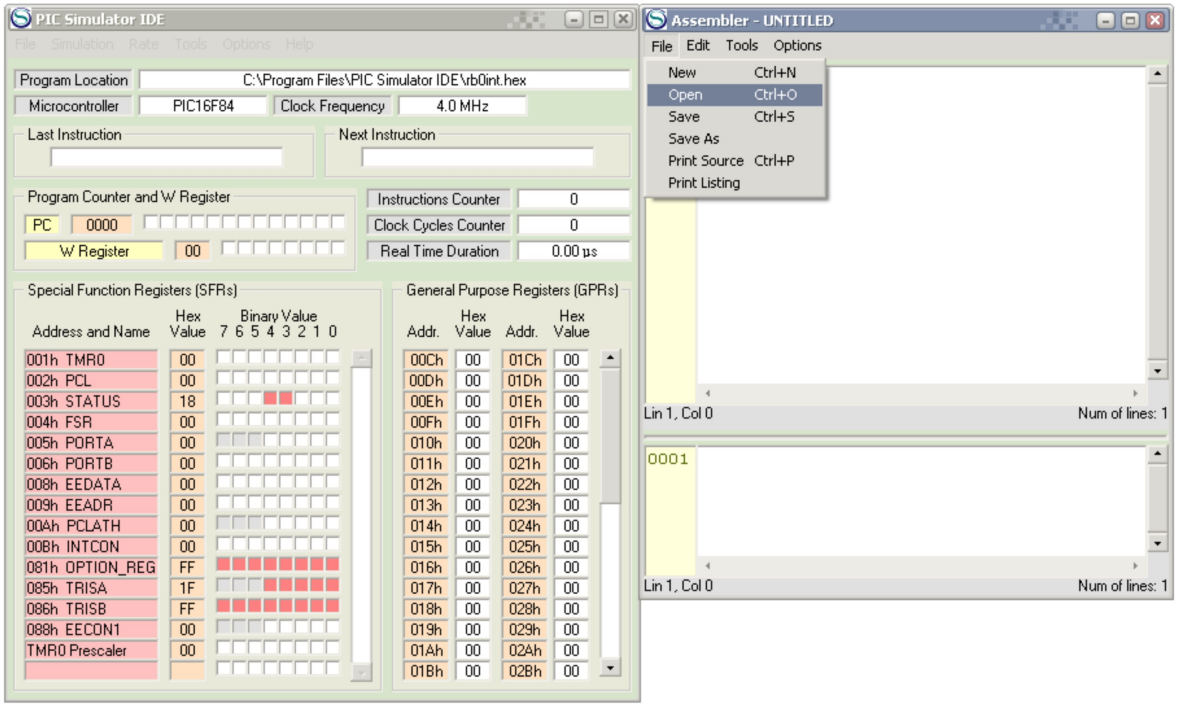

Рис. 3

3. Після вибору і завантаження файлу (наприклад. rb0int.asm), його текст з’явиться в верхній половині вікна Assembler (рис. 4).

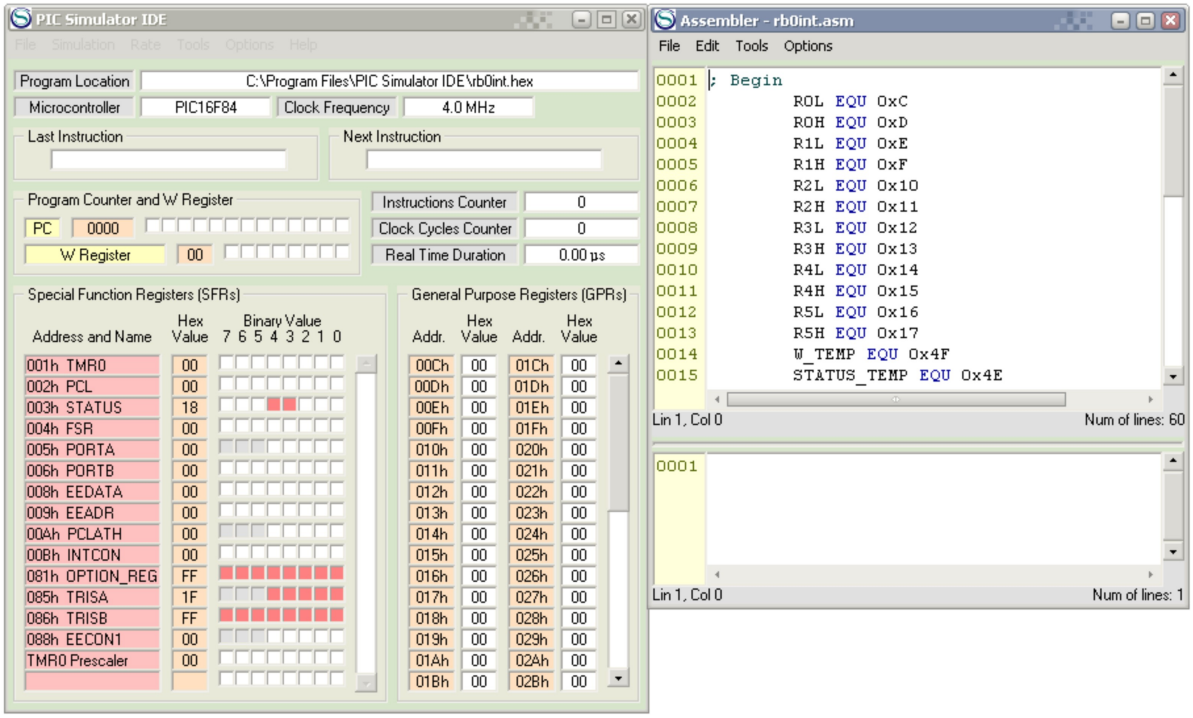

Рис. 4 Завантаження файлу rb0int.asm

4. Для компіляції створеного або завантаженого і потім зміненого файлу, натисніть Tools і у вікні, що розкриється – Assemle. В нижній половині вікна Assembler з’явиться відкомпільований файл і одночасно, при відсутності помилок, буде створений одноіменний hex-файл.

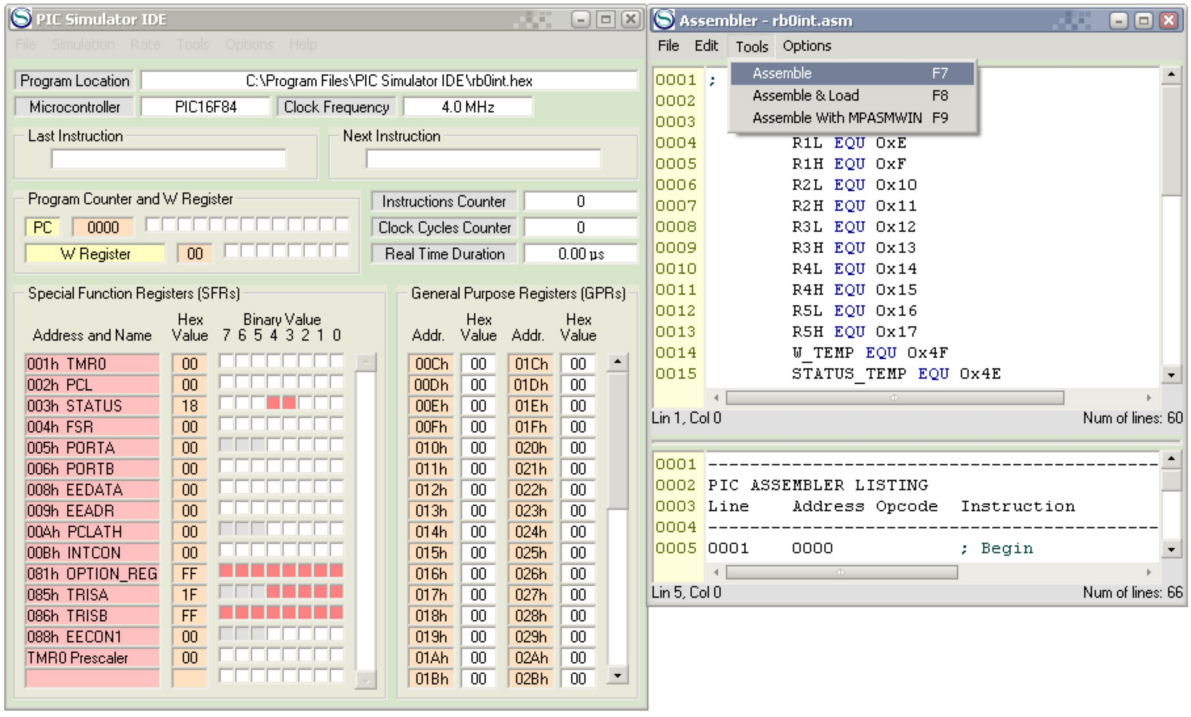

Рис. 5

2. **Завдання на лабораторну роботу:** робота мікроконтролера з модулем LCD.

1. Вивчити програмну модель PIC Simulator IDE.

2. Вивчити команди обміну даними через порти PIC – контролера.

3. Написати і дослідити роботу програми з Прикладу 1 та дослідити вміст регістрів контролера, які використовуються при виконанні цієї програми.

4. Користуючись вікном "PIC Disassembler" записати перші 7 команд виконуваної програми на асемблері та в шістнадцяткових кодах.

5. Записати для вибраних команд коментар щодо їх призначення (див. Приклад 2).

Приклад 1.

У програмі зчитується аналогове значення на аналоговому вході AN0, і отримані дані виводяться на ЖК- **екран (LCD 2 × 16).**

Текст програми з файлу lcd.asm має наступний вигляд:

```
; Begin
        R0L EQU 0x20
        R0H EQU 0x21
        R1L EQU 0x22
        R1H EQU 0x23
        R2L EQU 0x24
        R2H EQU 0x25
        R3L EQU 0x26
        R3H EQU 0x27
        R4L EQU 0x28
        R4H EQU 0x29
        R5L EQU 0x2A
        R5H EQU 0x2B
        ORG 0x0000
        BCF PCLATH,3
        BCF PCLATH,4
        GOTO L0002
        ORG 0x0004
        RETFIE
L0002:
; 1: Define ADC_CLOCK = 3 'default value is 3
; 2: Define ADC_SAMPLEUS = 10 'default value is 20
; 3: Define LCD_BITS = 8 'allowed values are 4 and 8 - the number of data interface lines
; 4: Define LCD_DREG = PORTB
; 5: Define LCD_DBIT = 0 '0 or 4 for 4-bit interface, ignored for 8-bit interface
; 6: Define LCD_RSREG = PORTD
; 7: Define LCD_RSBIT = 1
; 8: Define LCD_EREG = PORTD
; 9: Define LCD_EBIT = 3
; 10: Define LCD_RWREG = PORTD 'set to 0 if not used, 0 is default
; 11: Define LCD_RWBIT = 2 'set to 0 if not used, 0 is default
; 12: Define LCD_COMMANDUS = 2000 'delay after LCDCMDOUT, default value is 5000
; 13: Define LCD_DATAUS = 50 'delay after LCDOUT, default value is 100
; 14: Define LCD_INITMS = 2 'delay used by LCDINIT, default value is 100
; 15:  'the last three Define directives set the values suitable for simulation; they should be omitted for a real device
; 16:
; 17: Dim an0 As Word
;       The address of 'an0' is 0x2C
```

```
        an0 EQU 0x2C
; 18:
; 19: TRISA = 0xff 'set all PORTA pins as inputs
        BSF STATUS,RP0
        MOVLW 0xFF
        MOVWF 0x05
        BCF STATUS,RP0
; 20: ADCON1 = 0 'set all PORTA pins as analog inputs
        BSF STATUS,RP0
        CLRF 0x1F
        BCF STATUS,RP0
; 21: Lcdinit 1 'initialize LCD module; cursor is blinking
        BCF 0x08,3
        BCF 0x08,1
        BCF 0x08,2
        BSF STATUS,RP0
        BCF 0x08,3
        BCF 0x08,1
        BCF 0x08,2
        CLRF 0x06
        BCF STATUS,RP0
        MOVLW 0x02
        MOVWF R0L
        MOVLW 0x00
        MOVWF R0H
        CALL W001
        MOVLW 0x33
        CALL LC02
        MOVLW 0x33
        CALL LC02
        MOVLW 0x33
        CALL LC02
        MOVLW 0x38
        CALL LC02
        MOVLW 0x0D
        CALL LC02
        MOVLW 0x01
        CALL LC02
; 22:
; 23: loop:
L0001:
; 24: Adcin 0, an0
        BSF STATUS,RP0
        BSF ADCON1,ADFM
        MOVLW 0x00
        BCF STATUS,RP0
        MOVWF R0L
        CALL A001
        BSF STATUS,RP0
        MOVF ADRESL,W
        BCF STATUS,RP0
        MOVWF 0x2C
```

```
        MOVF ADRESH,W
        MOVWF 0x2D
; 25: Lcdcmdout LcdClear 'clear LCD display
        MOVLW 0x01
        CALL LC02
; 26: Lcdout "Analog input AN0" 'text for the line 1
        MOVLW 0x41
        CALL LC01
        MOVLW 0x6E
        CALL LC01
        MOVLW 0x61
        CALL LC01
        MOVLW 0x6C
        CALL LC01
        MOVLW 0x6F
        CALL LC01
        MOVLW 0x67
        CALL LC01
        MOVLW 0x20
        CALL LC01
        MOVLW 0x69
        CALL LC01
        MOVLW 0x6E
        CALL LC01
        MOVLW 0x70
        CALL LC01
        MOVLW 0x75
        CALL LC01
        MOVLW 0x74
        CALL LC01
        MOVLW 0x20
        CALL LC01
        MOVLW 0x41
        CALL LC01
        MOVLW 0x4E
        CALL LC01
        MOVLW 0x30
        CALL LC01
; 27: Lcdcmdout LcdLine2Home 'set cursor at the beginning of line 2
        MOVLW 0xC0
        CALL LC02
; 28: Lcdout "Value: ", #an0 'formatted text for line 2
        MOVLW 0x56
        CALL LC01
        MOVLW 0x61
        CALL LC01
        MOVLW 0x6C
        CALL LC01
        MOVLW 0x75
        CALL LC01
        MOVLW 0x65
        CALL LC01
```

```
        MOVLW 0x3A
        CALL LC01
        MOVLW 0x20
        CALL LC01
        MOVF 0x2C,W
        MOVWF R2L
        MOVF 0x2D,W
        MOVWF R2H
        CALL LC21
; 29: WaitMs 1 'larger value should be used in real device
        MOVLW 0x01
        MOVWF R0L
        CLRF R0H
        CALL W001
; 30: Goto loop 'loop forever
        GOTO L0001
; End of program
L0003:GOTO L0003
; Division Routine
D001:  MOVLW 0x10
        MOVWF R3L
        CLRF R2H
        CLRF R2L
D002:  RLF R0H,W
        RLF R2L,F
        RLF R2H,F
        MOVF R1L,W
        SUBWF R2L,F
        MOVF R1H,W
        BTFSS STATUS,C
        INCFSZ R1H,W
        SUBWF R2H,F
        BTFSC STATUS,C
        GOTO D003
        MOVF R1L,W
        ADDWF R2L,F
        MOVF R1H,W
        BTFSC STATUS,C
        INCFSZ R1H,W
        ADDWF R2H,F
        BCF STATUS,C
D003:  RLF R0L,F
        RLF R0H,F
        DECFSZ R3L,F
        GOTO D002
        MOVF R0L,W
        RETURN
; Waitms Routine
W001: MOVF R0L,F
        BTFSC STATUS,Z
        GOTO W002
        CALL W003
```

```
        DECF R0L,F
        NOP
        NOP
        NOP
        NOP
        NOP
        GOTO W001
W002:   MOVF R0H,F
        BTFSC STATUS,Z
        RETURN
        CALL W003
        DECF R0H,F
        DECF R0L,F
        GOTO W001
W003:   MOVLW 0x0C
        MOVWF R2H
W004:   DECFSZ R2H,F
        GOTO W004
        NOP
        NOP
        MOVLW 0x12
        MOVWF R1L
W005:   DECFSZ R1L,F
        GOTO W006
        CALL W007
        CALL W007
        NOP
        NOP
        RETURN
W006:   CALL W007
        GOTO W005
W007:   MOVLW 0x0D
        MOVWF R2L
W008:   DECFSZ R2L,F
        GOTO W008
        NOP
        RETURN
; Waitus Routine - Byte Argument
X001:   MOVLW 0x0A
        SUBWF R4L,F
        BTFSS STATUS,C
        RETURN
        GOTO X002
X002:   MOVLW 0x06
        SUBWF R4L,F
        BTFSS STATUS,C
        RETURN
        GOTO X002
; Waitus Routine - Word Argument
Y001:   MOVLW 0x10
        SUBWF R4L,F
        CLRW
```

```
        BTFSS STATUS,C
        ADDLW 0x01
        SUBWF R4H,F
        BTFSS STATUS,C
        RETURN
        GOTO Y002
Y002:   MOVLW 0x0A
        SUBWF R4L,F
        CLRW
        BTFSS STATUS,C
        ADDLW 0x01
        SUBWF R4H,F
        BTFSS STATUS,C
        RETURN
        GOTO Y002
; Adcin Routine
A001:   RLF R0L,F
        RLF R0L,F
        RLF R0L,F
        MOVLW 0x38
        ANDWF R0L,F
        MOVLW 0xC1
        IORWF R0L,W
        MOVWF ADCON0
        MOVLW 0x0A
        MOVWF R4L
        CALL X001
        BSF ADCON0,GO
A002:   BTFSC ADCON0,GO
        GOTO A002
        BCF PIR1,ADIF
        BCF ADCON0,ADON
        RETURN
; Lcdout Routine
LC01:   BSF 0x08,1
        BCF 0x08,2
        MOVWF 0x06
        BSF 0x08,3
        NOP
        BCF 0x08,3
        MOVLW 0x32
        MOVWF R4L
        CALL X001
        RETURN
; Lcdcmdout Routine
LC02:   BCF 0x08,1
        BCF 0x08,2
        MOVWF 0x06
        BSF 0x08,3
        NOP
        BCF 0x08,3
        MOVLW 0xD0
```

```
        MOVWF R4L
        MOVLW 0x07
        MOVWF R4H
        CALL Y001
        RETURN
; Lcdout Decimal Conversion Routine
LC21:  BSF R3H,7
        MOVLW 0x27
        MOVWF R1H
        MOVLW 0x10
        CALL LC22
        MOVLW 0x03
        MOVWF R1H
        MOVLW 0xE8
        CALL LC22
        CLRF R1H
        MOVLW 0x64
        CALL LC22
        CLRF R1H
        MOVLW 0x0A
        CALL LC22
        MOVF R2L,W
        GOTO LC23
LC22:  MOVWF R1L
        MOVF R2H,W
        MOVWF R0H
        MOVF R2L,W
        MOVWF R0L
        CALL D001
        MOVF R0L,W
        BTFSS STATUS,Z
        BCF R3H,7
        BTFSC R3H,7
        RETURN
LC23:  ADDLW 0x30
        CALL LC01
        RETURN
; End of listing
        END
```

3. Послідовність роботи з симулятором при виконанні програми

Переглянемо результати роботи цієї програми в PIC Simulator IDE.

Для цього виконаємо наступне, взявши модель **МК PIC16F877**:

1. Запустити PIC Simulator IDE;
2. Натиснути Options | Select Microcontroller;
3. Вибрати **PIC16F877** і натиснути кнопку Select;
4. Натиснути File | Load Program;
1. Вибрати файл lcd.hex і натиснути кнопку Open;
2. Натиснути Tools | LCD (відкриється вікно Module LCD);
3. Натиснути кнопку Setup у вікні Module LCD;
4. Натиснути Data Lines і встановити PORTB;
5. Натиснути поле Interface і встановити 8 біт;

6. Натиснути поле RS Line і встановити PORTD, 1;
7. Натиснути поле E Line і встановити PORTD, 3;
8. Натиснути поле R | W Line і встановити PORTD, 2;
9. Натиснути Apply! (Закриється вікно установки LCD interface);
10. Вибрати Rate | Extremely Fast simulation rate;
11. Натиснути кнопку «A» перед виводом RA0/AN0. З'явиться білий прямокутник в середній частині панелі «Microcontroller View» з «панеллю прокрутки», повзунком якої можна змінювати аналогове значення напруги на цьому виводі. Це значення відображається у вікні під написом «AN0» та одночасно перед виводом «AN0/RA0».
12. Натиснути кнопку «OK» на полі прокрутки і подивитися, як зміниться стан виводів порту PORTB та інформація на LCD – моніторі.
12. Натиснути Simulation | Start (почнеться виконання програми);

Цифровий код, що відповідає цьому значенню, відображується на лініях порту RB0… RB7 («ON» відповідає логічній «1», «OFF»- логічному «0») і одночасно через деякий час на LCD - індикаторі;

Вид екрану з виконуваною програмою показано на рис. 6.

.

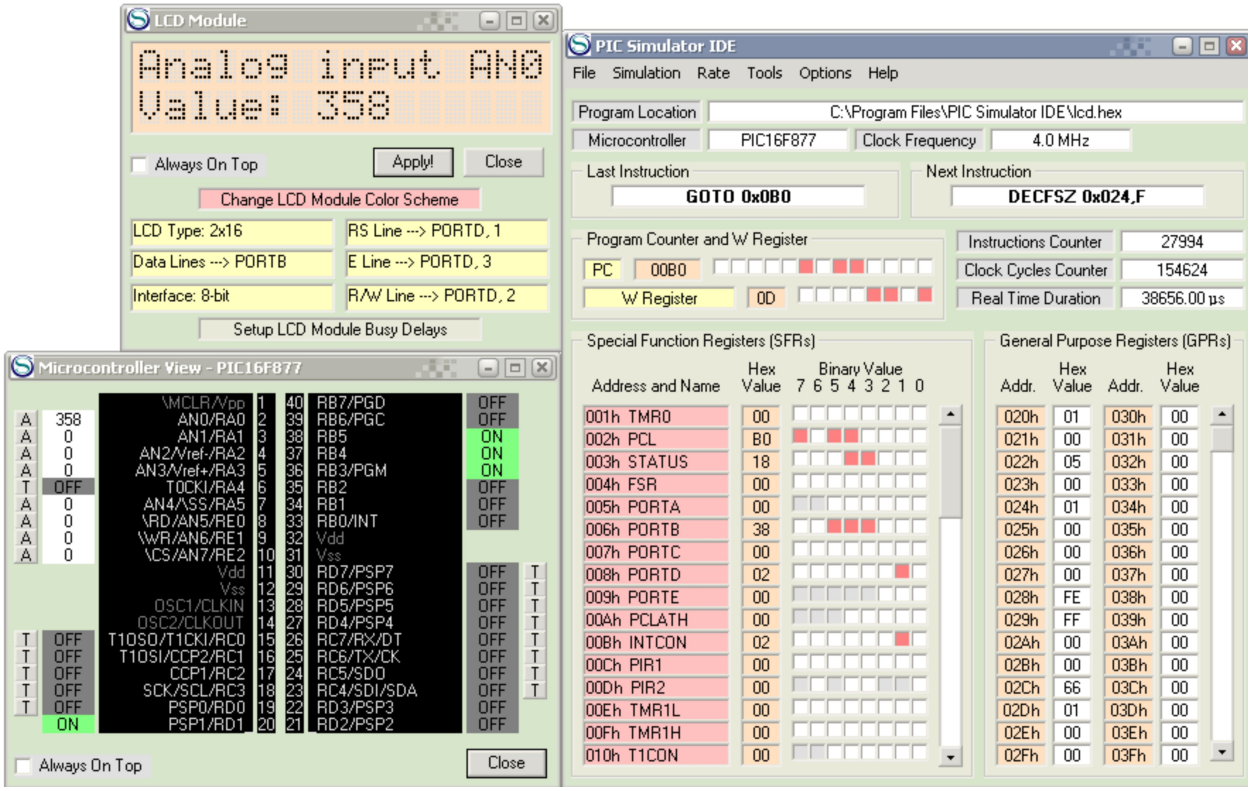

Рис. 6 Вигляд екрану з виконуваною програмою «Робота з модулем LCD»

Останні три кроки можна повторити кілька разів і подивитися, як змінюються значення логічних рівнів на виводах порту RB0…RB7.

З вікна "PIC Disassembler" (рис. 7) вибираємо сім перших команд з їх шістнадцятковими кодами і знаходимо з таблиці кодів асемблера PIC-контролера коментар щодо призначення цих команд (див. Приклад 2, де наведено такий запис для однієї команди).

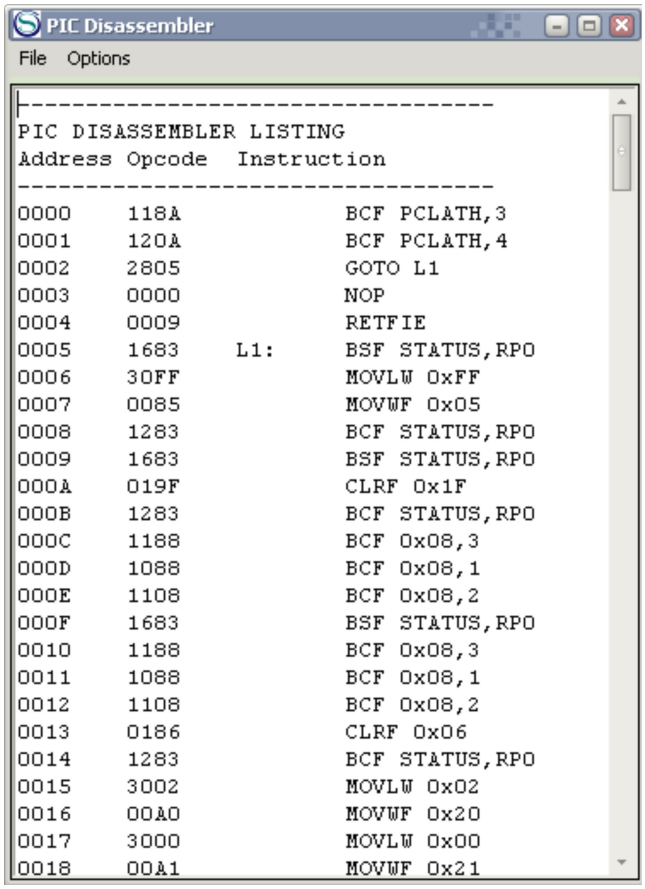

Рис. 7 Фрагмент виконуваної програми у вікні PIC Disassembler

Приклад 2.

Код команди	**Команда**	**Виконувана операція (коментар)**
118A	BCF PCLATH, 3	; скинути в “0” 3-ій біт регістра PCLATH

і т.д.

Вміст регістрів контролера, які використовуються при виконанні програми, знаходимо з області регістрів Adress and Name, яка розташована в лівій нижній частині основного вікна симулятора (виділені рожевим кольором). Всі регістри возьмирозрядні.

В процесі виконання програми по зміні кольору комірок видно, вміст яких регістрів змінюється. Забарвлення комірки відповідного розряду регістра помаранчевим кольором означає наявність “1”, білим - “0”.

Вміст регістрів записуємо в шістнадцятковому коді за Прикладом 3.

Приклад 3.

Регістр	**Вміст регістра**
PORTB	6C

і т. д.

4. Контрольні запитання

1. Послідовність роботи АЦП мікроконтролера.
2. Аналогові та цифрові виводи мікроконтролера.
3. Формат та призначення регістрів PC, W, PORTA…PORTE.
4. Призначення та позначення основних елементів програмної моделі мікроконтролера

5. Література

1. Данилин А. Программа-симулятор PIC Simulator IDE / Данилин А. // Современная электроника. 2006.- №4. -С. 68-76.

2. Таверньe К. PIC-микроконтроллеры. Практика применения. М.: ДМК, 2002.

3. Предко М. Создайте робота своими руками на PIC- контроллере./ Майкл Предко; Пер. с английского Земского Ю.В. – М.: ДМК Пресс, 2006. – 408 с.: ил. – (В помощь радиолюбителю).

4. Кениг А. и М. Полное руководство по PIC-микроконтроллерам.: Пер. с нем.-К.: “МК-Пресс”, 2007.-256 с., ил.