Программа-симулятор PIC Simulator IDE

Александр Данилин (Брянская обл.)

PIC Simulator IDE – это отладчик программ, написанных для микроконтроллеров microPIC серии 12 и 16 компании Microchip. Программа позволяет разработчику непосредственно в отлаживаемой программе работать с виртуальной периферией, а процесс отладки и написания программы выполнить непосредственно в PIC Simulator IDE. Автор приводит примеры работы с программой PIC Simulator IDE, проработав которые, даже начинающие разработчики смогут самостоятельно тестировать свои программы.

Введение

Автором программы PIC Simulator IDE является Владимир Coco (компания OshonSoft). Загрузить её можно из Интернета [1]. PIC Simulator IDE – это отладчик программ, написанных для микроконтроллеров (МК) microPIC серии 12 и 16 компании Microchip. Эта программа также позволяет разработчику непосредственно в отлаживаемом коде работать с виртуальной периферией. Имеется в виду, что специальные программные модули PIC Simulator IDE имитируют работу различных реальных электронных устройств: символьного ЖК-экрана, 7-сегментных индикаторов, терминала связи, 4-канального генератора, осциллографа и другого оборудования. Практически весь процесс отладки и написания программы можно выполнить непосредственно в PIC Simulator IDE, т.е. нет необходимости, внеся в программу какие-либо изменения, программировать МК и подключать его в отладочную плату с периферийными устройствами. Благодаря этому PIC Simulator IDE ускоряет процесс написания и отладки программ для МК.

Рис. 1. Основное окно программы PIC Simulator IDE

PIC Simulator IDE имеет встроенный ассемблер и компилятор Basic-подобного языка, что также упрощает процесс написания программ для МК и уменьшает время разработки изделий в целом. В PIC Simulator IDE имеется большое количество всевозможных настроек и режимов работы, что позволяет пользователю самостоятельно выбрать оптимальный режим отладки программы для МК.

Все эти возможности программы PIC Simulator IDE делают её отличным инструментом как для начинающих разработчиков, так и для опытных программистов.

Установка и работа с программой

Процесс установки программы прост и не требует особых знаний и навыков. Распаковав архив с программой, находим файл setup.exe и запускаем его. На появляющиеся вопросы необходимо только нажимать кнопки OK, Next или Continue. После завершения установки PIC Simulator IDE появляется в меню «Программы». Рекомендуется вывести ярлык этой программы на рабочий стол.

Запустив PIC Simulator IDE, мы увидим основное окно этой программы (рис. 1). В верхней части находятся различные меню, через которые мы получим доступ к различным настройкам и дополнительным модулям программы (на рис. 1 помечено как «1»). Далее, в строке Program Location указан путь в выбранной вами программе и её имя (на рис. 1 – «2»). Ниже, в строке Microcontrollers, отображается тип выбранного микроконтроллера (на рис. 1 – «3»). В нижней части окна имеются две панели (помечены как «4» и «5»). В них отображаются состояния специальных и управляющих регистров выбранного МК (в данный момент нет необходимости описывать расшифровку их содержимого, а в дальнейшем рассмотрим, как можно использовать эти данные и изменять ход выполнения загруженной в PIC Simulator IDE программы).

Порядок работы с программой-симулятором следующий:

- запуск программы PIC Simulator IDE;
- выбор типа процессора, для которого написана программа;
- выбор частоты кварцевого генератора (влияет только на отображаемые программой данные о времени выполнения программы или команды, но не на скорость работы программы, отлаживаемой в PIC Simulator IDE);
- загрузка программы в виде HEX-файла или запуск встроенного компилятора языка Basic и написание в нём нужной программы;
- выбор нужных модулей виртуальных устройств;
- выбор скорости и режима работы программы-симулятора;
- запуск процесса симуляции работы программы на выбранном МК;
- остановка работы программы PIC Simulator IDE.

Примеры работы с PIC Simulator IDE

В комплект поставки программы-симулятора входит несколько примеров работы с программой. Все примеры написаны на Basic, компилятор которого встроен в программу-симулятор.

Все приведённые в описании примеры программ расположены в папке, в которую установлена программа PIC Simulator IDE. По умолчанию эта папка расположена по адресу: C:\Program Files\PIC Simulator IDE [2]. Кроме файлов программ на Basic, в этой папке хранятся уже откомпилированные файлы на Ассемблере и файлы дампов памяти, подготовленные для непосредственной загрузки в МК. Файлы на Ассемблере с расширением .asm были сгенерированы встроенным компилятором Basic. Файлы дампов памяти с расширением .hex были сгенерированы встроенным ассемблером.

Ниже приведены 11 примеров работы с программой-симулятором. Начиная с примера 5, демонстрируется работа компилятора, ассемблера и отладчика. Все примеры написаны на Basic и имеют подробные комментарии. Во всех примерах остановить работу программы PIC Simulator IDE можно, нажав на Simulation|Stop. Изучив эти примеры, можно приступать к самостоятельной работе с PIC Simulator IDE и писать программы на Basic.

Пример 1: работа с Timer0, обработка прерывания TMR0

Эта программа использует прерывание Timer0, чтобы периодически изменять значение на выводах порта PORTB, т.е. при переполнении таймера происходит переход на подпрограмму обработки прерывания. Считывается значение из порта PORTB, и оно уменьшается на единицу. Это значение заносится обратно в порт PORTB. Все эти изменения состояния на выводах порта PORTB отображаются в PIC Simulator IDE. Текст программы из файла timer0.bas имеет следующий вид:

```
TRISB = 0x00
'настройка порта PORTB на вывод
данных
PORTB = %11111111
'на все выводы порта PORTB -
высокий уровень "1"
INTCON.T0IE = 1
'разрешает прерывание Timer0
INTCON.GIE = True
'включает все прерывания
OPTION_REG.T0CS = False
'устанавливает Timer0 и переключа-
ет его на внутренний генератор МК
End

On Interrupt
'подпрограмма обработки
прерывания
PORTB = PORTB - 1
'уменьшаем значение порта PORTB
INTCON.T0IF = 0
'включает заново прерывание TMR0
Resume
'выход из подпрограммы обработки
прерывания
```

Запустим эту программу в PIC Simulator IDE следующим образом:

1. Запустить PIC Simulator IDE;
2. Нажать Options|Select Microcontroller;
3. Выбрать PIC16F84 и нажать кнопку Select;

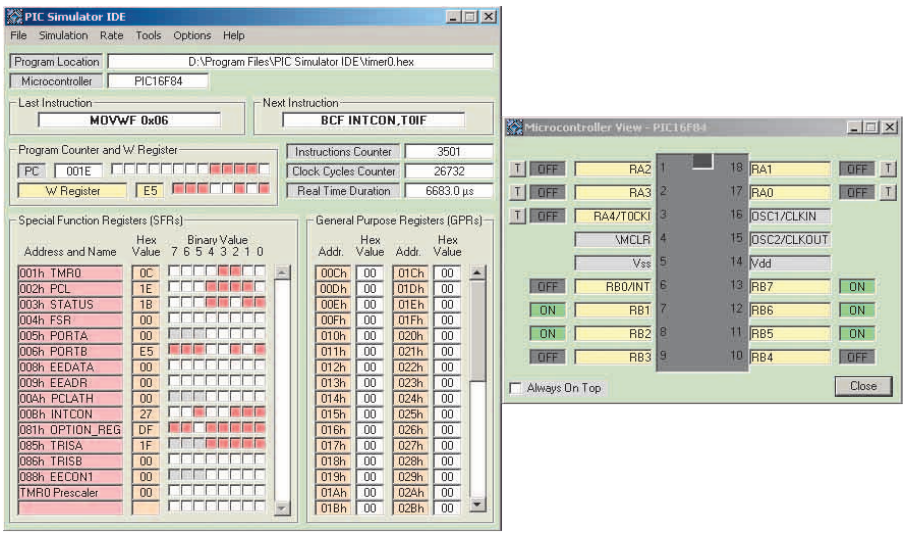

Рис. 2. Вид экрана с выполняющейся программой «Обработка прерываний Timer0 и TMR0»

4. Нажать File|Load Program;
5. Выбрать файл timer0.hex и нажать кнопку Open (программа загружена);
6. Нажать Tools|Microcontroller View (откроется окно Microcontroller View);
7. Выбрать Rate|Extremely Fast simulation rate;
8. Нажать Simulation|Start (начнётся выполнение загруженной программы).

Вид экрана с выполняющейся программой показан на рис. 2.

Пример 2: обработка внешних прерываний на примере RB0/INT

Эта программа изменяет значение порта PORTA при изменении состояния входа RB0/INT по фронту импульса. Текст программы из файла rb0int.bas имеет следующий вид:

```
TRISA = 0x00
'настройка порта PORTA на вывод
данных
PORTA = 0xff
'на все выводы порта PORTA -
высокий уровень "1"
INTCON.INTE = 1
'разрешает прерывание RB0/INT
INTCON.GIE = 1
'включает все прерывания
End

On Interrupt
'подпрограмма обработки прерывания
PORTA = PORTA - 1
'уменьшает значение порта PORTA
INTCON.INTF = 0
'включает заново прерывание
RB0/INT
Resume
'выход из подпрограммы обработки
прерывания
```

Запустим эту программу в PIC Simulator IDE, выполнив первые четыре действия из примера 1 и далее продолжив:

1. Выбрать файл rb0int.hex и нажать кнопку Open (программа загружена);
2. Нажать Tools|Microcontroller View (откроется окно Microcontroller View);
3. Выбрать Rate|Extremely Fast simulation rate;
4. Нажать Simulation|Start (начнётся выполнение загруженной программы).

Вид экрана с выполняющейся программой показан на рис. 3 (нажатие кнопки «T», связанной с ножкой RB0/INT, переключит логическое состояние этого вывода).

Пример 3: работа с EEPROM

Этот пример заполняет всю EEPROM разными значениями и вводит МК в бесконечный цикл. Текст программы из файла eeprom.bas имеет следующий вид:

```
Dim a As Byte
'адрес ячейки в EEPROM МК
Dim b As Byte
'данные, которые будут записаны
в EEPROM (у МК PIC16F84 64 байта
памяти EEPROM)
For a = 0 To 63
'цикл организован для всей
памяти EEPROM
b = 255 - a
'получим значение переменной для
записи в память
Write a, b
'запишем значение переменной "a"
в ячейку "b"
Next a
```

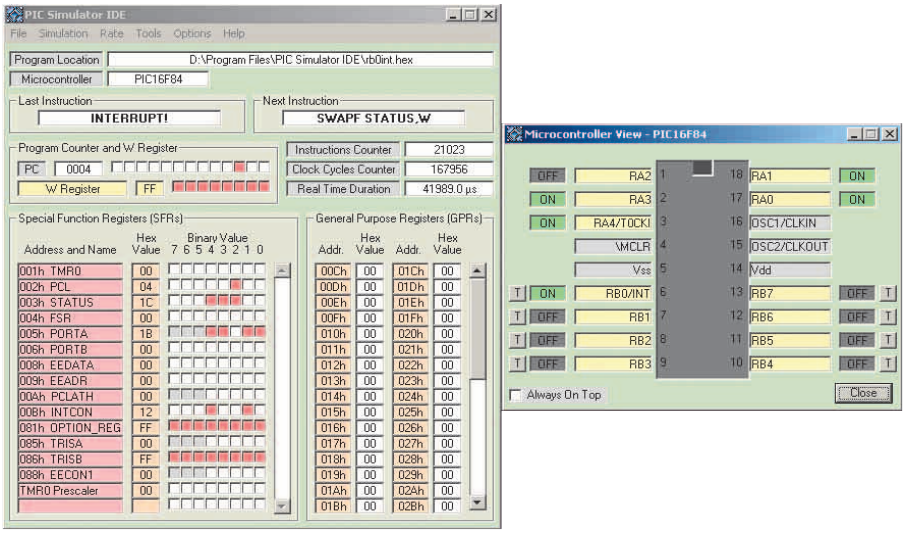

Рис. 3. Вид экрана с выполняющейся программой «Обработка внешних прерываний на примере RB0/INT»

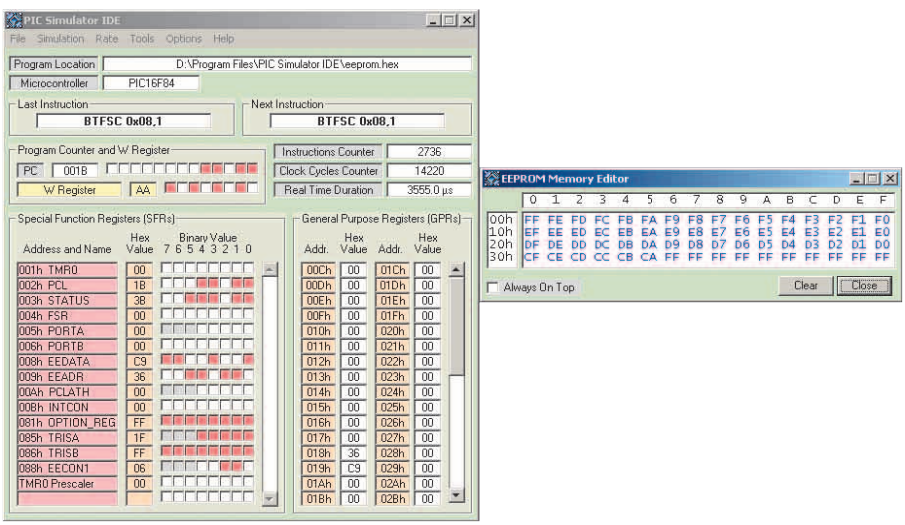

Рис. 4. Вид экрана с выполняющейся программой «Работа с EEPROM»

```
'выберем следующую ячейку памяти
EEPROM
```

Запустим программу в PIC Simulator IDE, выполнив первые четыре действия из примера 1 и далее продолжив:

1. Выбрать файл eeprom.hex и нажать Open (программа загружена);
2. Нажать Tools|EEPROM (редактор EEPROM; откроется окно Memory Editor EEPROM);
3. Нажать Options|Change (время записи EEPROM);
4. Ввести новое значение 100 и нажать OK (надо быть внимательнее при выборе этого значения, потому что действительное значение – примерно 20 000 тактовых циклов при частоте тактового генератора 4 МГц; в этом примере мы используем уменьшенное значение – это значительно ускорит процесс моделирования, но не нарушит работу программы);
5. Выбрать Rate|Extremely Fast simulation rate;
6. Нажать Options|Infinite Loop Stops Simulation;
7. Нажать Simulation|Start (начнется моделирование);

Программа заполнит память EEPROM данными и введёт МК в бесконечный цикл, после обнаружения которого PIC Simulator IDE автоматически остановит работу. Вид экрана с выполняющейся программой показан на рис. 4.

Пример 4: математические операции: подпрограмма умножения

Демонстрация работы компилятора, ассемблера и отладчика. В этом примере умножаются два числа: 123 (шестнадцатеричный 7B) и 234 (шестнадцатеричный EA), получим результат 28782 (шестнадцатеричный 706E). Текст программы имеет следующий вид:

```
Dim a As Word
'переменная для хранения первого
числа
Dim b As Word
'переменная для хранения второго
числа
Dim x As Word
'переменная для хранения резуль-
тата
a = 123
'установим первое значение
b = 234
'установим второе значение
x = a * b
'вычислим результат и поместим
его в переменную x
```

Запустим эту программу в PIC Simulator IDE, выполнив первые четыре действия из примера 1 и далее продолжив:

1. Выбрать файл multiply.bas и нажать кнопку Open (эта программа будет отображена в редакторе);
2. Нажать Tools|Compile (компилятор генерирует файл multiply.asm с исходным текстом на ассемблере);
3. Закрыть окно BASIC Compiler;
4. Нажать Tools|Assembler;
5. Нажать File|Open;
6. Выбрать файл multiply.asm и нажать кнопку Open (программа на Ассемблере отобразится в редакторе);
7. Нажать Tools\Assemble (после того как операция закончится, ассемблер сгенерирует два файла: multiply.lst и multiply.hex; выходной файл multiply.lst отобразится в редакторе);
8. Закрыть окно Assembler;
9. Нажать File|Load Program;
10. Выбрать файл multiply.hex и нажать кнопку Open;
11. Нажать Tools|Breakpoints Manager (откроется окно Breakpoints Manager);
12. Нажать «Да», чтобы использовать существующий ассемблер, выдавший файл;
13. Нажать строку, соответствующую адресу 0018, чтобы определить контрольную точку на этой команде;
14. Выбрать Hold PC In Focus option;
15. Выбрать Rate|Extremely Fast simulation rate;
16. Нажать Simulation|Start (начнётся моделирование).

Когда эта математическая подпрограмма закончится, программа вхо-

дит в бесконечный цикл в адресе 0018, но из-за контрольной точки PIC Simulator IDE автоматически переключится в режим моделирования Step by step. Работу PIC Simulator IDE можно остановить, нажав на Simulation|Stop, или продолжить её выполнение, очистив контрольную точку и нажав на Rate|Extremely.

Регистры 19-й и 18-й будут содержать первый параметр: 007B. Регистры 1BH и 1AH будут содержать второй параметр: 00EA. Результат 706E находится в регистрах 1DH и 1CH. Вид экрана с выполняющейся программой показан на рис. 5.

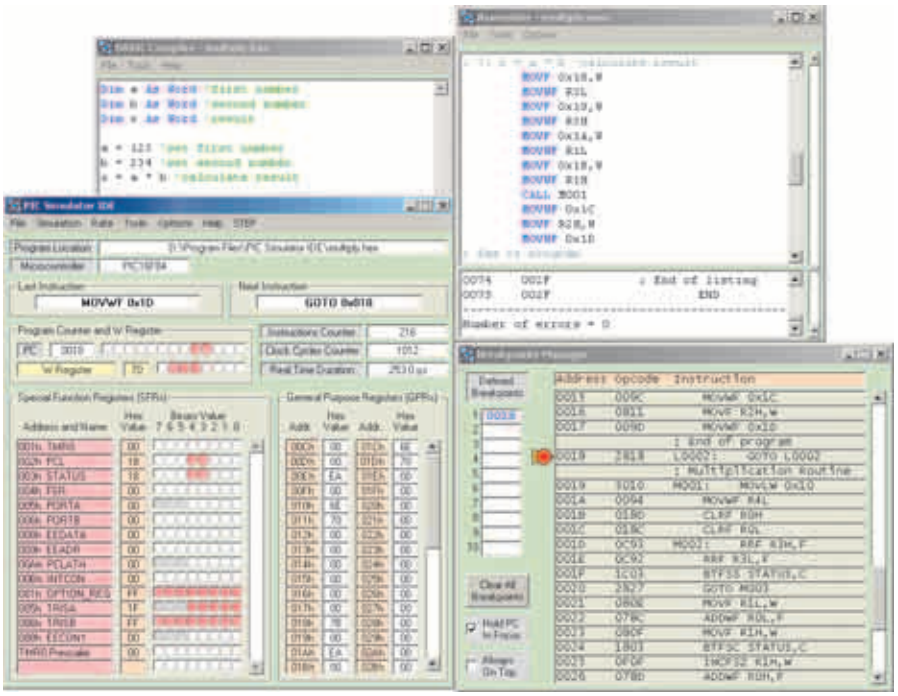

Рис. 5. Вид экрана с выполняющейся программой «Математические операции: подпрограмма умножения»

Пример 5: работа с аналого-цифровым преобразователем

Эта программа читает значение на аналоговом входе AN0 и отображает измеренные параметры на выводы порта PORTB как 8-битное значение. Текст программы из файла adc.bas имеет следующий вид:

```
Symbol ad_action =
ADCON0.GO_DONE
'новое название для бита
запуска A/D
Symbol display = PORTB
'новое название для PORTB
TRISB = %00000000
'установка ножек PORTB как выходов
TRISA = %111111
'установка ножек PORTA как входов
ADCON0 = 0xc0
'установка A/D
ADCON1 = 0
'настройка выводов PORTA как
аналоговых входов
High ADCON0.ADON
'запуск аналого-цифрового
преобразователя (A/D)

main:
Gosub getadresult
'переход в подпрограмму
преобразования
display = ADRESH
'отобразим результат
преобразования
Goto main
'бесконечное повторение программы
End

getadresult:
'подпрограмма преобразования
High ad_action
'запуск преобразования
While ad_action
'пауза для окончания преобразо-
вания
Wend
Return
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые четыре действия из примера 1, выбрав модель МК PIC16F877, и далее продолжим:

1. Выбрать файл adc.hex и нажать кнопку Open;
2. Нажать Tools|Microcontroller View (откроется окно Microcontroller View);
3. Выбрать Rate|Extremely Fast simulation rate;
4. Нажать Simulation|Start (начнется работа программы);
5. Нажать кнопку, связанную с выводом RA0/AN0 (использование «панели прокрутки» изменяет аналоговое значение на этом выводе);
6. Нажать кнопку Accept и посмотреть, как это изменит состояние выводов порта PORTB. Последние три шага можно повторить несколько раз и посмотреть на результаты. Вид экрана с выполняющейся программой показан на рис. 6.

Пример 6: работа с компаратором

Этот пример демонстрирует работу аналогового компаратора, аналоговых входов AN0 и AN1 и источника опорного напряжения. Текст программы из файла comp.bas имеет следующий вид:

```
Symbol comp_change = PIR1.CMIF
'флаг прерывания компаратора
CMCON = 0x06
'разрешим работу компаратора
TRISA = 0x07
'установка RA0, RA1 и RA2 как
входов и других ножек PORTA как
выходов
VRCON = 0xec
'настройка делителя напряжения
на 2,5 В на RA2
TRISB = 0x00
'установка выводов PORTB как
выходов

loop1:
While Not comp_change
'пауза для преобразования
Wend

PORTB = CMCON
'отобразим состояние регистра
CMCON на выводы порта PORTB, при
этом RB6 и RB7 - входы
компаратора
comp_change = 0
'сброс флага прерывания
компаратора
Goto loop1
'бесконечное повторение программы
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые четыре действия из примера 1, выбрав модель МК PIC16F628, и далее продолжим:

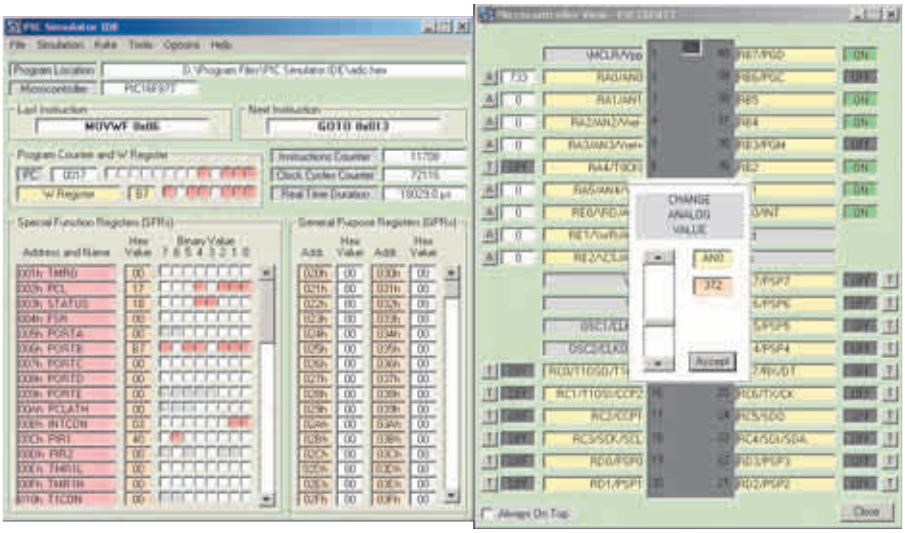

Рис. 6. Вид экрана с выполняющейся программой «Работа с аналого-цифровым преобразователем»

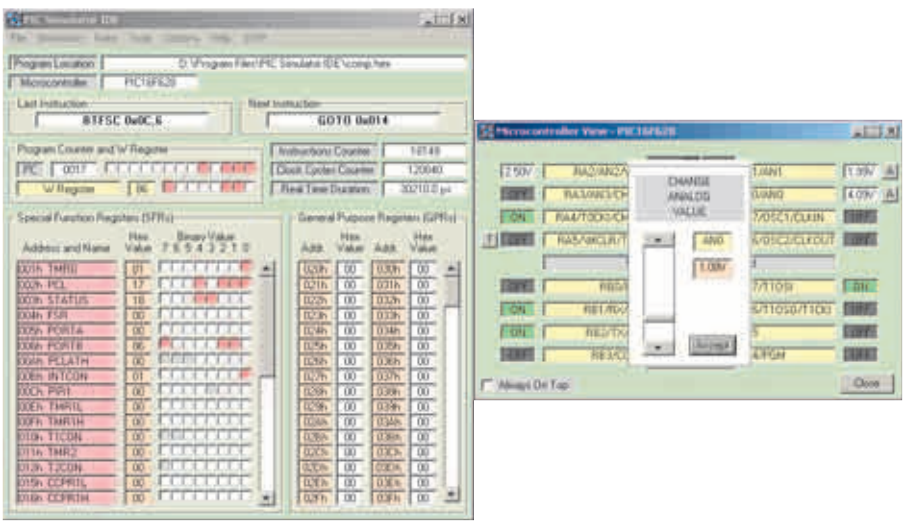

Рис. 7. Вид экрана с выполняющейся программой «Работа с компаратором»

1. Выбрать файл comp.hex и нажать кнопку Open;
2. Нажать Tools|Microcontroller View (откроется окно Microcontroller View);
3. Выбрать Rate|Extremely Fast simulation rate;
4. Нажать Simulation|Start (начнётся выполнение программы);
5. Нажать кнопку, связанную с выводами AN0 или AN1 (использование «панели прокрутки» изменяет аналоговое значение на этом вводе).

После изменения аналогового значения нужно нажать кнопку Accept и посмотреть, как это изменит состояние выводов портов PORTB и PORTA. Последние три шага можно несколько раз повторить и посмотреть на результаты. Вид экрана с выполняющейся программой показан на рис. 7.

Пример 7: работа с модулем LCD

В программе считывается аналоговое значение на аналоговом входе AN0, и полученные данные выводятся на ЖК-экран (LCD 2 × 16). Текст программы из файла lcd.bas имеет следующий вид:

```
Define ADC_CLOCK = 3
'значение по умолчанию - 3
Define ADC_SAMPLEUS = 10
'значение по умолчанию - 20
Define LCD_BITS = 8
'4 или 8 - количество линий
связи интерфейса данных
Define LCD_DREG = PORTB
Define LCD_DBIT = 0
'0 или 4 для интерфейса на 4
бита, игнорируется для
8-битового интерфейса
Define LCD_RSREG = PORTD
Define LCD_RSBIT = 1
Define LCD_EREG = PORTD
Define LCD_EBIT = 3
Define LCD_RWREG = PORTD
'по умолчанию 0, если не
используем
Define LCD_RWBIT = 2
'по умолчанию 0, если не
используем
Define LCD_COMMANDUS = 100
'задержка после LCDCMDOUT,
значение по умолчанию 5000
Define LCD_DATAUS = 10
'задержка после LCDOUT, по
умолчанию 50
Define LCD_INITMS = 1
'задержка, используемая LCDINIT,
значение по умолчанию 100
'последние три команды Define
устанавливают значения,
подходящие для моделирования; но
они должны быть опущены для
реального устройства!

Dim an0 As Word
TRISA = 0xff
'настроить все выводы PORTA как
входы
ADCON1 = 0
'все выводы PORTA - как аналого-
вые входы
Lcdinit
'инициализация LCD; курсор
выключен

loop:
 Adcin 0, an0
 Lcdcmdout LcdClear
'очистим дисплей LCD
 Lcdout "Analog input AN0"
'текст для строки 1
 Lcdcmdout LcdLine2Home
'курсор в начале строки 2
 Lcdout "Value: ", #an0
'текст для строки 2
 WaitMs 1
'в реальном устройстве должно
использоваться большее значение
Goto loop
'бесконечный цикл
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые четыре действия из примера 1, выбрав модель МК PIC16F877, и далее продолжим:

1. Выбрать файл comp.hex и нажать кнопку Open;
2. Нажать Tools|LCD (откроется окно Module LCD);
3. Нажать кнопку Setup в окне Module LCD;
4. Нажать Data Lines и установить PORTB;
5. Нажать поле Interface и установить 8 бит;
6. Нажать поле RS Line и установить PORTD, 1;
7. Нажать поле E Line и установить PORTD, 3;

8. Нажать поле R|W Line и установить PORTD, 2;
9. Нажать Apply! (закроется окно установки LCD interface);
10. Выбрать Rate|Extremely Fast simulation rate;
11. Нажать Simulation|Start (начнётся выполнение программы);
12. Нажать кнопку, связанную с выводом МК RA0/AN0 (использование «панели прокрутки» изменяет аналоговое значение на этом вводе).

После изменения аналогового значения нужно нажать кнопку Accept и подождать, пока на LCD отобразится новое значение. Последние три шага можно несколько раз повторить и посмотреть на результаты. Вид экрана с выполняющейся программой показан на рис. 8.

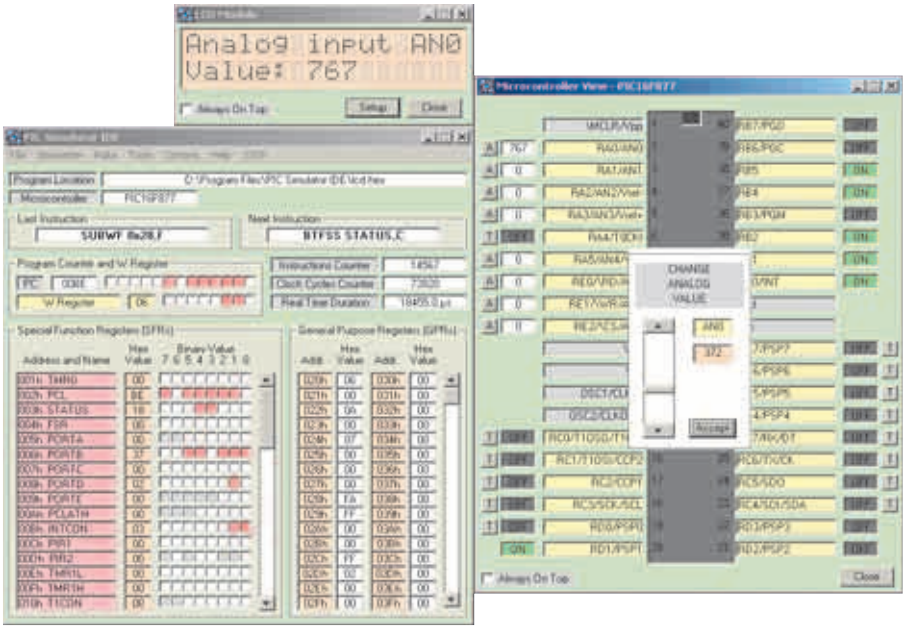

Рис. 8. Вид экрана с выполняющейся программой «Работа с модулем LCD»

Пример 8: приём и передача данных из «аппаратного» UART и отображение данных на LCD-экране

В этой программе используется несколько различных команд работы с внешними устройствами. Вначале программа посылает 6 строк данных из «аппаратного» UART, затем отвечает на полученные байты, посылая одну строку текста для каждого полученного байта. Текст программы из файла uart.bas имеет следующий вид:

```
Dim i As Byte
'объявим переменную I как байт
Hseropen 9600
'откроем порт hardware UART:
скорость 9600 бод
'WaitMs 1000
'эта задержка должна быть в
реальном устройстве
For i = 10 To 5 Step - 1
'организуем цикл с уменьшением
Hserout "Number: ", #i, CrLf
'передаём данные по
последовательному порту
'WaitMs 500
'эта задержка должна быть в
реальном устройстве
Next i

loop:
Hserin I
'ждём данные из порта;
Hserout "Number: ", #i, CrLf
'передаём данные в порт;
Goto loop
'бесконечный цикл.
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые три действия из примера 1, выбрав модель МК PIC16F877, и далее продолжим:

1. Нажать Options|Change Clock Frequency;
2. Ввести «4» и нажать кнопку OK;
3. Нажать Options|Change UART Transmit/Receive Time;
4. Ввести «100» и нажать кнопку OK;
5. Нажать File|Load Program;
6. Выбрать файл uart.hex и нажать кнопку Open;
7. Нажать Tools|Hardware UART Simulation Interface (откроется окно симуляции аппаратного интерфейса UART);
8. Выбрать Rate|Extremely Fast simulation rate;
9. Нажать Simulation|Start (начнётся выполнение программы).

После этого необходимо выждать, пока программа выдаст 6 строк текста в последовательный порт. Для пересылки данных в порт используется одна из трёх доступных кнопок интерфейса UART. Программа отвечает, посылая строку данных. Последние два шага можно несколько раз повторить и посмотреть на результаты. Вид экрана с выполняющейся программой показан на рис. 9.

Пример 9: приём и передача данных через «программный» UART

Эта программа посылает 6 строк через «программный» UART в последовательный порт (TX: PORTB.1). Затем отвечает на полученные байты (RX: PORTB.2), посылая одну строку текста для каждого полученного байта. Текст программы из файла softuart.bas имеет следующий вид:

```
Define SEROUT_DELAYUS = 500
Dim i As Byte
'объявим переменную
'WaitMs 1000
'эта задержка должна быть в ре-
альном устройстве
For i = 10 To 5 Step - 1
'цикл с уменьшением
Serout PORTB.1, 9600, "Number:
", #i, CrLf
'передаём данные через вывод
PORTB.1 (линия TX software
UART)
'WaitMs 500
'эта задержка должна быть в ре-
альном устройстве
Next i
loop:
Serin PORTB.2, 9600, I
'передаём данные через вывод
PORTB.2
'(линия TX software UART)
Serout PORTB.1, 9600, "Number:
", #i, CrLf
'передаём данные через
последовательный порт
Goto loop
'бесконечный цикл
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые три действия из примера 1, выбрав модель МК PIC16F84, два действия из примера 8 и далее продолжим:

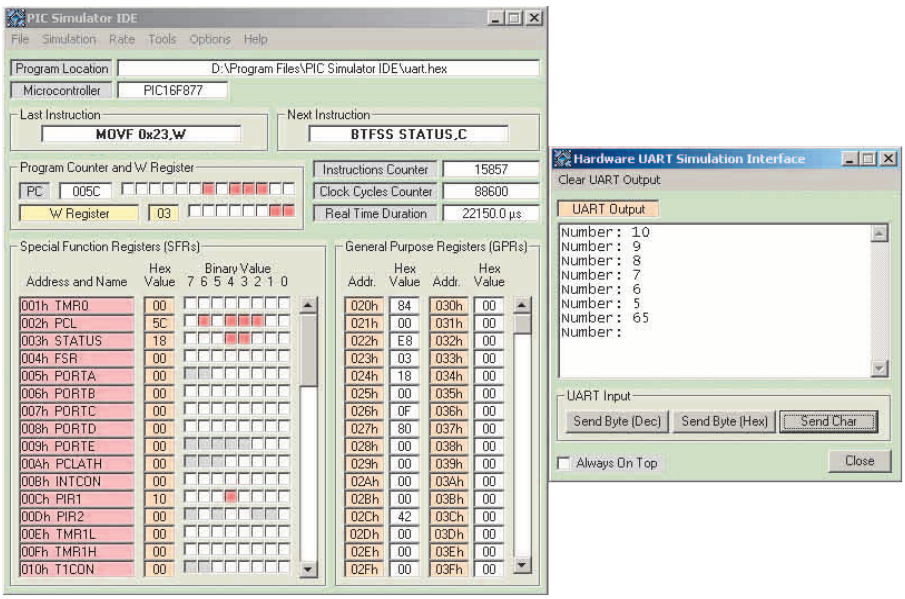

Рис. 9. Вид экрана с выполняющейся программой «Приём и передача данных из "аппаратного" UART и отображение данных на LCD экране»

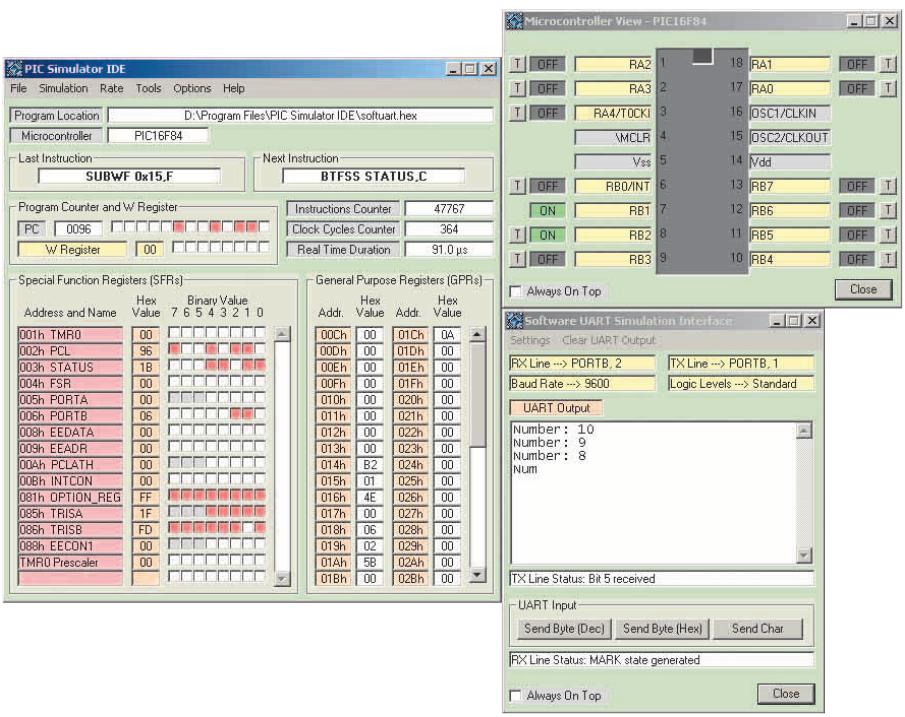

Рис. 10. Вид экрана с выполняющейся программой «Приём и передача данных через "программный" UART»

1. Нажать File|Load Program;
2. Выбрать файл softuart.hex и нажать кнопку Open (программа загружена);
3. Нажать Tools|Software UART Simulation Interface (откроется окно симуляции «программного» интерфейса UART; настройки по умолчанию: RX Line → PORTB.2, TX Line → PORTB.1, Baud Rate → 9600, Logic Levels → → Standard).
4. Нажать Tools|Microcontroller View (откроется окно Microcontroller View).
5. Выбрать скорость моделирования Rate|Ultimate (No Refresh).
6. Нажать Simulation|Start (начнётся выполнение программы).

После этого необходимо выждать, пока программа выдаст 6 строк текста в последовательный порт. Обратите внимание на состояние вывода PORTB.1 в окне Microcontroller View. Для пересылки данных в порт используется одна из трёх доступных кнопок интерфейса UART. Обратите внимание на состояние вывода PORTB.2 на окне Microcontroller View. Программа отвечает, посылая строку данных. Последние два шага можно несколько раз повторить и посмотреть на результаты. Вид экрана с выполняющейся программой показан на рис. 10.

Пример 10: отображение данных на 7-сегментном дисплее

Эта программа отображает числа от 0 до 99 на двух 7-сегментных дисплеях, которые подключены параллельно и управляются двумя линиями. Для переключения используется процедура мультиплексирования прерывания TMR0. Текст программы из файла 7segment.bas имеет следующий вид:

```
Dim digit As Byte
'входная переменная для подпро-
граммы GETMASK
Dim digit1 As Byte
'текущее значение старшей цифры
Dim digit2 As Byte
'текущее значение младшей цифры
Dim mask As Byte
'выходная переменная от подпро-
граммы GETMASK
Dim mask1 As Byte
'текущее значение старшей цифры
Dim mask2 As Byte
'текущее значение младшей цифры
Dim i As Byte
Dim phase As Bit
Symbol d1enable = PORTC.0
'линия управления для старшего
элемента 7-сегментного дисплея
Symbol d2enable = PORTC.1
'линия управления для младшего
элемента 7-сегментного дисплея
TRISB = %00000000
'настройка порта PORTB на вывод
TRISC.0 = 0
'настройка порта RC0 на вывод
TRISC.1 = 0
'настройка порта RC1 на вывод
d1enable = False
d2enable = False
mask1 = 0
mask2 = 0
phase = 0
INTCON.T0IE = 1
'включим прерывание Timer0
INTCON.GIE = 1
'включим все прерывания
OPTION_REG.T0CS = 0
'установим Timer0 на внутренний
генератор

loop:
For i = 0 To 99
digit1 = i / 10
'получим текущую цифру для
старшей цифры
```

```
digit2 = i Mod 10
'получим текущую цифру для
младшей цифры
TMR0 = 0
'сбросим Timer0, чтобы
предотвратить его прерывание
digit = digit1
Gosub getmask
'получим значение для старшей
цифры
mask1 = mask
digit = digit2
Gosub getmask
'получим значение для младшей
цифры
mask2 = mask
Gosub show1
'отобразим новое значение
старшей цифры
Gosub show2
'отобразим новое значение
младшей цифры
WaitUs 500
'задержка для моделирования
'используйте большую задержку
для реального устройства,
например WAITMS 500

Next i
Goto loop
End
On Interrupt
'подпрограмма прерывания Timer0
непрерывно переключает первый и
второй дисплей

If phase = 0 Then
phase = 1
Gosub show1
Else
phase = 0
Gosub show2
Endif
INTCON.T0IF = 0
'разрешает прерывание TMR0
Resume

getmask:
'получим соответствующее значение
7-сегментного дисплея для
входной цифры
mask = LookUp(0x3f, 0x06, 0x5b,
0x4f, 0x66, 0x6d, 0x7d, 0x07,
0x7f, 0x6f), digit
Return

show1:
'выводим старшую цифру на дисплей
d2enable = False
PORTB = mask1
d1enable = True
Return
```

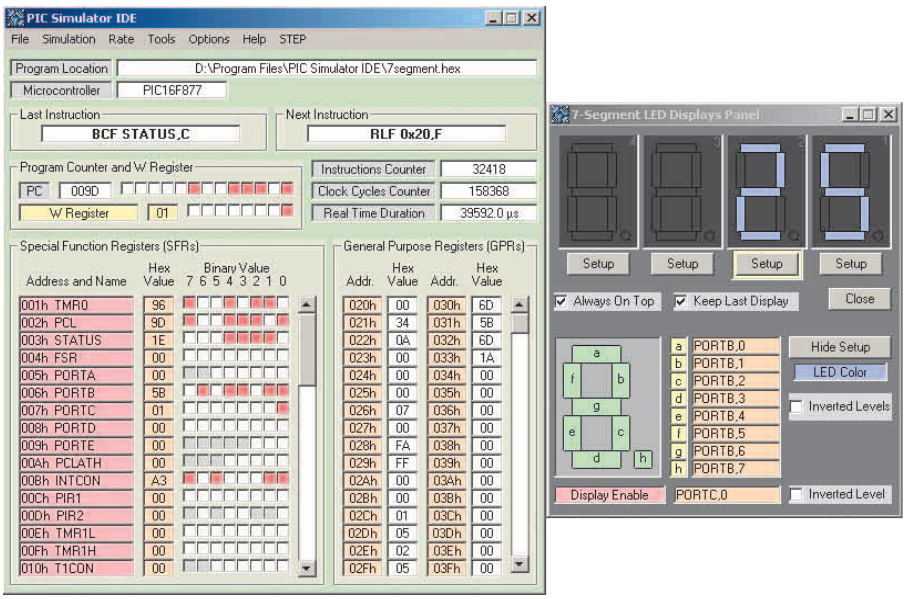

Рис. 11. Вид экрана с выполняющейся программой «Отображение данных на 7-сегментном дисплее»

```
show2:
'выводим младшую цифру на дисплей
d1enable = False
PORTB = mask2
d2enable = True
Return
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые три действия из примера 1, выбрав модель МК PIC16F877, три действия из примера 9 и далее продолжим:

1. Выбрать файл 7segment.hex и нажать кнопку Open (программа загружена);
2. Нажать Tools|7-Segment LED Displays Panel (откроется окно с четырьмя 7-сегментными дисплеями);
3. Нажать кнопку Setup ниже дисплея номер «2»;
4. Нажать оранжевое поле рядом с дисплеем, чтобы включить|отключить этот дисплей;
5. Ввести «4», чтобы выбрать PORTC и затем нажать OK;
6. Ввести «0», чтобы выбрать вывод RC0 и затем нажать OK;
7. Нажать кнопку Setup ниже дисплея номер «1»;
8. Нажать оранжевое поле рядом с дисплеем, чтобы включить|отключить этот дисплей;
9. Ввести «4», чтобы выбрать PORTC и затем нажать OK;
10. Ввести «1», чтобы выбрать вывод RC1 и затем нажать OK;
11. Нажать кнопку Hide Setup, чтобы сохранить немного экранного пространства;
12. Выбрать Rate|Ultimate (No Refresh);
13. Нажать Simulation|Start (начнётся выполнение программы).

Программа отобразит числа от 0 до 99 на двух 7-сегментных дисплеях, используя процедуру мультиплексирования прерывания TMR0. Сохраняя данные на экране, необходимо экспериментировать с опцией Keep Last Display. Вид экрана с выполняющейся программой показан на рис. 11.

Пример 11: работа с генератором сигналов и осциллографом

Этот пример демонстрирует работу осциллографа и генератора сигналов. Также в этой программе приведены примеры работы с внешней памятью (EEPROM) по протоколу I²C. Эта подпрограмма использует протокол связи I²C и заносит значения в первые 32 ячейки внешней EEPROM. Текст программы из файла i2c.bas имеет следующий вид:

```
Dim addr As Word
'переменная для хранения адреса
EEPROM
Dim data As Byte
'переменная для хранения байта
данных EEPROM
Symbol sda = PORTC.2  'новое имя
для вывода SDA
Symbol scl = PORTC.3  'новое имя
для вывода SCL

For addr = 0 To 31
'организуем цикл. Будут записаны
первые 32 байта
data = 255 - addr
'значение байта данных для записи
```

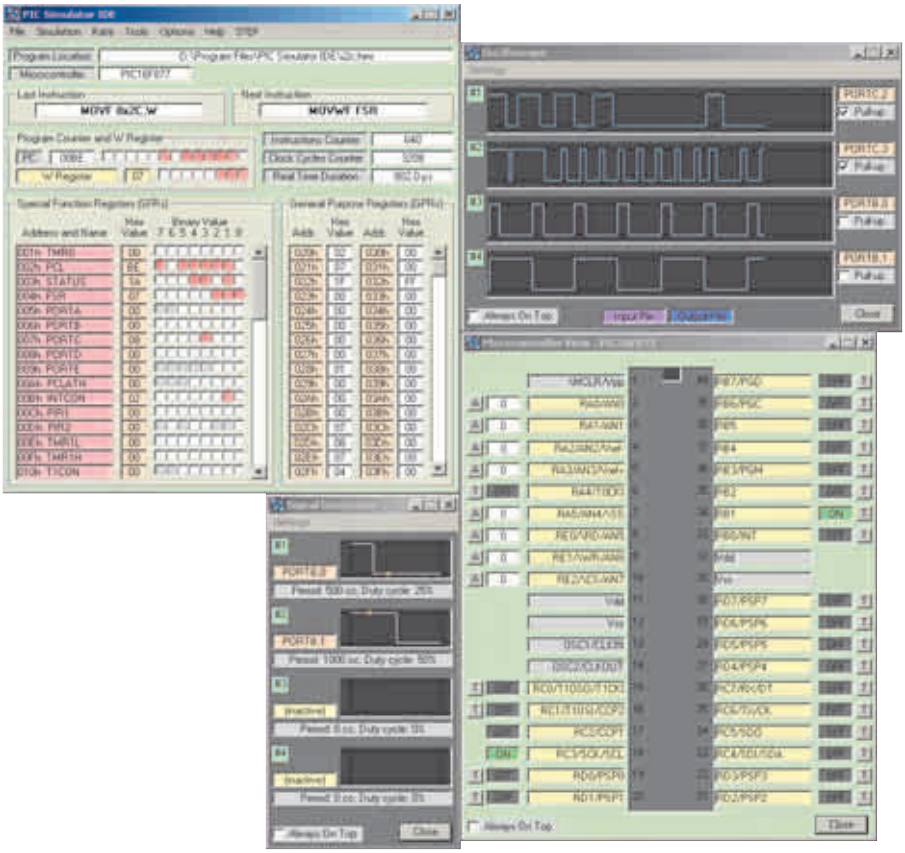

Рис. 12. Вид экрана с выполняющейся программой «Работа с генератором сигналов и осциллографом»

```
I2CWrite sda, scl, 0xa0, addr,
data    'используем протокол I²C
для записи в EEPROM
WaitMs 1
'маленькая задержка
Next addr
```

Просмотрим результаты работы этой программы в PIC Simulator IDE. Для этого выполним первые шесть действий из примера 10 и далее продолжим:

1. Выбрать файл i2c.hex и нажать кнопку Open (программа загружена);
2. Нажать Tools|Oscilloscope (откроется окно четырёхканального осциллографа);
3. Нажать Canal Settings|Turn On/Off Oscilloscope 1;
4. Ввести «2», чтобы выбрать PORTC для канала 1, и нажать OK;
5. Ввести «2», чтобы выбрать вывод RC2 для канала 1, и нажать OK;
6. Выбрать опцию Pull-up Select для канала 1;
7. Нажать канал Settings|Turn On/Off Oscilloscope 2;
8. Ввести «2», чтобы выбрать PORTC для канала 2, и нажать OK;
9. Ввести «3», чтобы выбрать вывод RC3 для канала 2, и нажать OK;
10. Выбрать опцию Pull-up Select для канала 2;
11. Нажать Canal Settings|Turn On/Off Oscilloscope 3;
12. Ввести «1», чтобы выбрать PORTB для канала 3, и нажать OK;
13. Ввести «0», чтобы выбрать вывод RB0 для канала 3, и нажать OK;
14. Нажать Canal Settings|Turn On/Off Oscilloscope 4;
15. Ввести «1», чтобы выбрать PORTB для канала 4, и нажать OK;
16. Ввести «1», чтобы выбрать вывод RB1 для канала 4, и нажать OK;
17. Нажать Tools|Signal Generator (откроется окно 4-канального генератора импульсов);
18. Нажать Settings|Turn On/Off Signal Generator 1;
19. Ввести «1», чтобы выбрать PORTB для канала 1, и нажать OK;
20. Ввести «0», чтобы выбрать вывод RB0 для канала 1, и нажать OK;
21. Ввести «500», чтобы определить период импульсов для канала 1, и нажать OK;
22. Ввести «25», чтобы определить режим работы для канала 1, и нажать OK;
23. Нажать Settings|Turn On/Off Signal Generator 2;
24. Ввести «1», чтобы выбрать PORTB для канала 2, и нажать OK;
25. Ввести «1», чтобы выбрать вывод RB1 для канала 2, и нажать OK;
26. Ввести «1000», чтобы определить период импульсов для канала 2, и нажать OK;
27. Ввести «50», чтобы определить режим работы для канала 2, и нажать OK;
28. Нажать Tools|Microcontroller View (откроется окно Microcontroller View; окна на экране установить так, чтобы получить лучшее представление);
29. Выбрать Rate|Extremely Fast simulation rate;
30. Нажать Simulation|Start (начнётся выполнение программы).

Процесс связи по протоколу I²C можно просмотреть на осциллографе как пачки импульсов. Вид экрана с выполняющейся программой показан на рис. 12.

Проработав примеры работы с программой PIC Simulator IDE, вы сможете самостоятельно тестировать свои программы. В приложении приведены адреса документов, отражающих вопросы работы с микроконтроллерами microPIC компании Microchip.

Литература

1. www.oshonsoft.com.
2. www.picbasic.narod.ru.

Приложение

Схема платы LAB-X3, для ICD с PIC16F628
http://www.melabs.com/downloads/labx3sch.pdf
Микроконтроллер PIC16C84. Краткое описание
http://www.nnov.rfnet.ru/pic/16c84.html
Набор команд PIC16XXX
http://yusoft.kulichki.com/russian/pic/opcodes.htm
Характеристики PIC16F628
http://www.microchip.ru:80/lit/pic/pic16f6xx/pic16f628
Полезные подпрограммы для PIC-контроллеров
http://www.kazus.ru/modules.php?name=News&file=article&sid=410
Самоучитель по программированию PIC-контроллеров для начинающих
http://ikarab.narod.ru/Kea_20.html
http://www.nnov.rfnet.ru/pic/first.html
Советы по программированию и схемотехнике
http://www.disall.newmail.ru/faq.htm
«Глюки» микроконтроллеров PIC
http://www.disall.newmail.ru/gluk.htm
FAQ по PIC-микроконтроллерам
http://yusoft.kulichki.com/russian/pic/faq.htm
Много ссылок по PIC-микроконтроллерам
http://yl2.narod.ru/rpic-lnk.htm

При поддержке Российского отделения
"Общества информационных дисплеев"
SID
RUSSIA CHAPTER

СПЕЦИАЛИЗИРОВАННАЯ ВЫСТАВКА

DISPLAY

14-16 июня
2006
МОСКВА ЦДХ

- Системы отображения специального исполнения
- Универсальные и специализированные экраны
- Отображение в системах связи и управления
- Городские информационно-справочные системы
- Средства отображения в рекламных технологиях
- Компоненты систем и средств отображения
- Информационные табло, мониторы, дисплеи
- Проекционное оборудование и системы
- Средства отображения на транспорте
- Средства отображения в военной технике

http://display.chipexpo.ru