Шестая межрегиональная научная конференция
«Шаг в будущее, Москва»

регистрационный номер

Информатика и системы управления

Интегрированная среда разработки для микроконтроллеров на ядре PIC-micro

Автор: *Константинов Петр Алексеевич*
лицей 1580, 11 класс

Руководитель: *Дединский Илья Рудольфович*
преподаватель программирования лицея 1580

Москва 2003

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время RISC-процессоры находят все большее применение в различных областях. Причина их успеха в их мобильности, высокой производительности и низкой цене. Типичные представители RISC-процессоров –однокристальные микроконтроллеры PIC (Programmable Interface Controller) фирмы Microchip. Это высокопроизводительные и, вместе с тем, дешёвые RISC-процессоры, поэтому они лучше всего подходят для разработки простых электронных устройств, где уже не хватает возможностей традиционной КМОП-логики.

Однако изучение этих устройств является достаточно проблематичным. Отсутствие простой документации на русском языке и труднодоступность средств разработки могут отпугнуть человека, раньше не имевшего дело с подобными устройствами. Но несмотря на эти недостатки, микроконтроллеры являются хорошей платформой для изучения низкоуровневого и системного программирования: они имеют простую архитектуру, понятную даже новичку, сравнительно небольшой набор команд, а также богатую самодостаточную периферию, с помощью которой можно создавать законченные приложения. Такие микроконтроллеры – хорошая база для изучения низкоуровневого программирования.

Программирование на PIC сопряжено с использованием следующего программного комплекса: ассемблера, дизассемблера, эмулятора и программатора. Ассемблер (компилятор) преобразует исходный текст программы на языке ассемблера в последовательность машинных кодов, исполняемых ядром контроллера. Дизассемблер же наоборот – переводит последовательность машинных кодов в исходный текст на языке ассемблера. Эмулятор имитирует исполнение команд ядром микроконтроллера и является основным отладочным модулем. Программатор записывает программу с жесткого диска компьютера в память микроконтроллера и обратно.

Анализ существующих разработок

На момент написания программы существуют следующие средства программирования для PIC-контроллеров:

А) MPLIB (интегрированная среда разработки)

Разработчик – Microchip

Текущая версия – 5.61

MPLIB это комплексная среда разработки для микроконтроллеров на базе ядра PICmicro. Включает в себя все необходимое для разработки приложений для этого семейства микроконтроллеров. Этим программным продуктом пользуется большая часть разработчиков для PICmicro.

Другие разработки представляют собой различные не связанные компоненты, необходимые для разработки приложений. Большинство этих продуктов созданы усилиями энтузиастов, и количество их исчисляется десятками, поэтому, в данной работе нет возможности провести их подробный анализ. Рассмотрим лишь самые интересные из них.

Б) Assembler for Microchip PIC16Cxx microcontrollers (компилятор)

Разработчик – Timo Rossi

Текущая версия – 1.10beta

Это мощный ассемблер поддерживает множество инструментов, которыми пользуется современный программист, таких, как макросы, условная компиляция и другие. Генерирует код для большого количества микроконтроллеров PIC. Многие разработчики используют этот продукт, вместо стандартного компилятора фирмы Microchip.

В) SimulPIC(эмулятор)

Разработчик – Tommaso Cucinotta, Alessandro Evangelista, Luigi Rizzo

Текущая версия – 1.0

SimulPIC – развитой эмулятор микроконтроллера PIC16C84. SimulPIC почти полностью эмулирует все особенности этого микроконтроллера. Отличительная особенность этого продукта – точный расчет времени исполнения кода, что часто бывает необходимо при отладке приложений взаимодействующих с внешними устройствами. Неудобство данной разработки заключается в использовании интерфейса командной строки.

Г) COMPIC-1(программатор)

Разработчик – ORMIX Ltd.

Текущая версия – 1.54beta

COMPIC-1 удобный программатор, поддерживающий кроме микроконтроллеров PIC множество других микросхем. В отличие от других программаторов COMPIC достаточно прост, чтобы его можно было изготовить самостоятельно.

Сравнительный анализ вышеописанных продуктов приведен в таблице 1.

Табл. 1. Сравнительный анализ продуктов, используемых при разработке на PIC.

Названия	Функции	Достоинства	Недостатки
MPLIB	Интегрированная среда разработки	Поддерживает все существующие на данный момент контроллеры, связанность компонентов	Несовместимость интерфейса, компилятора с известными пакетами разработки, сложность использования
Assembler for Microchip PIC16Cxx microcontrollers	Компиляция	Поддержка инструментов современного программирования, кросс-платформенность	Не связанность с другими компонентами необходимыми для разработки
SimulPIC	Эмуляция	Наиболее полная эмуляция ядра контроллера	Интерфейс командной строки
COMPIC-1	Программация[1]	Дружелюбный интерфейс, простота изготовления устройства	Не связанность с другими компонентами необходимыми для разработки

Актуальность темы

Анализ существующих разработок показал, что почти все они имеют некоторые недостатки:

1. Разрозненность, как следствие отсутствия единого интерфейса взаимодействия компонентов, что не дает возможности сопряжения компонентов в единый комплекс.
2. Несовместимость пользовательского интерфейса с распространенными пакетами разработки.
3. Сложность использования, что может отпугнуть новичка.

[1] Термин «программация» (не «программирование») означает запись откомпилированной программы на кристалл контроллера. В обиходе часто используется термин «прошивка».

В этой работе была предпринята попытка устранения этих недостатков. Был использован распространенный стандарт оконного пользовательского интерфейса. Также разработан полный комплекс простых в использовании программ необходимых для написания и отладки продуктов для микроконтроллеров PIC.

Цель работы

Разработать среду программирования для микроконтроллеров – комплекс программ, необходимых для разработки (компилятор, деассемблер, эмулятор и программатор) и интегрированную среду разработки, связывающий все эти компоненты. Среда должна иметь компонентную архитектуру, позволяющую подключать новые модули без перекомпиляции других, представлять собой законченный и связанный комплект продуктов для полного цикла разработки приложений для PIC и быть достаточно простой и удобной для конечного пользователя.

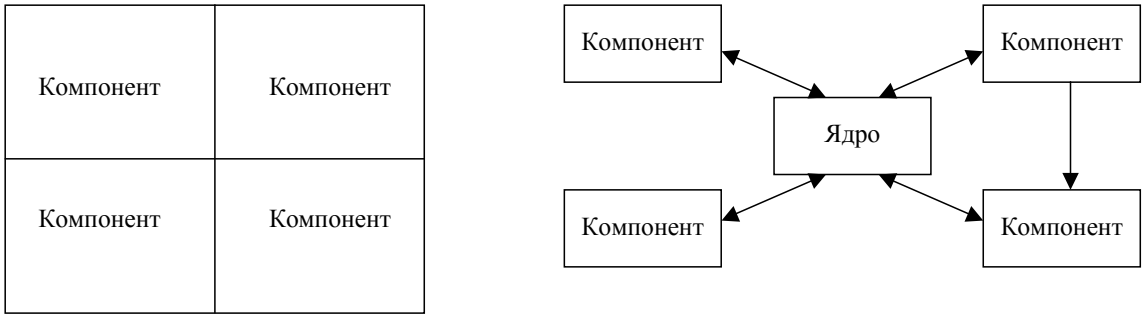
Область применения

Среда может быть использована как для проектирования конечных устройств, так и для изучения микроконтроллеров PIC.

1. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

1.1. Проектирование интегрированной среды разработки (ИСР)

Интегрированная среда разработки – множество программных продуктов для разработки приложений (компилятор, отладчик, редактор и т.д.) объединенный единым пользовательским интерфейсом. Все составляющие ИСР могут быть соединены в один запускаемый модуль или разделены на множество программ-компонентов. По такому критерию различают две архитектуры построения ИСР: монолитную и компонентную.



(а) (б)

Рис 1.1 Монолитная (а) архитектуры и компонентная (б) построения ИСР.

Основное отличие компонентной архитектуры от монолитной (см. рис. 1.1) – маштабируемость, то есть возможность изменять какой-либо компонент без изменения других компонентов. В данном проекте, при построение ИСР использована компонентная архитектура.

Определив архитектуру построения ИСР, определим набор компонентов достаточный для разработки приложений. Основные составляющие ИСР это набор программ-компонентов для разработки приложений, интерфейс пользователя и компонент связи. В этой работе компонент связи и интерфейс пользователя объединены в одни компонент, который далее будем называть основным модулем. Из всего вышеизложенного следуют задачи:

1) Разработать интерфейс пользователя

2) Разработать связующий модуль

3) Разработать набор программ-компонентов для разработки приложений.

Отметим, что в набор программ-компонентов для разработки приложений входят следующие приложения: компилятор языка ассемблера, эмулятор ядра микроконтроллера, программатор и дизассемблер.

1.2. Разработка компонента связи

Назначение компонента связи - осуществлять связи между интерфейсом пользователя и программным комплексом для разработки приложений, поэтому мы рассмотрим его в первую очередь. В этой работе компонент связи объединен с интерфейсом пользователя для более эффективного взаимодействия.

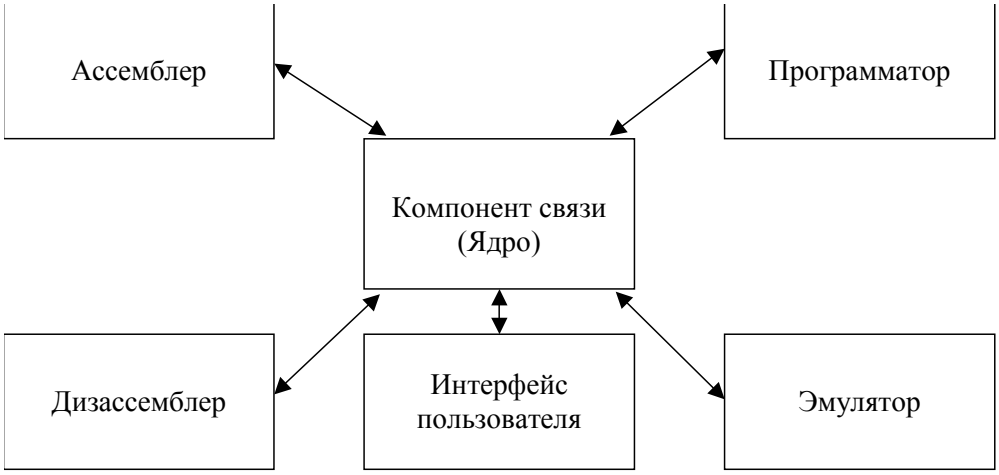


Рис 1.2 Схема работы компонента связи

В этом проекте все внешние компоненты реализованы, как загружаемые модули, а компонент связи – обычная функция DOS, загружающая программы-компоненты.

Листинг 1.1. Пример загрузки внешнего компонента

```
int asm_func ()
    {
    if (prog->GetAct() == NULL)return 0;
    if (!prog->GetAct()->Save(NULL))
        {
        CWindow* wnd = NEW (CWarning("Can not assembly file"," Error "));
        prog->AddWnd(wnd);
        return 0;
        }
```

```
if ((spawnlp(P_WAIT, "asm.exe", " ", prog->GetAct()->title_c, NULL)) == -1)
    {
    CWindow* wnd = NEW (CWarning("Can not exec ASM.EXE"," Error "));
    prog->AddWnd(wnd);
    return 0;
    }
prog->ReDraw();
return 1;
}
```

Из листинга 1.1 видно, что вызов компонента происходит с помощью библиотечной функции «swapnlp», и в случаи ошибки выдается соответствующее сообщение. Так реализован компонент связи для ассемблера, программатора и дизассемблера, однако компонент связи для эмулятора имеет более сложное строение — он реализован как отдельный класс окна отладчика, более подробно строение этого класса описано ниже.

1.2.1. Класс CDebugWnd

Класс CDebugWnd организует связь между отладчиком и основным модулем. Этот класс - потомок класса CContainer (см. ниже) содержащий в себе и управляющий пятью списками: списком команд, списками регистров, списком данных, содержащихся в энергонезависимой памяти (EEPROM), и списком стека. При любом изменении данных в этих списках происходит изменение файла данных отладчика (out.sim), и на следующем шаге отладки будут использованы новые данные. Также этот класс трассировкой программы.

1.3. Проектирование интерфейса пользователя

В широко известных ИСР использован оконный интерфейс, как показала практика такой тип интерфейса наиболее удобен пользователю и не сложен в реализации при использовании объектно-ориентированного подхода. Эти факторы послужили поводом для выбора именно оконного интерфейса при построении этой ИСР. Итак встает задача по разработке оконной библиотеке для простого и эффективного построения пользовательского интерфейса.

1.3.1. Структура и описание классов оконной библиотеки

Из рисунка 1.3 видно, что все используемые при построении интерфейса делятся на

несколько категории: менеджеры, такие как CManager и CProg, окна – потомки класса CWindow, окна-дети – потомки класса CchildWnd и классы ввода-вывода такие как CSwap и CScreen. В следующих разделах будут описаны основные классы-родители и их методы.

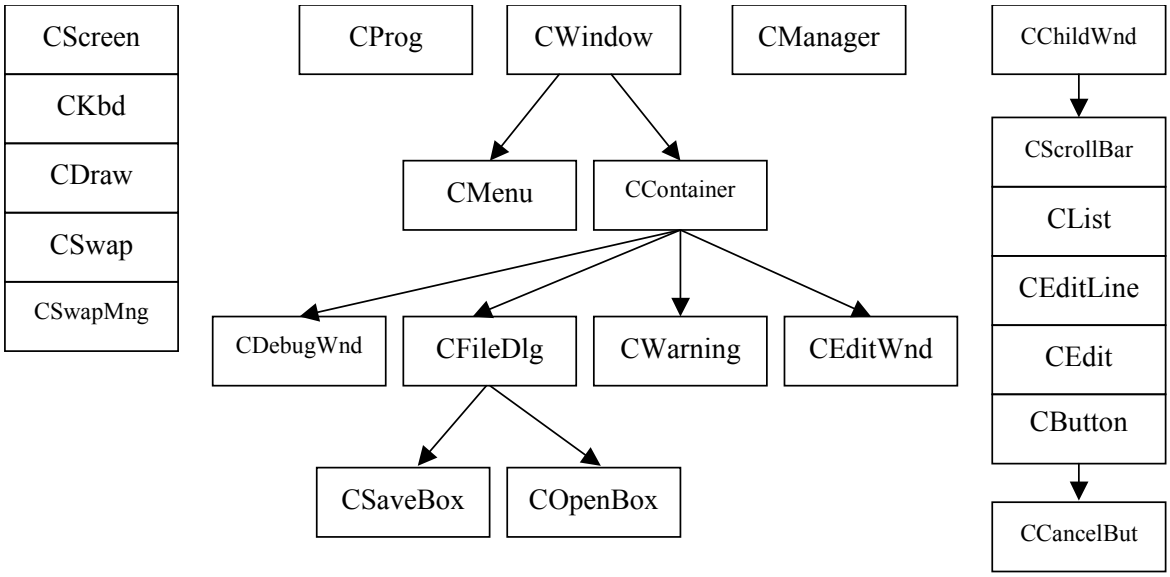


Рис 1.3 Структура классов оконной библиотеки

1.3.1.1. Класс CManager

Класс CManager – основной класс, организующий работу с окнами. Он содержит контейнер указателей на объекты классов наследуемых от класса CWindow и содержит набор методов для работы в этим контейнером. Это методы Add и Del позволяющие добавлять объекты в этот контейнер и удалять объекты из этого контейнера. Также реализованы методы для организации работы окон. Набор этих методов включает в себя методы для перерисовки, закрытия, перемещения, изменения размера окон (ReDraw, Close, Move, Size, Zoom) ,а также методы для изменения активного окна и получения указателя на активное окно (Next и GetAct). Основной метод класса CManager - Run этот метод содержит основной цикл, организующий взаимодействие между пользователем и окном.

1.3.1.2. Класс CWindow

Класс CWindow описывает основной объект оконной системы – окно. Этот класс управляется непосредственно оконным менеджером (CManager) и содержит следующий ныбор методов: ReDraw, OutText, ManMes и OnKeyDo. С помощью методов ReDraw, ManMes и OnKeyDo окно перерисовывается и получает информацию от менеджера: на-

жатую пользователем кнопку при помощи метода OnKeyDo и "сообщение" от менеджера при помощи метода ManMes. При помощи "сообщений" окно получает информацию о изменении своего размера и положения на экране. Кроме того окно содержит контейнер отображаемой на нем текстовой информации, управление этим контейнером осуществляется методам OutText.

1.3.1.3. Класс CContainer

Класс CContainer с одной стороны наследуется от класса CWindow, а с другой содержит контейнер объектов типа CChildWnd и управляет ими, как класс CManager управляет объектами типа CWindow. Таким образом, набор методов класса CContainer сочетает в себе методы по управлению контейнером (Add и Del) и методы класса CWindow.

1.3.1.4. CChildWnd

Как описано выше, объекты класса CChildWnd, содержатся и управляются объектами класса CContainer. От этого класса наследуются такие классы как CList – класс списка, CButton - класс кнопки, CScroll – класс движущивося ползунка-индикатора и другие. Эти классы наряду с классом окна – важнейшие компоненты пользовательского интерфейса. Набор методов класса CChildWnd включает в себя методы перерисовки (ReDrawAct, ReDraw) и методы взаимодействия (OnKeyDo, Run) все эти методы виртуальны, а класс CChildWnd абстрактен.

1.3.1.5. CButton

Класс CButton – класс кнопки. Этот класс – потомок абстрактного класса CChildWnd, рассматривается в качестве примера реализации других классов-потомков класса CChildWnd. В этом классе переопределены методы активной (ReDrawAct) и обычной (ReDraw) перерисовки, а также метод Run, позваляющий выполнять кнопке функцию при нажатии на неё. Кроме методов класса-родителя, во многих классах-потомках CChildWnd определены специфичные для этих классов методы. Так например в классе CButton реализован метод HotKey, специфичный для этого класса.

1.3.2. Проектирование главного меню

Главное меню – инструмент доступа к основным функциям ИСР. В следующих разделах описывается принцип проектирования меню, как части оконной библиотеки и

принцип выбора пунктов главного меню.

1.3.2.1. Класс CMenu

Класс CMenu – потомок класса CWindow, содержащий контейнер пунктов меню, являющихся объектами класса CButton. Таким образом класс CMenu объединяет в себе методы класса CWindow и методы по управлению контейнером пунктов меню (AddElement), кроме того этот класс содержит набор специфичных методов (TDraw, TDrawAct, NextTest) которые осуществляют рисование заголовка меню и переключение между соседними меню. Объект класса CMenu управляется оконным менеджером.

1.3.2.2. Принцип выбора пунктов меню

Основные критерии использованные при выборе пунктов главного меню: полнота и функциональность. То есть главное меню должно содержать все необходимые пункты меню и быть простым в использовании. Исходя из этих критериев главное меню поделено на следующие всплывающие подменю: меню файловых операций (Files), меню компиляции (Compile), меню отладки (Debug), меню программации (Program), меню оконных операций (Window), меню встроенной помощи (Help). Набор пунктов, включённых в эти меню, а также их использование описаны ниже.

1.3.3. Проектирование встроенных возможностей

При проектировании компонентной ИСР целесообразно совместить интерфейс пользователя и часть средств для разработки приложений. В этой работе такими средствами стали редактор и система помощи. Кроме того в этом разделе будет рассмотрен механизм работы ИСР с оперативной памятью.

1.3.3.1. Принцип построения встроенного редактора

Использование редактор – неотъемлемая часть цикла разработки приложений. В этой работе редактор реализован как набор из двух классов: класса окна редактора CEditWnd и собственно компонента-редактора класса CEditor.

Класс CEditWnd – наследуется от класса CContainer и управляет объектом класса CEditor и двумя объектами класса CScroll: горизонтальной и вертикальной полосками прокрутки. Класс CEditor обрабатывает и отображает введенный пользователем текст, а также управляет навигацией по тексту. В качестве "хранилища" данных CEditor исполь-

зует объект класса CSwap описанного ниже. Использование такого метода хранения данных позволяет содержать в каждый момент времени лишь 50 строк из 1000 возможных стров редактирования текста.

1.3.3.2. Механизм работы с оперативной памятью

Описываемая ИСР работает в реальном режиме и имеет доступ только к первому мегабайту оперативной памяти. Если бы окна, обрабатывающие большие объемы данных (такие как редактор или отладчик) хранили все используемую ими информацию в оперативной памяти, то при открытии нескольких таких окон свободная оперативная память быстро закончилась бы. Для решения этой проблемы в ИСР реализован механизм "свопинга" – хранения редко используемой информации на жестком диске. Этот механизм реализован в шаблонном классе CSwap. Применение этого механизма, при использование объекта класса CSwap незаметно для использующей его функции, при использовании он представляется, как массив указателей на объекты, определенные при инициализации.

Листинг 1.2 Использование объекта класса Cswap

```
...
CSwap<char[MAX_EDIT_SYM]> buf_c;
...
for (i = 0; i <= MAX_EDIT_SYM; i++)
     {
     buf_c[0][i] = ' ';
     }
```

Такой эффект “прозрачности” достигается благодаря перегрузке оператора "[]". Для выбора информацию, записываемую на жесткий диск CSwap содержит в себе кэш-таблицу, сочетающую обращение к каждому элементу данных. Также реализован класс управления свопингом CSwapMng, так что можно использовать несколько объектов класса CSwap одновременно.

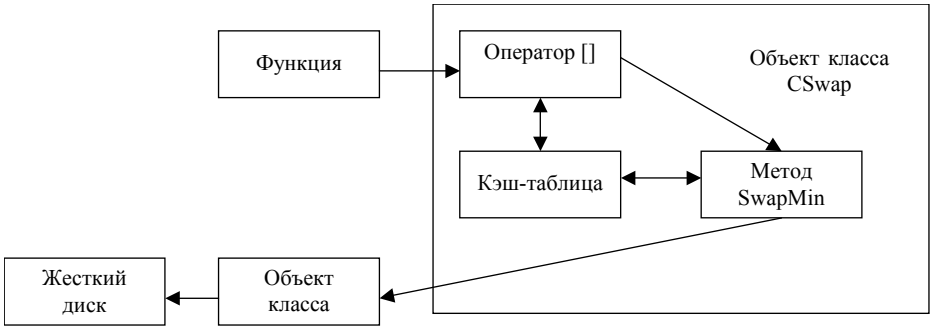


Рис 1.4 Схема работы механизма свопинга

1.4. Проектирование программного комплекса для разработки приложений

Как было описано выше, в набор программ-компонентов для разработки приложений входят следующие компоненты: компилятор языка ассемблера, эмулятор ядра микроконтроллера, программатор и дизассемблер. Опишем принцип построения каждого из них.

1.4.1. Проектирование ассемблера

Разработка компилятора даже такого простого языка как ассемблер, не токая простая задача, как может показаться на первый взгляд. Основная задача, возникающая при построении компилятора – придания компилируемому коду, определенную свободу. То есть возможность использования директив компилятора, таких как поиск-замена (equ) или добавление файла (include). Однако эта задача легко реализуема, при реализации простого лингвистического анализатора (см рис. 1.5).

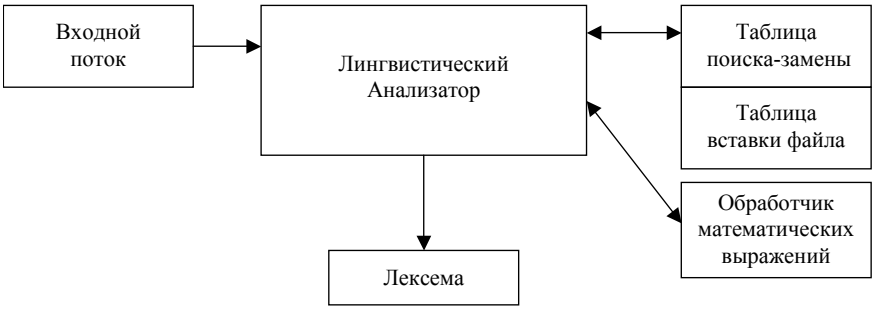


Рис. 1.5 Схема работы лингвистического анализатора.

На рисунки 1.5 показан видно, что лингвистический анализатор использует таблицы поиска-замены и вставки файла при обработке файла. На выходи лингвистический анализатор формирует лексему, либо числовое значение, обрабатываемое основным циклом компилятора. При компиляции основной цикл совершает несколько проходов по компилируемому файлу. При первом проходе компилятор формирует таблице поиска-замены и вставки файла. Следующие три прохода уже производят компиляцию, при этом файл ошибок (разрешение err) формируется только на последнем проходе. Если компилируемый файл не содержал ошибок, после последнего прохода будет скомпилирован обрабатываемый файл с расширением HEX.

1.4.2. Проектирование дизассемблера

Дизассемблер обрабатывает входной файл в формате Intel 16 Hex. После дизассемблирования получается файл с расширением ASM содержащий исходный текст программы. Сам дизассемблер состоит из двух частей: ядра и управляющего ядром кода. Ядро принимает на вход слов данных и возвращает строку символов, содержащую мнемонику команды. Управляющий код производит первичный проход, дизассемблируя команды и вторичный, восстанавливая метки и шестьнадцатиричные числа. Полученный после дизассемблирования файл может быть использован для повторной компиляции.

1.4.3. Проектирование отладчика

При проектировании отладчика, встроенного в ИСР, решаются два основные задачи: разработка ядра отладчика (эмулятора) и компонента связи между ядром отладчика и интерфейсом пользователя. Основные задачи ядра отладчика: "исполняя" поступающие на вход машинные команды изменять область данных эмулируемого процессора (регистры, стек и т.д.) и имитировать работу основных периферийных устройств процессора (таймеры, порты ввода-вывода). Применимо к этой работе, эмулятор должен имитировать исполнение команд и работу основных периферийных устройств ядра микроконтроллера pic16f84.

Рис. 1.6 Схема отладчика

Ядро эмулятора взаимодействует со средой с помощью рабочих файлов (см . рис. 1.6) в этих файлах хранится информация находящаяся в ядре микроконтроллера в данный момент.

1.4.4. Проектирование программатора

Программатор – устройство взаимодействующее с памятью контроллера, с его помощью можно читать и писать программу и данные из ядра контроллера. При реализации приложения управляющего программатором был использован объектно-ориентированный подход, что значительно упростило разработку. При этом были реали-

зованы классы CCom и CPic (см. Рис 1.7).

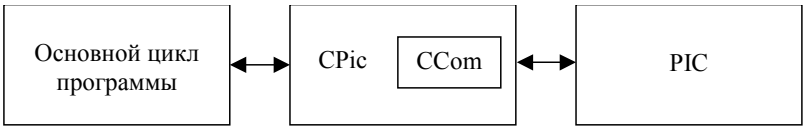


Рис. 1.7 Схема работы программатора

2. МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

2.1. Системные требования

Минимальные требования:

1. Процессор Intel 80386.
2. Оперативная память 1 Мб.
3. Минимальное свободное пространство на диске 500 Кб.
4. Операционная система MS-DOS 6.22, или совместимая с ней.

Рекомендуемые требования:

1. Процессор Pentium 500 или выше.
2. Оперативная память 1 Мб.
3. Свободное пространство на диске 2 Мб.
4. Операционная система MS-DOS 6.22, или совместимая с ней.

На редактируемые файлы накладываются следующие ограничения: максимальное количество строк – 1000, максимальное количество символов в строке – 256.

2.2. Установка

Запустите файл PicLib.exe с дискеты и дождитесь окончания установки.

2.3. Методика использования программного комплекса

После успешной установки, загрузите управляющий модуль программного комплекса Core.exe(см. рис. 3.1). На этом рисунку мы видим всплывающее меню «Files», и два открытых окна редактора.

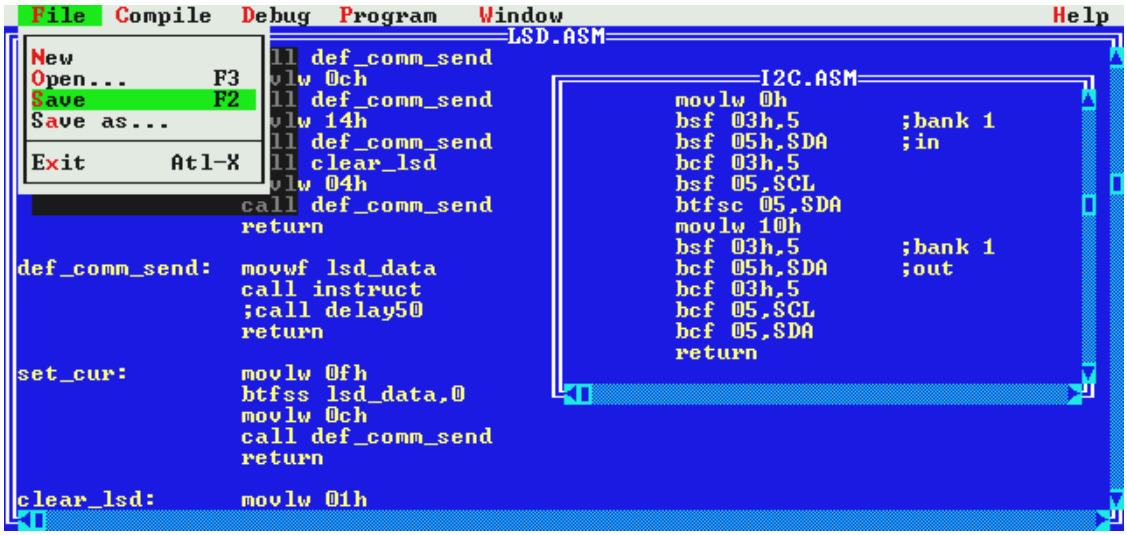


Рис. 2.1. Основное окно программы.

Для работы с меню можно пользоваться следующими функциональными клавишами:

Alt + [«горячая» клавиша меню] – Вызов меню.

Alt + F – Вызов меню «Files».

Alt + C – Вызов меню «Compile».

Alt + D – Вызов меню «Debug».

Alt + P – Вызов меню «Program».

Alt + W – Вызов меню «Window».

Alt + H – Вызов меню «Help».

F10 – Вызов меню «Files».

Клавиши-стрелки - навигация в меню.

Enter – Выполнение выбранного пункта меню.

Esc – Выход из меню.

2.3.1. Принцип построения меню файловых операций

Для эффективного использования интегрированной среды разработки необходимо реализовать в ней возможность работы с файлами. Необходимым минимумом в этом случаи будет являться набор из следующих возможностей: возможность создавать, открывать и редактировать файлы, возможность сохранять отредактированные файлы. Методика использования реализованных в среде файловых операций описана ниже.

Рис. 2.2. Меню файловых операций

New – создать новый файл и открыть его в редакторе. В этом случаи файл будет носить имя «nonameXX.asm», где XX порядковый номер файла, созданного с начала сеанса работы.

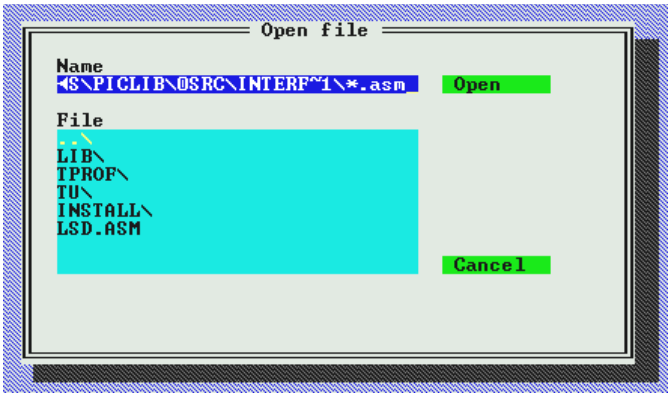


Рис. 2.3. Файловый диалог «Open File»

Open(F3) – вызывает файловый диалог «Open file», в котором следует выбрать файл для загрузки в редактор.

Save(F2) – сохранить файл из активного окна редактора.

Save as - вызывает файловый диалог «Save file as» в котором следует выбрать имя файла под которым будет сохранен файл из активного окна редактора. Окно файлового диалога аналогично окну «Open file».

Exit(Alt-X) – выход из программы. При нажатии на Alt-X в любой ситуации происходит выход из программы.

2.3.2. Принцп построения меню компиляции

Рис. 2.4. Меню компиляции

При проектировании среды были реализованы следующие функции для работы с исполняемыми последовательностями машинных кодов: преобразование исходного текста программы на языке ассемблера в исполняемую последовательность машинных кодов (ассемблирование), и преобразование исполняемой последовательности машинных кодов в исходный текст программы на языке ассемблера (дизассемблирование). Вызов обеих этих функций осуществляется из меню компиляции «Compile».

Assembly (F9) – компилирует (ассемблирует) файл из активного окна редактора.

Disassembly (Alt-F10) – дизассемблирует файл из активного окна редактора.

2.3.3. Принцип построения меню отладки

При проектировании среды разработан и включен в её состав эмулятор ядра микроконтроллера PIC16F84, на основе этого компонента реализованны следующие возможности для отладки: пошаговая трассировка исполняемого кода, пошаговая трассировка без трассировки процедур и возможность сразу перейти к выполнению необходимой команды.

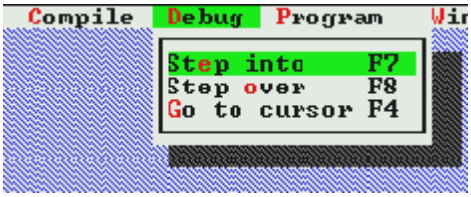


Рис. 2.5. Меню отладки

Step into (F7) – Выполнить текущую команду, либо загрузить файл из активного окна редактора в эмулятор.

Step over (F8) – Выполнить текущую команду «перескакивая» call.

Go to cursor (F4) – Выполнять команды до тех пор пока не будет достигнута команда на которой стоит курсор.

2.3.4. Принцип построения меню программации

В меню программации реализованы функции для работы с внутренней памятью микроконтроллера: запись кода в память контроллера, чтение кода из памяти контроллера и очистка памяти контроллера.

Рис. 2.6. Меню «Program»

Write (Ctrl-F9) – Записать в контроллер файл из активного окна редактора.

Verify (Alt-F9) – Сравнить программу прошитую в контроллере с программой из активного окна редактора.

Read – Прочитать программу из контроллера и записать ее в соответствующий файл.

Erase (Alt-F8) – Очистить память контроллера.

2.3.5. Принцип построения меню оконных операций

Для работы с основными элементами графического интерфейса программы – окнами реализован следующий минимальный набор функций: изменение размеров окон, изменение местоположения окон и закрытие окон. Методика использования этих функций описана ниже.

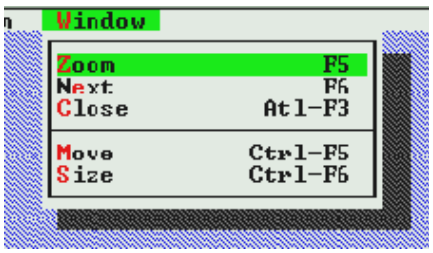


Рис. 2.7. Меню оконных операций

Zoom (F5) – Развернуть активное окно на полный экран.

Next (F6) – Сделать активным следующее окно.

Close (Alt-F3) – Закрыть активное окно.

Move (Ctrl-F5) – Перейти в режим движения активного окна. Выход из режиме – Esc. Движение осуществляется с помощью клавиш – стрелок.

Size (Ctrl-F6) – Перейти в режим изменения размера активного окна. Выход из режима – Esc. Изменение размера осуществляется с помощью клавиш – стрелок.

2.3.6. Принцип построения меню встроенной помощи

Для эффективного использования среды разработана внутренняя система помощи, с помощью которой можно получить справку по следующим разделам: команды ядра микроконтроллера, архитектура микроконтроллера и интегрированная среда разработки. . Методика использования этих функций описана ниже.

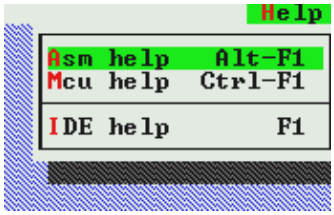


Рис. 2.8. Меню встроенной помощи

ASM Help (Alt-F1) – Вызвать помощь по системе команд процессора.

MCU Help (Ctrl-F1) – Вызвать помощь по архитектуре процессора.

IDE Help (F1) – Вызвать помощь по среде разработки.

2.4. Редактор

Встроенный в среду разработки редактор является основным средством для создания, просмотра и редактирования исходных текстов программ для последующей работы с ними. Открытие файла в редакторе осуществляется из файлового диалога “Open file”, который вызывается из меню или при нажатии клавиши F3. Навигация в файловом диалоге осуществляется при помощи следующих клавиш: Tab – переключение между компонентами, клавиши-стрелки – навигация в компонентах. Сохранение также осуществляется из меню или по нажатию клавиши F2, при необходимости текущий файл можно сохранить под другим именем. Навигация по тексту осуществляется при клавиш-стрелок и клавиш PageUp, PageDown, Home и End. В настоящей версии не поддерживается работа с блоками.

2.5. Обоснование выбора лингвистического обеспечения

В качестве лингвистического обеспечения был выбран язык ассемблера. Этот язык прост в изучении, и вместе с тем он в ряде случаев может быть использован для разработки сложных программных продуктов, особенно системного программного обеспечения.

Ассемблер – компилятор, преобразующий исходный код на языке ассемблера в последовательность машинных кодов исполняемых ядром контроллера. Синтаксис, используемый во встроенном ассемблере совместим с синтаксисом рекомендованным фирмой Microchip. Рассмотрение мнемоник команд микроконтроллеров семейства PIC достаточно объемно и выходит за рамки данного руководства. Для подробного изучения мнемоник команд и регистров процессора можно обратиться к документации фирмы Microchip к соответствующему процессору.

При компиляции на вход подается файл с расширением ASM, а после обработки формируются файлы с расширениями ERR и HEX. Файл с расширением ERR содержит описание ошибок, произошедших во время компиляции. Если ошибок не было, то формируется файл с расширением HEX, содержащий последовательность машинных кодов, готовую для загрузки в PIC.

2.5.1. Директивы компилятора

Для упрощения программ, программист может пользоваться следующими директивами компилятора: include и equ. Директива include используется для подключения ещё одного файла исходного текста к основному, и имеет синтаксис “include filename”. Директива equ используется для замены часто встречаемых числовых значений на символьные имена, и имеет синтаксис “символьное_имя equ числовое_значение”

2.6. Дизассемблер

Используемый в среде дизассемблер – средство для изучения программ, считанных из микроконтроллера. В качестве входного используется файл в формате Intel 16 Hex. После дизассемблирования получается файл с расширением ASM содержащий исходный

текст программы. Полученный после дизассемблирования файл может быть использован для повторной компиляции.

2.7. Эмулятор

Эмулятор – основной отладочный модуль, имитирующий исполнение команд ядром микроконтроллера. Эмулятор состоит из 2 частей: программы эмуляции и кода управления. Программа эмуляции принимает на вход данные о состоянии ядра контроллера и команду, исполняемую в данный момент, и возвращает данные о новом состоянии ядра после обработки входных данных. Код управления является частью интегрированной среды. Он визуализирует и изменяет обрабатываемые программой эмуляции данные. Также этот код формирует и обрабатывает окно отладчика (см. Рис. 3.9.)

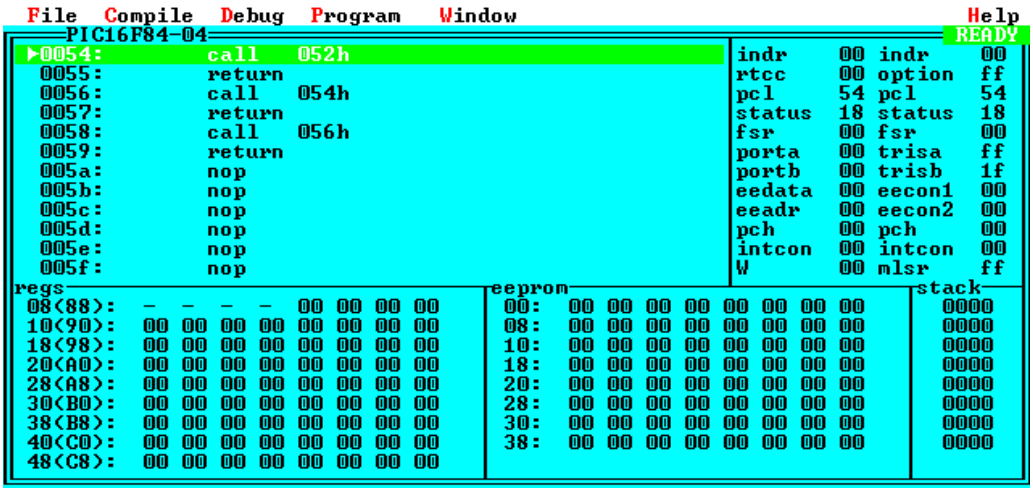


Рис. 2.9. Окно отладчика.

Окно отладчика отображает следующие группы данных: группа кода, регистры, внутреннюю энергонезависимую память EEPROM и стек. Все цифры в группах данных представлены в шестнадцатеричной системе исчисления.

Навигация в группах данных осуществляется при помощи клавиш-стрелок и клавиши Tab. Также возможно изменять данный вводя новые значения.

Пошаговая отладка осуществляется при нажатии клавиши F7 или при выборе соответствующего пункта меню. Также возможна отладка без трассировки процедур при нажатии F8. При нажатии F7 во время редактирования исходного текста программы происходит компиляция и загрузка этой программы в отладчик.

2.8. Программатор

Программатор – аппаратное устройство взаимодействия между компьютером и микроконтроллером, а также программа управления этим устройствам. Программатор используется для считывания и записи программы во внутреннюю память микроконтроллера. В среде используется программатор, имеющий простое устройство и соединяющийся с компьютером через COM-порт. При записи программы на вход программатору подается файл в формате Intel 16 Hex, а также данные необходимые для настройки контроллера. При чтении на выходе также получается файл в формате Intel 16 Hex.

3. ЗАКЛЮЧЕНИЕ

Эта программа может быть использована для изучения простейших RISC-процессоров. Она содержит все компоненты необходимые при разработке и отладке приложений для процессора PIC16F84. Но вместе с тем он проста, так что даже не опытному в этой сфере человеку будет не сложно с ней работать. В будущем планируется расширить возможности программы.

1. Добавить к ней компоненты разработки приложений для других процессоров.
2. Доработать интегрированную среду.

3.1. Список использованных литературных источников

1. Д. Кнут Искусство программирования. Том 1.– Москва: Вильямс, 2001.– 712 с.
2. Б. Страуструп Язык программирования C++.– Москва: БИНОМ, 2002.– 1098 с.
3. Д. Грис Проектирование компиляторов для цифровых вычислительных машин.– Москва: МИР, 1975. – 544 с.
4. Д. Бэнтли Жемчужины программирования.– Санкт-Петербург: Питер, 2002.– 268 с.
5. PIC16F8X Reference Manual. – Microchip Technology Inc., 1998.– 124 с.

ПРИЛОЖЕНИЕ.
ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММ

File Asm\Asm.cpp

```
//---------------------------------------------------------------------------
//Pic16XXXX Asm  v0.90alpha                                          (c)Pit
//---------------------------------------------------------------------------

#include <stdio.h>
#include <ctype.h>
#include <io.h>
#include <FCNTL.H>
typedef char* lexime;

const int MAX_EQU      = 100;
const int MAX_INCLUDE  = 10;
const int MAX_LEX      = 100;
const int MAX_FILENAME = 12*10;
const int MAX_EQUNAME  = 20;
const int MAX_EQUVALUE = 30;
const int MAX_CODE     = 10000;
const int MAX_LABLES   = 100;
//const int MAX_MACRO  = 10;

unsigned char*       NowPointer = NULL;
char*                Lex        = new char [MAX_LEX];
static unsigned char Simbol     = ' ';
static unsigned int  Vaule      = 0;
int                  Db_flag    = 0;
int                  pass       = 0;

struct {unsigned int vaule; char* name;}    EquList   [MAX_EQU];
struct {unsigned int adress; lexime lable;} LableList [MAX_LABLES];
char* IncludeList [MAX_INCLUDE];
char* IncludeNameList [MAX_INCLUDE];
struct {char* ptr; int inc_num; int str_count;} ZeroList [MAX_INCLUDE];
//struct {char* start; int count;} MacroList [MAX_MACRO];
int EquCount       = 0,
    ZeroCount      = 0,
    IncludeCount   = 0,
    LableCount     = 0,
    ErrorsCount    = 0,
    CountFlag      = 0,
    StrCount       = 0,
    CurrentInclude = 0;
    //MacroCount = 0;

unsigned int CodeCount = 0;
unsigned int* CodeA    = new unsigned int [MAX_CODE];
short int EOfF         = 0;
FILE* OutFile          = 0;
char* basefilea         = 0;

static char* Names [39] =
    {
    "addlw",
    "addwf",
    "andlw",
    "andwf",
    "bcf",
    "bsf",
    "btfsc",
    "btfss",
    "call",
    "clrf",
    "clrw",
    "clrwdt",
    "comf",
    "decf",
    "decfsz",
```

```
    "goto",
    "incf",
    "incfsz",
    "iorlw",
    "iorwf",
    "movlw",
    "movf",
    "movwf",
    "nop",
    "option",
    "retfie",
    "retlw",
    "return",
    "rlf",
    "rrf",
    "sleep",
    "sublw",
    "subwf",
    "swapf",
    "tris",
    "xorlw",
    "xorwf",
    "trisa",
    "trisb"
    };

static unsigned int Codes [39] [2] =
    {
    {0x3e00,2},
    {0x0700,1},
    {0x3900,2},
    {0x0500,1},
    {0x1000,4},
    {0x1400,4},
    {0x1800,4},
    {0x1c00,4},
    {0x2000,3},
    {0x0100,1},
    {0x0103,0},
    {0x0064,0},
    {0x0900,1},
    {0x0300,1},
    {0x0b00,1},
    {0x2800,3},
    {0x0a00,1},
    {0x0f00,1},
    {0x3800,2},
    {0x0400,1},
    {0x3000,2},
    {0x0800,1},
    {0x0080,1},
    {0x0000,0},
    {0x0062,0},
    {0x0009,0},
    {0x3400,2},
    {0x0008,0},
    {0x0d00,1},
    {0x0c00,1},
    {0x0063,0},
    {0x3c00,2},
    {0x0200,1},
    {0x0e00,1},
    {0x0060,1},  //???
    {0x3a00,2},
    {0x0600,1},
    {0x0065,0},
    {0x0066,0}
    };

//---------------------------------------------------------------------------
//Function definitions
//---------------------------------------------------------------------------
int           Isspace      (char);
int       ismath      (char);
void          NextLine    ();
unsigned char GetNextSim  ();
int           LexLgh      (lexime);
int           GetLex      ();
int           GetNum      ();
void          ClearStr    ();
void          ZeroDo      ();
int           EquLex      (lexime,lexime);
void          LexCpy      (lexime,lexime);
void          StartLine   ();

//void        DoMacro     (char*);          //todo
//void        DoMacroCode (char*,int);      //todo
```

```
void          DoInclude   ();
int           DoEqu       ();
int       DoCount     ();
void          DoLable     ();
void          DoOpcode    ();
void          Error       (char*);
void          Asem        (char*);
//---------------------------------------------------------------------------

void main (int argc, char* argv [])
    {
    printf ("PIC16x84x Assembler v 0.5alpha\n");
    if (argc == 1)
        {
        printf ("Using asm.exe [filename]\n\n");
        return;
        }
    Asem (argv[1]);
    printf ("\nErrors - %d\n\n",ErrorsCount);
    fprintf (OutFile,"Errors - %d\n",ErrorsCount);

    char* codefilename = new char [MAX_FILENAME];
    for (int i = 0; i < MAX_FILENAME - 3; i++)
        {
        codefilename[i] = argv[1][i];
        if (argv[1][i] == '.' || argv[1][i] == 0) break;
        }
    if (codefilename[i] == 0) codefilename[i] = '.';
    codefilename[i + 1] = 'h';
    codefilename[i + 2] = 'e';
    codefilename[i + 3] = 'x';
    codefilename[i + 4] = 0;
    remove(codefilename);
    if(ErrorsCount == 0)
        {
        FILE* codefile;
        codefile = fopen (codefilename,"w+");

        int check = 0;
        for (i = 0; i < CodeCount; i += 8)
            {
            check = 0;
            int num = 0x10;
            if ((i + 8) >= CodeCount) num = 2*(CodeCount - i/* + 1*/);
            fprintf (codefile, ":%02X%04X00", num, (i*2));
            check += num + ((i*2) & 0xff) + (((i*2) >> 8) & 0xff);
            for (int j = 0; j < 8 && (i+j) < CodeCount; j++)
                {
                fprintf (codefile, "%02X%02X", CodeA[i + j]&0xff, (CodeA[i + j]>>8)&0xff);
                check += (*(CodeA + i + j)&0xff) + ((*(CodeA + i + j)>>8)&0xff);
                }
            fprintf (codefile, "%02X\n", (0x100 - (check & 0xff)) & 0xff);
            }
        fprintf (codefile, ":00000001FF\n");
        delete codefilename;
        fclose (codefile);
        }
    fclose (OutFile);
    }

//---------------------------------------------------------------------------
//Lexical Analizator
//---------------------------------------------------------------------------
int Isspace (char sim)
    {
    if (sim == '\n') return 0;
    else if (isspace(sim)) return 1;
        else return 0;
    }

int ismath (char sim)
    {
    if (sim == '+' || sim == '-' || sim == '/' ||
        sim == '*' || sim == '|' || sim == '&') return 1;
    return 0;
    }

void NextLine ()
    {
    if (Simbol == 0x00 || Simbol == (char)EOF) return;
    while (*(NowPointer) != '\n' && *(NowPointer) != (char)EOF && *(NowPointer) != 0x00) Now-
    Pointer++;
    NowPointer++;
    Simbol = *(NowPointer - 1);
    }

unsigned char GetNextSim ()
```

```
    {
    NowPointer++;
    return *(NowPointer - 1);
    }

int LexLgh (leximе lex)
    {
    int i = 0;
    while (*(lex + i) != '\0') i++;
    return i;
    }

int GetLex ()
    {
    int i = 0;
    while (Isspace(Simbol)) Simbol = GetNextSim();
    if (Simbol == ';')
        {
        NextLine();
        }
    if (isalpha(Simbol) || Simbol == '_')
        {
        while (i < MAX_LEX && isalnum(Simbol) ||
               Simbol == '_' || Simbol == '.' || Simbol == ':')
            {
            *(Lex + (i++)) = Simbol;
            Simbol = GetNextSim();
            if (Simbol == (char)EOF)
                {
                EOfF = 1;
                break;
                }
            }
        if (Simbol == (char)EOF) EOfF = 1;
        *(Lex + i) = '\0';
        for (int j = 0; j < EquCount; j++)
            {
            if (EquLex(Lex,EquList[j].name))
                {
                Vaule  = EquList[j].vaule;
                *(Lex) = '\0';
                if (CountFlag == 0)
                    {
                    NowPointer -= (i + 1);
                    Simbol = GetNextSim();
                    if (DoCount()) return 1;
                    return 0;
                    }
                else return 1;
                }
            }
        for (j = 0; j < LableCount; j++)
            {
            if (EquLex(Lex,LableList[j].lable))
                {
                Vaule = LableList[j].adress;
                *Lex = '\0';
                if (CountFlag == 0)
                    {
                    NowPointer -= (i + 1);
                    Simbol = GetNextSim();
                    if (DoCount()) return 1;
                    return 0;
                    }
                else return 1;
                }
            }
        return 1;
        }

    if (isdigit(Simbol))
        {
        if (DoCount()) return 1;
        NextLine();
        return 0;
        }

    if (Simbol == '"')
        {
        while (Simbol != '"' || i < MAX_LEX || Simbol != (char)EOF)
            {
            *(Lex + (i++)) = Simbol;
            Simbol = GetNextSim();
            }
        if (Simbol != '"' && Simbol != (char)EOF)
            {
            Error ("String to long");
```

```
            NextLine();
            return 0;
            }
        *(Lex + i) = '\0';
        if (Simbol == (char)EOF)
            {
            EOfF = 1;
            return 1;
            }
        GetNextSim();
        return 1;
        }

    if (Simbol == '\'')
        {
        Vaule = GetNextSim();
        *(Lex) = '\0';
        GetNextSim();
        if (Simbol != '\'') goto error;
        GetNextSim();
        return 1;
        }

    if (ismath(Simbol) || Simbol == ')' || Simbol == '(' || Simbol == ',')
        {
        *(Lex) = Simbol;
        *(Lex + 1) = '\0';
        Simbol = GetNextSim();
        return 1;
        }

    if (Simbol == (char)EOF)
        {
        EOfF = 1;
        return 1;
        }

    if (Simbol == 0x00)
        {
        ZeroDo();
        Simbol = GetNextSim();
        if(GetLex()) return 1;
        }

    if (Simbol == '\n')
        {
        *(Lex)   = '\n';
        *(Lex+1) = '\0';
        Simbol   = GetNextSim();
        StrCount++;
        return 1;
        }

    if (Simbol == '$')
        {
        *(Lex) = '\0';
        Vaule  = CodeCount;
        Simbol = GetNextSim();
        return 1;
        }

error:  Error ("Incorrect indeficator");
    NextLine();
    return 0;
    }

int GetNum ()
    {
    int i = 0;
    while (isspace(Simbol)) Simbol = GetNextSim();
    if (!isdigit(Simbol))
        {
        if (GetLex())
            {
            if (*(Lex) != '\0')
                {
                Error ("Incorrect statment");
                return 0;
                }
            return 1;
            }
        else return 0;
        }
    while (isxdigit(Simbol) || Simbol == 'h' || Simbol == 'H')
            {
            *(Lex + (i++)) = Simbol;
            Simbol = GetNextSim();
```

```
            }
    *(Lex + i) = '\0';
    if (!ismath(Simbol) && !isspace(Simbol) && Simbol != (char)EOF && Simbol != '\n' && Simbol
    != 0x00 && Simbol != ',') return 0;
    unsigned int num = 0;
    switch (*(Lex + i - 1))
        {
        case 'b': case 'B':
            char n = 0;
            for (int j = (i - 2); j >= 0; j--)
                {
                n >>= 1;
                n += (*(Lex + j) - '0') * 0x80;
                }
            num = n;break;
        case 'h': case 'H': for (i = 0; i < LexLgh(Lex) - 1; i++)
            {
            num *= 0x10;
            switch (*(Lex + i))
                {
                case 'a': case 'A': num += 0x0a;break;
                case 'b': case 'B': num += 0x0b;break;
                case 'c': case 'C': num += 0x0c;break;
                case 'd': case 'D': num += 0x0d;break;
                case 'e': case 'E': num += 0x0e;break;
                case 'f': case 'F': num += 0x0f;break;
                default : num += *(Lex + i) - '0';break;
                }
            }
            break;
        case 'd': case 'D': default:
            for (j = 0; j <= i - 1; j++) {num *= 10;num += *(Lex + j) - '0';} break;
        }
    Vaule  = num;
    Lex[0] = 0;
    return 1;
    }

void ClearStr ()
    {
    int j = 0;
    while (*(NowPointer + j) != '\n' && *(NowPointer + j) != 0x00 && *(NowPointer + j) !=
    (char)EOF) {*(NowPointer + j) = ' ';j++;}
    j = 0;
    while (*(NowPointer - j) != '\n' && (NowPointer-(j-1)) != basefilea) {*(NowPointer - j) = '
    ';j++;}
    NowPointer -= (j-2);
    Simbol = *(NowPointer);
    NowPointer++;
    }

void ZeroDo ()
    {
    switch (*(NowPointer))
        {
        case 'i': if (IncludeList [*(NowPointer + 1)] != NULL)
                    {
                    ZeroList[ZeroCount].ptr       = NowPointer + 2;
                    ZeroList[ZeroCount].inc_num   = CurrentInclude;
                    ZeroList[ZeroCount].str_count = StrCount;
                    CurrentInclude = NowPointer[1] + 1;
                    StrCount = 1;
                    ZeroCount++;
                    NowPointer = IncludeList[*(NowPointer + 1)];
                    break;
                    }
                  else
                    {
                    NowPointer += 3;
                    break;
                    }
        case 'e': ZeroCount--;
                  NowPointer     = ZeroList[ZeroCount].ptr;
                  CurrentInclude = ZeroList[ZeroCount].inc_num;
                  StrCount       = ZeroList[ZeroCount].str_count;
                  break;
        default : NowPointer++;break;
        }
    }

int EquLex (lexime l1,lexime l2)
    {
    int i = 0;
    while (*(l1+i) == *(l2+i) && *(l1+i) != '\0') i++;
    if (*(l1+i) == *(l2+i) && *(l1+i) == '\0') return 1;
    return 0;
    }
```

```
void LexCpy (char* l1,char* l2)
    {
    int i = 0;
    *(l2) = *(l1);
    while (*(l1 + i) != '\0')
        {
        i++;
        *(l2 + i) = *(l1 + i);
        }
    }

void StartLine()
    {
    int i = 3;
    while (*(NowPointer - i) != '\n' && (NowPointer-(i-1)) != basefilea) i++;
    NowPointer -= (i--);
    NowPointer++;
    Simbol = GetNextSim();
    }
//=========================================================================
void DoInclude ()
    {
    GetLex ();
    NowPointer--;
    StartLine();
    ClearStr();
    *(NowPointer++) = 0x00;
    *(NowPointer++) = 'i';
    *(NowPointer++) = (unsigned char)IncludeCount;
    FILE* incfile;
    if ((incfile = fopen (Lex,"r")) == NULL)
        {
        Error ("Can not open file");
        IncludeList [IncludeCount] = NULL;
        IncludeCount++;
        return;
        }
    int i = 0;
    while (!feof(incfile))
        {
        char c = fgetc(incfile);
        i++;
        }
    char* incfilea = new char [i+2];
    i = 0;
    fseek(incfile, 0, SEEK_SET);
    while (!feof(incfile))
        {
        *(incfilea + i) = fgetc(incfile);
        i++;
        }
    i--;
    *(incfilea + i) = 0x00;
    i++;
    *(incfilea + i) = 'e';
    IncludeNameList[IncludeCount+1] = new char[MAX_LEX];
    LexCpy(Lex,IncludeNameList[IncludeCount+1]);
    IncludeList [IncludeCount] = incfilea;
    IncludeCount++;
    fclose (incfile);
    }

int DoEqu ()
    {
    StartLine();
    if (!GetLex()) return 0;
    EquList [EquCount].name = new char [MAX_LEX];
    LexCpy (Lex,EquList [EquCount].name);
    GetLex();
    if (!EquLex(Lex,"equ"))
        {
        Error ("Sintacsic error");
        EquList [EquCount].name = "\0";
        NextLine();
        return 0;
        }
    if (!GetLex() || !EquLex(Lex,"\0"))
        {
        EquList [EquCount].name = "";
        return 0;
        }
    EquList [EquCount].vaule = Vaule;
    StartLine();
    ClearStr();
    EquCount++;
    return 1;
```

```
    }

int DoCount ()
    {
    CountFlag = 1;
    int op1   = 0,
        op2   = 0;
    char oper = 0;
    GetNum();
    op1 = Vaule;
    if (!EquLex(Lex,"\0"))
            {
            Error ("Incorrect staetment");
            NextLine();
            CountFlag = 0;
            return 0;
            }
    while (!EquLex(Lex,"\n") && !EquLex(Lex,",") && Simbol != '\n' && Simbol != (char)EOF &&
    Simbol != 0x00 && Simbol != ';')
        {
        while (Isspace(Simbol)) Simbol = GetNextSim();
        if (EquLex(Lex,"\n") || Simbol == 0x00 || Simbol == '\n' || Simbol == (char)EOF || Sim-
    bol == ',' || Simbol == ';') break;
        GetLex();
        if (EquLex(Lex,"\n") || EquLex(Lex,",") || Simbol == '\n' || Simbol == (char)EOF ||
    Simbol == 0x00 || Simbol == ';') break;
        if (!ismath(*(Lex)))
            {
            Error ("Sintacsic error");
            NextLine();
            CountFlag = 0;
            return 0;
            }
        oper = *(Lex);
        if (*(Lex + 1) != '\0')
            {
            Error ("Sintacsic error");
            NextLine();
            CountFlag = 0;
            return 0;
            }
        GetNum();
        if (!EquLex(Lex,"\0"))
            {
            Error ("Incorrect staetment");
            NextLine();
            CountFlag = 0;
            return 0;
            }
        op2 = Vaule;
        switch (oper)
            {
            case '+':op1 += op2;break;
            case '-':op1 -= op2;break;
            case '*':op1 *= op2;break;
            case '/':op1 /= op2;break;
            }
        GetLex();
        }
    Vaule = op1;
    *Lex = '\0';
    CountFlag = 0;
    return 1;
    }

void DoLable ()
    {
    LableList [LableCount].lable = new char [MAX_LEX];
    LexCpy (Lex,LableList[LableCount].lable);
    LableList[LableCount].adress = 0;
    LableCount++;
    }

void DoOpcode ()
    {
    unsigned int code = 0;
    if (EquLex(Lex,"db"))
        {
        while (!EquLex(Lex,"\n"))
            {
            if(!GetLex()) return;
            if(!EquLex(Lex,"\0"))
                {
                Error ("Constant waiting");
                return;
                }
            if(Vaule > 0xff)
```

```
                {
                Error ("Constant out of range");
                return;
                }

            if (Db_flag)
                {
                Db_flag = 0;
                *(CodeA + CodeCount) += Vaule * 0x100;
                CodeCount++;
                }
            else
                {
                Db_flag = 1;
                *(CodeA + CodeCount) = Vaule;
                }
            if(!GetLex()) return;
            if (!EquLex(Lex,"\n") && !EquLex(Lex,",")) return;
            }
        return;
        }
    else
        {
        if (Db_flag)
            {
            Db_flag = 0;
            CodeCount++;
            }
        }

    if (EquLex(Lex,"end"))
        {
        GetLex();
        return;
        }

    if (EquLex(Lex,"dw"))
        {
        while (!EquLex(Lex,"\n"))
            {
            if(!GetLex()) return;
            if(!EquLex(Lex,"\0"))
                {
                Error ("Constant waiting");
                return;
                }
            if(Vaule > 0xffff)
                {
                Error ("Constant out of range");
                return;
                }
            unsigned int temp = Vaule;
            temp >>= 8;
            temp += Vaule << 8;
            CodeA[CodeCount] = temp;
            CodeCount++;
            if(!GetLex()) return;
            if (!EquLex(Lex,"\n") && !EquLex(Lex,",")) return;
            }
        return;
        }
    if (EquLex(Lex,"org"))
        {
        if(!GetLex()) return;
        if(!EquLex(Lex,"\0"))
            {
            Error ("Adress waiting");
            return;
            }
        if (Vaule < CodeCount)
            {
            Error ("Adress bigger then current");
            return;
            }
        for (int i = CodeCount; i <= (Vaule/2); i++) *(CodeA + i) = 0x0000;
        CodeCount = Vaule/2;
        if(Vaule%2) Db_flag = 1;
        return;
        }
    for (int i = 0; i < 39; i++) if (EquLex(Lex,Names[i])) break;
    if (i == 39)
        {
        Error ("Unknown command");
        NextLine();
        return;
        }
// 0 - none
```

```
// 1 - register
// 2 - const
// 3 - goto,call
// 4 - bit
     switch (Codes[i][1])
          {
          case 0: if(!GetLex()) return;
               if(!EquLex(Lex,"\n") && EOfF != 1)
                    {
                    Error ("Sintacsic error");
                    return;
                    }
               code = Codes [i][0];
               break;

          case 1: if(!GetLex()) return;
               if(!EquLex(Lex,"\0"))
                    {
                    Error ("Register waiting");
                    return;
                    }
               if(Vaule > 0x7f)
                    {
                    Error ("Register out of range");
                    return;
                    }
               code = (Codes[i][0] | Vaule);
               if(!GetLex()) return;
               if(EquLex(Lex,"\n") || EOfF == 1) break;
               if(EquLex(Lex,","))
                    {
                    if(!GetLex()) return;
                    Vaule = 0;
                    if(EquLex(Lex,"d") || EquLex(Lex,"f") ||
                       EquLex(Lex,"D") || EquLex(Lex,"F") ||
                       Vaule == 1)
                         {
                         code |= 0x80;
                         }
                    if(!GetLex()) return;
                    if(EquLex(Lex,"\n") || EOfF == 1) break;
                    else
                         {
                         Error ("Sintacsic error");
                         return;
                         }
                    }
               else
                    {
                    Error ("Sintacsic error");
                    return;
                    }

          case 2: if(!GetLex()) return;
               if(!EquLex(Lex,"\0"))
                    {
                    Error ("Constant waiting");
                    return;
                    }
               if(Vaule > 0xff)
                    {
                    Error ("Constant out of range");
                    return;
                    }
               code = (Codes[i][0] | Vaule);
               if(!GetLex()) return;
               if(EquLex(Lex,"\n") || EOfF == 1) break;
               else
                    {
                    Error ("Sintacsic error");
                    return;
                    }

          case 3: if(!GetLex()) return;
               if(!EquLex(Lex,"\0"))
                    {
                    Error ("Lable waiting");
                    return;
                    }
               if(Vaule > 0x3ff)
                    {
                    Error ("Label out of range");
                    return;
                    }
               code = (Codes[i][0] | Vaule);
               if(!GetLex()) return;
               if(EquLex(Lex,"\n") || EOfF == 1) break;
```

```
            else
                {
                Error ("Sintacsic error");
                return;
                }

        case 4: if(!GetLex()) return;
            if(!EquLex(Lex,"\0"))
                {
                Error ("Register waiting");
                return;
                }
            if(Vaule > 0x7f)
                {
                Error ("Register out of range");
                return;
                }
            code = (Codes[i][0] | Vaule);
            if(!GetLex()) return;
            if(EquLex(Lex,","))
                {
                if(!GetLex()) return;
                if(!EquLex(Lex,"\0"))
                    {
                    Error ("Bit waiting");
                    return;
                    }
                if(Vaule > 0x08)
                    {
                    Error ("Bit out of range");
                    return;
                    }
                code |= (Vaule << 7);
                if(!GetLex()) return;
                if(EquLex(Lex,"\n") || EOfF == 1) break;
                else
                    {
                    Error ("Sintacsic error");
                    return;
                    }
                }
            else
                {
                Error ("Sintacsic error");
                return;
                }

        }
    *(CodeA + CodeCount) = code;
    CodeCount++;
    }

void Error (char* str)
    {
    if (pass != 3 && pass != 0) return;
    ErrorsCount++;
    fprintf(OutFile,"Error %s(%d): %s.\n", IncludeNameList[CurrentInclude], StrCount, str);
    NextLine();  //!!!
    }

//=========================================================================
void Asem (char* start)
    {
    char* outfilename = new char [MAX_FILENAME];
    for (int i = 0; i < MAX_FILENAME - 3; i++)
        {
        outfilename[i] = start[i];
        if (start[i] == '.' || start[i] == 0) break;
        }
    if (outfilename[i] == 0) outfilename[i] = '.';
    outfilename[i + 1] = 'e';
    outfilename[i + 2] = outfilename[i + 3] = 'r';
    outfilename[i + 4] = 0;
    OutFile = fopen (outfilename,"w");
    delete outfilename;
    FILE* basefile;
    i = 0;
    while (start[i] != 0 && start[i] != '.') i++;
    if (start[i] == 0)
        {
        start[i]   = '.';
        start[i+1] = 'a';
        start[i+2] = 's';
        start[i+3] = 'm';
        start[i+4] = 0;
        }
    IncludeNameList[0] = new char [9];
```

```
    LexCpy(start,IncludeNameList[0]);
    if ((basefile = fopen (start,"r")) == NULL)
        {
        Error ("Can not open file");
        return;
        }
    i = 0;
    while (!feof(basefile))
        {
        char c = fgetc(basefile);
        i++;
        }
    fclose (basefile);
    basefile = fopen (start,"r");
    basefilea = new char [i+2];
    i = 0;
    while (!feof(basefile))
        {
        *(basefilea + i) = fgetc(basefile);
        i++;
        }
    NowPointer = basefilea;
    EOfF = 0;

//FirstPass (equs list,lables list,includes list)
    while (EOfF != 1)
        {
        if (GetLex())
            {
            if (EquLex(Lex,"include"))
                {
                DoInclude();
                NowPointer -= 3;
                }
            if (EquLex(Lex,"equ"))
                {
                DoEqu();
                }
            if (*(Lex + LexLgh(Lex) - 1) == ':')
                {
                *(Lex + LexLgh(Lex) - 1) = '\0';
                DoLable();
                }

            }
        }

//NextPasses (Compiletion)
    for (pass = 1; pass < 4; pass++)
        {
        NowPointer = basefilea;
        Simbol = GetNextSim();
        StrCount = 1;
        EOfF = 0;
        CodeCount = 0;
        while (EOfF != 1)
            {
            if (GetLex() && *(Lex + LexLgh(Lex) - 1) != ':' && *(Lex) != '\n')
                {
                DoOpcode();
                }
            if (*(Lex + LexLgh(Lex) - 1) == ':')
                {
                *(Lex + LexLgh(Lex) - 1) = '\0';
                for (int j = 0; j < LableCount; j++)
                    {
                    if (EquLex(Lex,LableList[j].lable))
                        {
                        LableList[j].adress = CodeCount;
                        }
                    }
                }
            }
        if (Db_flag) Db_flag = 0;
        }
    if (Db_flag) CodeCount++;
    fclose (basefile);
    }
```

File Core\CDraw.h

```
#ifndef __CDRAW_H__
#define __CDRAW_H__
```

```
//===CONSTANTS========================================================================
#define HOR  0
#define VERT 1

//===DEFINITION=======================================================================
class CDraw
{

//---INTERFACE---
public:
CDraw            (CScreen* scr);
void line        (int, int, int, int);
void Frame       (int, int, int, int, int, int, int, int, int, int);
void Button      (int, int, char*, int, int, int);
void FrameButton (int, int, char*, int, int, int, int);
void DeskDraw    ();
void Scroll      (int, int, int, int, int, int);

//---ATRIBUTES---
private:
CScreen* scr_c;

};

//--REALISATION---
CDraw::CDraw (CScreen* scr = NULL)
    { scr_c = scr; }

void CDraw::line (int x, int y, int count, int options = 0)
    {
    char simbols [ 4 ] = {'‹','œ','''','¥'};
    char correct_sim = simbols [options];
    for (int c = 0; c <= count; c++)
        {
        scr_c->Put (options % 2? x+c:x, options % 2? y:y+c ,correct_sim);
        }
    }

void CDraw::Frame (int x_start,     int y_start,
                int x_end,       int y_end,
                int double_ = 0, int shadow_ = 0,
                int frame_color_t = 0, int frame_color_b = 0, int frame_color_s = 0)

    {
    scr_c->TextColor (frame_color_t);
    scr_c->BackColor (frame_color_b);
    scr_c->Put (x_start, y_start, double_? '¡':'²');
    scr_c->Put (x_start, y_end,   double_? ' ':'˜');
    scr_c->Put (x_end, y_start,   double_? '“':'–');
    scr_c->Put (x_end, y_end,     double_? '”':'±');
    line (x_start+1, y_start,   x_end - x_start - 2, double_ * 2 + 1);
    line (x_start+1, y_end,     x_end - x_start - 2, double_ * 2 + 1);
    line (x_start,   y_start+1, y_end - y_start - 2, double_ * 2);
    line (x_end,     y_start+1, y_end - y_start - 2, double_ * 2);
//  frame_color_s += double_;
    if (shadow_)
        {
        int col = scr_c->GetTextColor();
        scr_c->TextColor (8);
        scr_c->BackColor (0);
        for (int i = x_start+1; i <= x_end+2; i++)
            {
            if (i < SCR_ROW) scr_c->SetColor (i, y_end+1);
            }
        for (i = y_start+1; i < y_end+1; i++)
            {
            if (x_end+1 < SCR_ROW) scr_c->SetColor (x_end+1, i);
            if (x_end+1 < SCR_ROW) scr_c->SetColor (x_end+2, i);
            }
        scr_c->TextColor (col);
        }
    }

void CDraw::Button (int x = 0,           int y = 0,
                    char* string = NULL, int count = 0,
                    int base_color = 8,  int spec_color = 1)
    {
    scr_c->TextColor (base_color);
    for (int i = 0; i < count; i++)
        {
        if (*string == '~')
            {
            string++;
            scr_c->TextColor (spec_color);
            scr_c->Put (x+i, y, *string);
            scr_c->TextColor (base_color);
```

```
            string++;
            }
        else
            {
            scr_c->Put (x+i, y, *string);
            string++;
            }
        }
    }

void CDraw::FrameButton (int x = 0,           int y = 0,
                         char* string = NULL, int count = 0, int double_ = 0,
                         int base_color = 8,  int spec_color = 1)
    {
    Frame  (x,   y,   x+count+1, y+2,   double_, 0, base_color);
    Button (x+1, y+1, string,  count, base_color, spec_color);
    }

void CDraw::DeskDraw()
    {
    Scr.TextColor (1);            //!!!
    Scr.BackColor (7);                //!!!
    for (int i = 0; i < SCR_ROW; i++)
        for (int j = 1; j < SCR_STRING; j++)
            Scr.Put (i, j, '^');    //177
    }

void CDraw::Scroll (int x_start, int y_start, int count, int dir, int max, int current)
    {
    Scr.Put (x_start, y_start, dir == HOR ? '' : '-');
    Scr.Put (dir == HOR ? x_start + count : x_start,
         dir == HOR ? y_start : y_start + count,
         dir == HOR ? '' : '' );
    for (int i = 1; i < count; i++)
        {
        Scr.Put (dir == HOR ? x_start + i : x_start,
             dir == HOR ? y_start : y_start + i, '');
        }

    float place = 0;

    if (max != 0)
        {
        place = (current*(count-2)/max);
        }
    if (place >= (count - 2) || current >= max) place = (count - 2);

    Scr.Put (dir == HOR ? x_start + place+1 : x_start,
         dir == HOR ? y_start : y_start + place+1, '');
    }
//EOF
#endif __CDRAW_H__
```

File Core\CFSwap.h

```
#ifndef __CFSWAP_H__
#define __CFSWAP_H__

//===INCLUDES=============================================================
#include <assert.h>
#include "CSwapMng.h"

#define NULL 0
#define TRUE 1
#define FALSE 0
#define MAX_LOADED 100
//===DEFINITION===========================================================
struct element_f_c
    {
    int  index;
    void far *offset;
    };

template <class T> class CFSwap
{

//---INTERFACE---
public:
      CFSwap  ();
      CFSwap      (int max_lgh, int max_loaded);
void  SetOffset   (T far& dist,        void far* scr,        int del = 1);
void  SetOffset1  (void far* dist, void far* scr = NULL, int del = 1);
void  SetOffset2  (int index1 = 0, int index2 = 0,       int del = 1);
```

```
void  SetOffset3  (int index1 = 0, void far* scr = NULL, int del = 1);

T far&    operator[]  (int index);
void  SwapMin     ();
      ~CFSwap      ();

//---ATRIBUTES---
private:
unsigned long  file_offset_c;
int            max_lgh_c,
               loaded_c,
               max_loaded_c,
               last_loaded_c;
unsigned int far* cach_table_c;
element_f_c  far* incach_c;
};

//--REALISATION---
template<class T> CFSwap<T>::CFSwap ()
    {
    max_lgh_c     = 1000;
    file_offset_c = SwapMng.NewSwap(max_lgh_c, sizeof(T));
    loaded_c      = 0;
    max_loaded_c  = 50;
    cach_table_c  = new unsigned int far[max_lgh_c];
    incach_c      = new element_f_c far[max_loaded_c];
    assert(cach_table_c && incach_c);
    for (int i = 0; i < max_lgh_c; i++) cach_table_c[i] = 0;
    for (i = 0; i < max_loaded_c; i++)
        {
        incach_c[i].offset = NULL;
        incach_c[i].index  = 0;
        }
    }

template<class T> CFSwap<T>::CFSwap (int max_lgh, int max_loaded)
    {
    last_loaded_c = 0;
    max_lgh_c     = max_lgh;
    file_offset_c = SwapMng.NewSwap(max_lgh_c, sizeof(T));
    loaded_c      = 0;
    max_loaded_c  = max_loaded;
    cach_table_c  = new unsigned int far[max_lgh];
    incach_c      = new element_f_c far[max_loaded];
    assert(cach_table_c && incach_c);
    for (int i = 0; i < max_lgh; i++) cach_table_c[i] = 0;
    for (i = 0; i < max_loaded; i++)
        {
        incach_c[i].offset = NULL;
        incach_c[i].index  = 0;
        }
    }

template<class T> void CFSwap<T>::SetOffset (T far& dist, void far* scr = NULL, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].offset == (void far*)&dist)
            {
            if(del)
                delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
            }
    }

template<class T> void CFSwap<T>::SetOffset3 (int index1 = 0, void far* scr, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].index == index1)
            {
            if (del) delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
            }
    }

template<class T> void CFSwap<T>::SetOffset2 (int index1 = 0, int index2 = 0, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].index == index1)
            {
            for (int j = 0; j < loaded_c; j++)
            if(incach_c[j].index == index2)
                {
                if (del) delete incach_c[i].offset;
                incach_c[i].offset = incach_c[j].offset;
                return;
```

```
                }
            }
    }

template<class T> void CFSwap<T>::SetOffset1 (void far* dist, void far* scr = NULL, int del =
    1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].offset == (void far*)dist)
            {
            if(del) delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
            }
    }

template<class T> T far& CFSwap<T>::operator[] (int index)
    {
    if (cach_table_c[index] < 0xfffe) cach_table_c[index]++;
    for (int count = 0; count < loaded_c; count++)
        {
        if(incach_c[count].index == index)
            {
            T far& ret = *((T far*)incach_c[count].offset);
            return ret;
            }
        }
    void far *temp = new far T;
    while (!temp && !loaded_c)
        {
        SwapMin();
        temp = new far T;
        }
    while (loaded_c >= max_loaded_c) SwapMin();
    assert(SwapMng.InMem(temp, 1, file_offset_c + (long)index * (long)sizeof(T), sizeof(T)));
    incach_c[loaded_c].index  = index;
    last_loaded_c             = index;
    incach_c[loaded_c].offset = temp;
    loaded_c++;
    T far& ret = *((T far*)temp);
    return ret;
    }

template<class T> CFSwap<T>::~CFSwap()
    {
    SwapMng.DelSwap(file_offset_c);
    for (int i = 0; i < loaded_c; i++)
        {
        delete incach_c[i].offset;
        }
    delete cach_table_c;
    delete incach_c;
    }

template<class T> void CFSwap<T>::SwapMin()
    {
    int min     = 0xffff;
    int min_num = 0;
    for (int i = 0; i < loaded_c; i++)
        {
        if (cach_table_c[incach_c[i].index] < min)
            {
            if (incach_c[i].index == last_loaded_c)
                continue;
            min = cach_table_c[incach_c[i].index];
            min_num = i;
            }
        }
    assert(SwapMng.Swap(incach_c[min_num].offset, 1,
        file_offset_c + (long)incach_c[min_num].index * (long)sizeof(T), sizeof(T)));
    delete incach_c[min_num].offset;
    for (i = min_num;i < loaded_c; i++) incach_c[i] = incach_c[i+1];
    loaded_c--;
    incach_c[loaded_c].offset = NULL;
    incach_c[loaded_c].index  = 0;
    }

//EOF
#endif __CFSWAP_H__
```

File Core\CKbd.h

```
#ifndef __CKBD_H__
```

```
#define __CKBD_H__

//===CONSTANTS========================================================
#define KBD_UP          372
#define KBD_DOWN        380
#define KBD_LEFT        375
#define KBD_RIGHT       377
#define KBD_ENTER        13
#define KBD_INS         382
#define KBD_HOME        371
#define KBD_END         379
#define KBD_PGUP        373
#define KBD_PGDOWN      381
#define KBD_DEL     383
#define KBD_BKSP          8
#define KBD_TAB           9
#define KBD_ESC          27
#define KBD_ALTX    345
#define KBD_ALTF3       406
#define KBD_ALTF1       404
#define KBD_ALTF10      413
#define KBD_ALTF9       412
#define KBD_ALTF8       411
#define KBD_F1           359
#define KBD_F2           360
#define KBD_F3           361
#define KBD_F4           362
#define KBD_F5           363
#define KBD_F6           364
#define KBD_F7           365
#define KBD_F8           366
#define KBD_F9           367
#define KBD_F10          368
#define KBD_SHIFTF9     392
#define KBD_CTRLF9      402
#define KBD_CTRLF1      394
#define KBD_CTRLF5      398
#define KBD_CTRLF6      399

//===DEFINITION=======================================================
class CKbd
{
//---INTERFACE---
public:
int WaitKey ();

//---IN LIFE---
protected:
int getch ();
};

//--REALISATION---
int CKbd::WaitKey ()
    {
    int c = 0;
    c = getch();
    int h = c;
    if ((h & 0x00ff) == 0 )
        {
        c = c >> 8;
        c += 300;
        }
    else c = c & 0x00ff;
    return c;
    }

int CKbd::getch ()
    {
    asm {
        mov ah,0x00
        int 0x16
        }
    return _AX;
    }
//EOF
#endif  __KBD_H__
```

File Core\Classes.cpp

```
//====================================================================
class CCancelBut : public CButton
{
CWindow* wnd_c;
```

```
public:
CCancelBut (int x_start_c, int y_start_c, CWindow* wnd, char* str) : CButton (x_start_c,
    y_start_c, 0, str, 0, 0, 14, 2, 15, 14, 2)
    {
    wnd_c = wnd;
    int f = 0;
    for (; count_c < SCR_ROW; count_c++)
        {
        if (str[count_c] == '~')  f = 1;
        if (str[count_c] == 0)    break;
        }
    if(f) count_c--;
    }

void Run ()
    {
    prog->Del(wnd_c);
    }
};
//============================================================================
class CWarning : public CContainer
{
char *mes_c, *title_c;

public:
CWarning (char *mes = NULL, char *title = NULL) : CContainer (0,0,0,0)
    {
    mes_c   = mes;
    title_c = title;
    for (int title_lgh = 0; title_lgh < SCR_ROW; title_lgh++) if (!title_c[title_lgh]) break;
    for (int mes_lgh = 0;   mes_lgh < MAX_MES;   mes_lgh++)   if (!mes_c[mes_lgh])     break;
    int i     = 0,
        j     = 2,
        x     = 1,
        x_max = 0;
    while (i < mes_lgh)
        {
        if (mes_c[i] == '\n' || x > SCR_ROW)
            {
            j++;
            if (x > x_max) x_max = x;
            x = 1;
            }
        OutText (mes_c+i, x, j ,1);
        i++;
        x++;
        }
    if (x > x_max) x_max = x;
    if (x_max > title_lgh)
        {
        x_c     = (SCR_ROW - x_max) / 2;
        x_end_c = x_c + x_max;
        }
    else
        {
        x_c     = (SCR_ROW - title_lgh - 2) / 2;
        x_end_c = SCR_ROW - ((SCR_ROW - title_lgh - 2) / 2);
        }
    y_c     = (SCR_STRING - (2 + 3 + j)) / 2;
    y_end_c = y_c + 2 + 3 + j;
    OutText (title_c, ((x_end_c - x_c) - title_lgh)/2 ,0 ,title_lgh);
    CChildWnd* wnd = NEW (CCancelBut ((x_end_c-x_c-2)/2, j+2, this, "  ~OK  "));
    Add (wnd);
    }

void WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, TRUE, TRUE, 0 , 7, 0);
    Scr.BackColor (7);
    for (int i = x_c+1; i < x_end_c; i++)
        {
        for (int j = y_c+1; j < y_end_c; j++)
            {
            Scr.Put (i,j);
            }
        }
    }

void ReDraw ()
    {
    WndFrameDraw();
    DrawText();
    if (child_c[0]) ChildReDraw();
    prog->EndDraw();
    OnKeyDo(0);
    }
```

```
void OnKeyDo (int c)
    {
    Scr.SetCur ();
    while(1)
        {
        switch (c)
            {
            case KBD_TAB:   CChildWnd* temp = child_c[0];
                             for (int i = 0; i <= count_c-1; i++)
                                 {
                                 child_c[i] = child_c[i+1];
                                 }
                             child_c[count_c] = temp;
                             break;
            case KBD_ENTER: child_c[0]->Run();return;
            }
        if (child_c[0]->HotKey(c)) {child_c[0]->Run();return;}
        if (child_c[0]) ChildReDraw();
        c = Kbd.WaitKey();
        }
    }
};
//=============================================================================
class CEditWnd : public CContainer
{
protected:
CScrollBar *hor_scroll_c, *ver_scroll_c;
int        title_lgh_c;

public:
CEdit      *edit_c;

CEditWnd (int x, int y, int x_end, int y_end, char* title, int title_lgh) : CContainer
    (x,y,x_end,y_end)
    {
    title_c            = title;
    title_lgh_c        = title_lgh;
    title_c[title_lgh] = 0;
    edit_c             = NEW (CEdit);
    hor_scroll_c       = NEW (CScrollBar);
    ver_scroll_c       = NEW (CScrollBar);
    edit_c->SetSize(x_end-x-1,y_end-y-1);
    Add (edit_c);
    hor_scroll_c->Set(1,y_end-y,x_end-x-2,HOR);
    ver_scroll_c->Set(x_end-x,1,y_end-y-2,VERT);
    hor_scroll_c->SetScrollMax(MAX_EDIT_SYM-(x_end_c-x_c));
    Add (hor_scroll_c);
    Add (ver_scroll_c);
    TitleFix();
    }

int ManMes (int mes)
    {
    switch (mes)
        {
        case MOVE_LEFT:    x_c--; x_end_c--;if (testxy()) return TRUE; else x_c++;
    x_end_c++;break;
        case MOVE_RIGHT:   x_c++; x_end_c++;if (testxy()) return TRUE; else x_c--; x_end_c--
    ;break;
        case MOVE_UP:      y_c--; y_end_c--;if (testxy()) return TRUE; else y_c++;
    y_end_c++;break;
        case MOVE_DOWN:    y_c++; y_end_c++;if (testxy()) return TRUE; else y_c--; y_end_c--
    ;break;
        case BIGGER_LEFT:  x_end_c--; if (testxy()) {SetChilds();return TRUE;} else
    x_end_c++;break;
        case BIGGER_RIGHT: x_end_c++; if (testxy()) {SetChilds();return TRUE;} else x_end_c--
    ;break;
        case BIGGER_UP:    y_end_c--; if (testxy()) {SetChilds();return TRUE;} else
    y_end_c++;break;
        case BIGGER_DOWN:  y_end_c++; if (testxy()) {SetChilds();return TRUE;} else y_end_c--
    ;break;
        case FULL_SCR:      int temp    = y_c;
                            y_c         = y_old_c;
                            y_old_c     = temp;
                            temp        = x_c;
                            x_c         = x_old_c;
                            x_old_c     = temp;
                            temp        = y_end_c;
                            y_end_c     = y_end_old_c;
                            y_end_old_c = temp;
                            temp        = x_end_c;
                            x_end_c     = x_end_old_c;
                            x_end_old_c = temp;
                            SetChilds();
                            ChildReDraw();
                            return TRUE;
```

```
        default:           return FALSE;
        }
    return FALSE;
    }

void SetChilds ()
    {
    TitleFix();
    edit_c->SetSize(x_end_c-x_c-1,y_end_c-y_c-1);
    hor_scroll_c->Set(1,y_end_c-y_c,x_end_c-x_c-2,HOR);
    ver_scroll_c->Set(x_end_c-x_c,1,y_end_c-y_c-2,VERT);
    hor_scroll_c->SetScrollMax(MAX_EDIT_SYM);
    child_c[0]->OnKeyDo('□');
    }

void TitleFix  ()
    {
    clear_str (text_c[0]);
    OutText (title_c,
    (x_end_c-x_c) > (title_lgh_c+4) ? ((x_end_c - x_c) - title_lgh_c)/2 : 2,0,
    (x_end_c-x_c) > (title_lgh_c) ? title_lgh_c : ((x_end_c - x_c) - 4));
    }

void OnKeyDo(int c)
    {
    child_c[0]->OnKeyDo(c);
    child_c[0]->ReDrawAct(x_c,y_c);
    if (edit_c->GetX() >= (x_end_c-x_c)) hor_scroll_c->SetScroll(edit_c->GetX()-(x_end_c-x_c));
    else hor_scroll_c->SetScroll(0);
    ver_scroll_c->SetScrollMax(edit_c->GetMaxY());
    ver_scroll_c->SetScroll(edit_c->GetY());
    child_c[1]->ReDraw(x_c,y_c);
    child_c[2]->ReDraw(x_c,y_c);
    return;
    }

void ReDraw (int x, int y)
    {
    WndFrameDraw();
    DrawText();
    child_c[0]->ReDraw(x_c,y_c);
    child_c[1]->ReDraw(x_c,y_c);
    child_c[2]->ReDraw(x_c,y_c);
    }

void WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, TRUE, FALSE, 15, 1, 1);
    Scr.BackColor (1);
    }

~CEditWnd()
    {
    DEL (title_c);
    DEL (edit_c);
    DEL (hor_scroll_c);
    DEL (ver_scroll_c);
    Scr.SetCur();
    }

int Save (char* file = NULL) {if (!file) file = title_c;return edit_c->Save(file);}
};
//=============================================================================
class CFileDlg : public CContainer
{
public:
char *title_c, *path_c, *dir_c;
int  error_c, title_lgh_c;

CList*       list_c;
CEditLine*   edit_c;
CCancelBut*  canc_c;
//CScrollBar scroll_c;

//!!! ReDraw { OnKeyDo }

CFileDlg (char* title, int title_lgh) : CContainer (14,3,64,21)
    {
    title_c     = title;
    title_lgh_c = title_lgh < 50 ? title_lgh : 50;
    TitleFix();

    edit_c = NEW (CEditLine (3,3,28));
    Add(edit_c);
    list_c = NEW (CList (3,6,8,15,2));
    Add(list_c);
    canc_c = NEW (CCancelBut (35, 6, this, " Cancel  "));
```

```
    Add(canc_c);

    OutText("Name",3,2,4);
    OutText("File",3,5,4);
    path_c  = NEW (char [MAX_STR+1]);
    dir_c   = NEW (char [MAX_EDIT_SYM]);
    error_c = 0;

    FileList();
//  Add(&scroll_c);
    }

void LineDo ()
    {
    edit_c->Clear();
    getcwd (path_c, MAX_STR);
    for (int i = 0; i < MAX_STR; i++)
        {
        if (*(path_c+i) == '\0') break;
        edit_c->OnKeyDo(*(path_c+i));
        }
    if (*(path_c+i-1) != '\\') edit_c->OnKeyDo('\\');
    for (i = 0; i < MAX_FILE_NAME; i++)
        {
        if (*(mask+i) == '\0') break;
        edit_c->OnKeyDo(*(mask+i));
        }
    }

void Error ()
    {
    error_c = 1;
    CWindow* wnd = NEW (CWarning ("  Invalide drive or directory  "," Error "));
    prog->AddWnd (wnd);
    }

void TitleFix()
    {
    clear_str (text_c[0]);
    OutText (title_c,
    (x_end_c-x_c) > (title_lgh_c+4) ? ((x_end_c - x_c) - title_lgh_c)/2 : 2,0,
    (x_end_c-x_c) > (title_lgh_c) ? title_lgh_c : ((x_end_c - x_c) - 4));
    }

void WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, TRUE, TRUE, 0 , 7, 0);
    Scr.BackColor (7);
    for (int i = x_c+1; i < x_end_c; i++)
        {
        for (int j = y_c+1; j < y_end_c; j++)
            {
            Scr.Put (i,j);
            }
        }
    }

void FileList ()
    {
    struct ffblk ffblk;
    int done;
    list_c->Clear();
    edit_c->Clear();

    getcwd (path_c, MAX_STR);
    for (int i = 0; i < MAX_STR; i++)
        {
        if (*(path_c+i) == '\0') break;
        edit_c->OnKeyDo(*(path_c+i));
        }
    if (*(path_c+i-1) != '\\') edit_c->OnKeyDo('\\');
    for (i = 0; i < MAX_FILE_NAME; i++)
        {
        if (*(mask+i) == '\0') break;
        edit_c->OnKeyDo(*(mask+i));
        }
    char* dir = NEW (char [MAX_FILE_NAME+1]);
    done = findfirst("*.*",&ffblk,0x10);
    while (!done)
        {
        if (((ffblk.ff_attrib & 0x10) != 0x10) || ((ffblk.ff_name[0] == '.') &&
    (ffblk.ff_name[1] == '\0')))
            {
            done = findnext(&ffblk);
            continue;
            }
        for (int i = 0; i < MAX_FILE_NAME; i++)
```

```
                {
                if (ffblk.ff_name[i] == '\0')
                    {
                    *(dir+i) = '\\';
                    *(dir+i+1) = '\0';
                    break;
                    }
                *(dir+i) = ffblk.ff_name[i];
                }
            list_c->Add(dir,i+1);
            done = findnext(&ffblk);
            }
        DEL (dir);

        done = findfirst(mask,&ffblk,0);
        while (!done)
            {
            for (int i = 0; i < MAX_FILE_NAME; i++) if (ffblk.ff_name[i] == '\0') break;
            list_c->Add(ffblk.ff_name,i);
            done = findnext(&ffblk);
            }
        }

    char* LineRead ()
        {
        char* line = edit_c->GetEdit();
        int drive = ((*(line) | 0x20) - 'a');
        if (*(line + 1) == ':')
            {
            if (drive != getdisk())
                {
                setdisk (drive);
                if (getdisk() != (drive)) {Error();return "\0";}
                }
            line += 2;
            }
        char* dir     = NEW (char [MAX_FILE_NAME]);
        int   pnt     = 0,
              dir_pnt = 0;
        *(dir_c + pnt) = getdisk() + 'A';
        pnt++;
        *(dir_c + pnt) = ':';
        pnt++;
        *(dir_c + pnt) = '\\';
        pnt++;
        if (*(line) == '\\') line++;
        while (*(line) != '\0')
            {
            if (*(line) == '\\')
                {
                for (int i = 0; i < dir_pnt; i++)
                    {
                    *(dir_c + pnt) = *(dir + i);
                    pnt++;
                    }
                *(dir_c + pnt) = '\\';
                pnt++;
                dir_pnt = 0;
                line++;
                continue;
                }
            dir[dir_pnt] = *line;
            dir_pnt++;
            line++;
            }
        dir_c[pnt]     = 0;
        dir[dir_pnt]   = 0;
        int i = 0;       //for goto
        if (!pnt) {goto no_ch;}
        for (i = 0; i < dir_pnt; i++) dir_c[pnt + i] = dir[i];
        dir_c[pnt + i] = 0;
        if (!chdir(dir_c)) {DEL (dir);return "\0";}
        dir_c[pnt - 1] = 0;
        if (chdir(dir_c)) {Error();DEL (dir);return "\0";}
        if (dir_pnt == 0) {DEL (dir);return "\0";}

    no_ch:
        int exp = 0;
        for (i = 0; i < dir_pnt; i++)
            {
            if (*(dir + i) == '*' || *(dir + i) == '?')
                {
                for (int j = 0; j < dir_pnt; j++) *(mask + j) = *(dir + j);
                *(mask + j) = '\0';
                DEL (dir);
                return "\0";
                }
```

```
        if (*(dir + i) == '.')
            {
            exp = 1;
            }
        }
    if (exp == 1) return dir;
    *(dir + dir_pnt) = '.';
    dir_pnt++;
    *(dir + dir_pnt) = 'a';
    dir_pnt++;
    *(dir + dir_pnt) = 's';
    dir_pnt++;
    *(dir + dir_pnt) = 'm';
    dir_pnt++;
    *(dir + dir_pnt) = '\0';
    return dir;
    }

void OnKeyDo(int c)
    {
    while (1)
        {
        switch (c)
            {
            case KBD_TAB:   if (child_c [0] == edit_c) LineDo();
                    CChildWnd* temp = child_c [0];
                    for (int i = 0; i <= count_c-1; i++)
                        {
                        int k = i + 1;
                        child_c [i] = child_c [k];
                        }
                    child_c [count_c] = temp;
                    break;
            case KBD_ENTER: if (child_c[0] == list_c)
                        {
                        for (i = 0; i < MAX_FILE_NAME; i++)
                            {
                            if (*(list_c->GetCurrent() + i) == '\\') break;
                            }
                        if (i == MAX_FILE_NAME)
                            {
                            char* file = new char [MAX_FILE_NAME];
                            for (int i = 0; i < MAX_FILE_NAME; i++)
                                {
                                *(file + i) = *(list_c->GetCurrent() + i);
                                if (*(list_c->GetCurrent()) == '\0') break;
                                }
                            canc_c->Run();
                            Run (file);
                            return;
                            }
                        *(list_c->GetCurrent() + i) = '\0';
                        chdir(list_c->GetCurrent());
                        FileList();
                        break;
                        }
                    if (child_c[0] == edit_c)
                        {
                        char* file = LineRead();
                        if (*(file) == '\0' || error_c == 1) {/*DEL file;*/FileList();break;}
                        canc_c->Run();
                        Run (file);
                        DEL (file);
                        return;
                        }
                    if (child_c[0] == canc_c)
                        {
                        canc_c->Run();return;
                        }
                    else break;
            case KBD_ESC: case KBD_ALTF3: prog->Del(this);return;
            default:    child_c[0]->OnKeyDo(c);break;
            }
        if (child_c[0] != NULL) ChildReDraw();
        c = Kbd.WaitKey();
        }
    }

void virtual Run (char*) {}

virtual ~CFileDlg()
    {
    Scr.SetCur(0,0,NO_CUR);
    DEL (canc_c);
    DEL (list_c);
    DEL (edit_c);
    DEL (dir_c);
```

```
    DEL (path_c);
    }
};

//============================================================================
class CSaveBox : public CFileDlg
{
CCancelBut* save_c;
public:

CSaveBox () : CFileDlg (" Save file as ", 14)
    {
    save_c = NEW (CCancelBut (35,3,NULL," Save    "));
    Add (save_c);
    }

    void OnKeyDo(int c)
    {
    while (1)
        {
        switch (c)
            {
            case KBD_TAB:   if (child_c [0] == edit_c) LineDo();
                    CChildWnd* temp = child_c [0];
                    for (int i = 0; i <= count_c-1; i++)
                        {
                        int k = i + 1;
                        child_c [i] = child_c [k];
                        }
                    child_c [count_c] = temp;
                    break;
            case KBD_ENTER: if (child_c[0] == list_c || child_c[0] == save_c)
                        {
                        for (i = 0; i < MAX_FILE_NAME; i++)
                            {
                            if (*(list_c->GetCurrent() + i) == '\\') break;
                            }
                        if (i == MAX_FILE_NAME)
                            {
                            for (i = 0; i < MAX_FILE_NAME; i++)
                                if (*(list_c->GetCurrent() + i) == ' ') break;
                            *(list_c->GetCurrent() + i) = '\0';
                            char* file = new char [MAX_FILE_NAME];
                            for (i = 0; i < MAX_FILE_NAME; i++)
                                {
                                *(file + i) = *(list_c->GetCurrent() + i);
                                if (*(list_c->GetCurrent()) == '\0') break;
                                }
                            Run (file);
                            canc_c->Run();return;
                            }
                        *(list_c->GetCurrent() + i) = '\0';
                        chdir(list_c->GetCurrent());
                        FileList();
                        break;
                        }
                    if (child_c[0] == edit_c)
                        {
                        char* file = LineRead();
                        if (*(file) == '\0' || error_c == 1) {/*DEL file*/;FileList();break;}
                        Run (file);
                        DEL (file);
                        canc_c->Run();return;
                        }
                    if (child_c[0] == canc_c)
                        {
                        canc_c->Run();return;
                        }
                    else break;
            case KBD_ESC: case KBD_ALTF3: prog->Del(this);return;
            default:         child_c[0]->OnKeyDo(c);break;
            }
        if (child_c[0] != NULL) ChildReDraw();
        c = Kbd.WaitKey();
        }
    }

void static Run (char* file)
    {
    if (prog->GetAct1()->Save(file) == FALSE)
        prog->AddWnd(new CWarning("       Can not save file.       "," Error "));
    }

~CSaveBox ()
    {
    DEL (save_c);
    }
```

```
};
//============================================================================
class COpenBox : public CFileDlg
{
CCancelBut* open_c;
public:
COpenBox () : CFileDlg (" Open file ", 11)
    {
    open_c = NEW (CCancelBut (35,3,NULL," Open    "));
    Add (open_c);
    }

    void OnKeyDo(int c)
    {
    while (1)
        {
        switch (c)
            {
            case KBD_TAB:   if (child_c [0] == edit_c) LineDo();
                    CChildWnd* temp = child_c [0];
                    for (int i = 0; i <= count_c-1; i++)
                        {
                        int k = i + 1;
                        child_c [i] = child_c [k];
                        }
                    child_c [count_c] = temp;
                    break;
            case KBD_ENTER: if (child_c[0] == list_c || child_c[0] == open_c)
                        {
                        for (i = 0; i < MAX_FILE_NAME; i++)
                            {
                            if (*(list_c->GetCurrent() + i) == '\\') break;
                            }
                        if (i == MAX_FILE_NAME)
                            {
                            for (i = 0; i < MAX_FILE_NAME; i++)
                                if (*(list_c->GetCurrent() + i) == ' ') break;
                            *(list_c->GetCurrent() + i) = '\0';
                            char* file = new char [MAX_FILE_NAME];
                            for (i = 0; i < MAX_FILE_NAME; i++)
                                {
                                *(file + i) = *(list_c->GetCurrent() + i);
                                if (*(list_c->GetCurrent()) == '\0') break;
                                }
                            Run (file);
                            canc_c->Run();return;
                            }
                        *(list_c->GetCurrent() + i) = '\0';
                        chdir(list_c->GetCurrent());
                        FileList();
                        break;
                        }
                    if (child_c[0] == edit_c)
                        {
                        char* file = LineRead();
                        if (*(file) == '\0' || error_c == 1) {/*DEL file*/;FileList();break;}
                        Run (file);
                        DEL (file);
                        canc_c->Run();return;
                        }
                    if (child_c[0] == canc_c)
                        {
                        canc_c->Run();return;
                        }
                    else break;
            case KBD_ESC: case KBD_ALTF3: prog->Del(this);return;
            default:    child_c[0]->OnKeyDo(c);break;
            }
        if (child_c[0] != NULL) ChildReDraw();
        c = Kbd.WaitKey();
        }
    }

void static Run (char* file)
    {
    for (int i = 0; i < MAX_FILE_NAME; i++) if (*(file + i) == '\0') break;
    CEditWnd* edit = NEW (CEditWnd (0, 1, SCR_ROW-1, SCR_STRING-1, file, i));
    FILE* handler;
    if ((handler = fopen (file, "r")) != NULL)
        {
        int i = 0;
        while (!feof(handler))
            {
            int c = fgetc (handler);
            if (c == '\n')
                {
                edit->edit_c->OnKeyDo(KBD_ENTER);
```

```
                i++;
                continue;
                }
            edit->edit_c->OnKeyDo(c);
            }
        for (i = 0; i < 1000/25; i++) edit->edit_c->OnKeyDo(KBD_PGUP);
        edit->edit_c->OnKeyDo(KBD_HOME);
        }
    else if ((handler = fopen (file, "w")) == NULL)
        {
        DEL (edit);
        CWindow* wnd = NEW (CWarning ("  Error  ","      Open error      "));
        prog->AddWnd (wnd);
        return;
        }
    fclose (handler);
    prog->AddWnd (edit);
    }

~COpenBox ()
    {
    DEL (open_c);
    }
};
```

File Core\Core.cpp

```
//---INCLUDES-------------------------------------------------------------------
//For files
#include <stdio.h>
#include <io.h>
#include <dos.h>

//For strcat,itoa
#include <stdlib.h>
#include <string.h>

//Default includes
#include <process.h>
#include <dir.h>
#include <fcntl.h>

//---MACROS---------------------------------------------------------------------
#define WND_DEBUG 0
#define DEBUG     0

void debug_new (int line, char *file);
void debug_del (int line, char *file, size_t size);
#define NEW(i) new i;debug_new(__LINE__, __FILE__);
#define DEL(i) debug_del(__LINE__, __FILE__, sizeof(*i));delete i;i=NULL

#define iget1(num)  ((num)%10)
#define iget2(num) (((num)%100)/10)
#define iget3(num) (((num)%1000)/100)
#define iget4(num) (((num)%10000)/1000)

//-DEFINITION------------------------------------------------------------------
#include "cswap.h"
#include "cfswap.h"
char *mask         = "*.asm";
char *no_name      = "nonameXX.asm";
int  no_name_count = 0;
FILE *deb          = NULL;

void open_func      ();
void disasm_func    ();
void save_func      ();
int  asm1_func      ();
void asm_func       ();
void prog_func      ();
void zoom_func      ();
void next_func      ();
void close_func     ();
void step_into_func ();
void step_over_func ();
void goto_cur_func  ();
void read_func   ();
void write_func      ();
void erase_func     ();
void verify_func    ();
void help_func      ();

#include "w.cpp"
```

```
CProg* prog        = NULL;

#include "classes.cpp"
#include "debugwnd.cpp"

void debug_new (int line, char *file)
     {
     #if DEBUG
     fprintf(deb,"NEW in (%d, %s), size %d\n", line, file, size);
     #endif
     }

void debug_del (int line, char *file, size_t size)
     {
     #if DEBUG
     fprintf(deb, "del in (%d, %s), size %d\n", line, file, size);
     #endif
     }

//-------------------------------------------------------------------------
void read_func   ()
     {
     if ((spawnlp(P_WAIT, "prog.exe", " ", "r", "1", NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec PROG.EXE       "," Error "));
          prog->AddWnd(wnd);
          }
     }

void write_func      ()
     {
     rename(prog->GetAct()->title_c,"in.hex");
     if ((spawnlp(P_WAIT, "prog.exe", " ", "w", "1", NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec PROG.EXE       "," Error "));
          prog->AddWnd(wnd);
          }
     remove("in.hex");
     }

void erase_func     ()
     {
     if ((spawnlp(P_WAIT, "prog.exe", " ", "e", "1", NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec PROG.EXE       "," Error "));
          prog->AddWnd(wnd);
          }
     }

void verify_func    ()
     {
     rename(prog->GetAct()->title_c,"in.hex");
     if ((spawnlp(P_WAIT, "prog.exe", " ", "v", "1", NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec PROG.EXE       "," Error "));
          prog->AddWnd(wnd);
          }
     remove("in.hex");
     }

//-HELP-----------------------------------------------------------------------
void help_func()
     {
     CWindow* wnd = NEW (CWarning("  Sorry no help in this vertion  "," Error "));
     prog->AddWnd(wnd);
     }

//-STEP-OVER------------------------------------------------------------------
void step_over_func()
     {
     char* asm_file = prog->GetAct()->title_c;
     if(strstr(asm_file, "PIC16F84-04") != 0)
          {
          prog->GetAct()->OnKeyDo(KBD_F8);
          Scr.SetCur();
          return;
          }
     }

//-GOTO-CURSOR----------------------------------------------------------------
void goto_cur_func()
     {
     char* asm_file = prog->GetAct()->title_c;
     if(strstr(asm_file, "PIC16F84-04") != 0)
          {
          prog->GetAct()->OnKeyDo(KBD_F4);
          Scr.SetCur();
```

```
        return;
        }
    }

//-STEP-INTO-------------------------------------------------------------------
void step_into_func()
    {
    char* asm_file = prog->GetAct()->title_c;
    if(strstr(asm_file, "PIC16F84-04") != 0)
        {
        prog->GetAct()->OnKeyDo(KBD_F7);
        Scr.SetCur();
        return;
        }
    char* file = NEW (char[MAX_FILE_NAME]);
    for (int i = 0; i < (MAX_FILE_NAME-3); i++)
        {
        *(file + i) = *(asm_file + i);
        if (*(asm_file + i) == '.' || *(asm_file + i) == 0) break;
        }
    if (*(asm_file + i) == 0) *(asm_file + i++) = '.';
    else i++;
    *(file + i++) = 'h';
    *(file + i++) = 'e';
    *(file + i++) = 'x';
    *(file + i++) = 0;

    if (strcmpi(prog->GetAct()->title_c, file) != 0) if (!asm1_func()) return;
    CWindow* wnd = NEW (CDebugWnd(file));
    prog->AddWnd(wnd);
    Scr.SetCur();
    }

//-MOVE-----------------------------------------------------------------------
void move_func ()
    {
    prog->Move();
    }

//-SIZE-----------------------------------------------------------------------
void size_func ()
    {
    prog->Size();
    }

//-ZOOM-----------------------------------------------------------------------
void zoom_func ()
    {
    prog->Zoom();
    }

//-NEXT-----------------------------------------------------------------------
void next_func ()
    {
    prog->Next();
    }

//-CLOSE----------------------------------------------------------------------
void close_func ()
    {
    prog->Close();
    }

//-EXIT-----------------------------------------------------------------------
void exit_func ()
    {
    DEL (prog);
    exit(0);
    }

//-NEW------------------------------------------------------------------------
void new_func ()
    {
    char *title = new char [MAX_FILE_NAME];
    for (int i = 0; i <= MAX_FILE_NAME; i++)
        {
        title[i] = no_name[i];
        if (!*no_name) break;
        }
    title[i - 7] = iget1(no_name_count) + '0';
    title[i - 8] = iget2(no_name_count) + '0';
    no_name_count++;
    CEditWnd* wnd = NEW (CEditWnd(0, 1, SCR_ROW-1, SCR_STRING-1, title, 12));
    prog->AddWnd (wnd);
    }

//-OPEN-----------------------------------------------------------------------
```

```
void open_func ()
     {
     CWindow* wnd = NEW (COpenBox);
     prog->AddWnd (wnd);
     }

//-SAVE-------------------------------------------------------------------------
void save_func ()
     {
     if (!prog->GetAct()->Save(NULL))
          {
          CWindow* wnd = NEW (CWarning("       Can not save file.       "," Error "));
          prog->AddWnd(wnd);
          }
     }

//-SAVE-AS----------------------------------------------------------------------
void save_as_func ()
     {
     CWindow* wnd = NEW (CSaveBox);
     prog->AddWnd (wnd);
     }

//-ASM--------------------------------------------------------------------------
void asm_func ()
     {
     asm1_func();
     }

int asm1_func ()
     {
     if (prog->GetAct() == NULL)   return 0;
     if (!prog->GetAct()->Save(NULL))
          {
          CWindow* wnd = NEW (CWarning("       Can not assembly file       "," Error "));
          prog->AddWnd(wnd);
          return 0;
          }
     if ((spawnlp(P_WAIT, "asm.exe", " ", prog->GetAct()->title_c, NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec ASM.EXE        "," Error "));
          prog->AddWnd(wnd);
          return 0;
          }
     prog->ReDraw();
     return 1;
     }

//-DISASM-----------------------------------------------------------------------
void disasm_func ()
     {
     if (prog->GetAct() == NULL)   return;
     if (!prog->GetAct()->Save(NULL))
          {
          CWindow* wnd = NEW (CWarning("       Can not disassembly file       "," Error "));
          prog->AddWnd(wnd);
          return;
          }
     if ((spawnlp(P_WAIT, "disasm.exe", "", prog->GetAct()->title_c, NULL)) == -1)
          {
          CWindow* wnd = NEW (CWarning("     Can not exec DISASM.EXE     "," Error "));
          prog->AddWnd(wnd);
          return;
          }
     prog->ReDraw();
     }

//-PROG-------------------------------------------------------------------------
void prog_func ()
     {
     if (spawnlp(P_WAIT, "prog.exe", "", NULL) == -1)
          {
          CWindow* wnd = NEW (CWarning("       Can not exec PROG.EXE       "," Error "));
          prog->AddWnd(wnd);
          }
     prog->ReDraw();
     }

//-MAIN-------------------------------------------------------------------------
void main (int argc, char* argv [])
     {
     #if DEBUG
     deb = fopen ("debug","w+");
     #endif

     prog = NEW (CProg);
     CMenu* file_menu = NEW (CMenu (1," ~File ",6));
```

```
    prog->AddMenu (file_menu);
    file_menu->AddElement ("~New             ",16,new_func);
    file_menu->AddElement ("~Open...      F3 ",16,open_func);
    file_menu->AddElement ("~Save         F2 ",16,save_func);
    file_menu->AddElement ("S~ave as...      ",16,save_as_func);
    file_menu->OutText    ("□œœœœœœœœœœœœœœœœŒ",0,5,18);
    file_menu->AddElement ("E~xit      Atl-X ",16,exit_func);

    CMenu* comp_menu = NEW (CMenu (7," ~Compile ",9));
    prog->AddMenu (comp_menu);
    comp_menu->AddElement ("~Assembly          F9 ",21,asm_func);
    comp_menu->AddElement ("~Disassembly  Alt-F10 ",21,disasm_func);

    CMenu* debug_menu = NEW (CMenu (16," ~Debug ",7));
    prog->AddMenu (debug_menu);
    debug_menu->AddElement ("St~ep into    F7 ",16,step_into_func);
    debug_menu->AddElement ("Step ~over    F8 ",16,step_over_func);
    debug_menu->AddElement ("~Go to cursor F4 ",16,goto_cur_func);

    CMenu* prog_menu = NEW (CMenu (23," ~Program ",9));
    prog->AddMenu (prog_menu);
    prog_menu->AddElement ("~Write   Ctrl-F9 ",16,write_func);
    prog_menu->AddElement ("~Verify   Alt-F9 ",16,verify_func);
    prog_menu->OutText    ("□œœœœœœœœœœœœœœœœŒ",0,3,18);
    prog_menu->AddElement ("~Read            ",16,read_func);
    prog_menu->OutText    ("□œœœœœœœœœœœœœœœœŒ",0,5,18);
    prog_menu->AddElement ("~Erase    Atl-F8 ",16,erase_func);

    CMenu* win_menu = NEW (CMenu (33," ~Window ",8));
    prog->AddMenu (win_menu);
    win_menu->AddElement ("~Zoom              F5 ",21,zoom_func);
    win_menu->AddElement ("N~ext              F6 ",21,next_func);
    win_menu->AddElement ("~Close         Atl-F3 ",21,close_func);
    win_menu->OutText    ("□œœœœœœœœœœœœœœœœœœœœœŒ",0,4,23);
    win_menu->AddElement ("~Move         Ctrl-F5 ",21,move_func);
    win_menu->AddElement ("~Size         Ctrl-F6 ",21,size_func);

    CMenu* help_menu  = NEW (CMenu (74," ~Help ",6));
    prog->AddMenu (help_menu);
    help_menu->AddElement ("~Asm help   Alt-F1 ",18,help_func);
    help_menu->AddElement ("~Mcu help  Ctrl-F1 ",18,help_func);
    help_menu->OutText    ("□œœœœœœœœœœœœœœœœœœŒ",0,3,20);
    help_menu->AddElement ("~IDE help       F1 ",18,help_func);

//-DEBUG------------------------------------------------------------------
    #if WND_DEBUG
    CWindow* wnd = NEW (CDebugWnd("test"));
    prog->AddWnd(wnd);
    #endif WND_DEBUG
//-----------------------------------------------------------------------

    prog->Run();

    DEL (help_menu);
    DEL (win_menu);
    DEL (prog_menu);
    DEL (debug_menu);
    DEL (comp_menu);
    DEL (file_menu);
    DEL (prog);

    #if DEBUG
    fclose (deb);
    #endif
    }
```

File Core\CScreen.h

```
#ifndef __CSCREEN_H__
#define __CSCREEN_H__

//===CONSTANTS============================================================
#define CSCREEN_VER  1
#define SCR_ROW      80
#define SCR_STRING   25
#define NO_CUR       0
#define NOR_CUR      1
#define FULL_CUR     2

//===DEFINITION===========================================================
class CScreen
{
```

```
//---INTERFACE---
public:
     CScreen       (char far*);
     ~CScreen      ()                                { if (real_video) delete (scr_adr_c); }
void Put           (int, int, char, int, int);
char Get           (int x = 0, int y = 0)            { return *(get_adr(x, y)); }
void PutStr        (char*, int, int, int, int);
void Clr           ();
void SetColor      (int, int, int, int);
void TextColor     (int color = 79)                  { if (color != 79) text_color_c = color; }
void BackColor     (int color = 79)                  { if (color != 79) backgnd_color_c = color;
    }
int  GetTextColor  ()                                { return text_color_c; }
int  GetBackColor  ()                                { return backgnd_color_c; }
void SetCur        (char, char, int);
void SetBaseAdress (char far*);

//---IN LIFE---
protected:
int  real_scr     (int far*);
char far* get_adr (int, int);
int  gotoxy       (int, int);

//---ATRIBUTES---
private:
char far *scr_adr_c;
int      real_video,
         x_c, y_c,
         text_color_c, backgnd_color_c;
};

//--REALISATION---
CScreen::CScreen (char far* scr_adr = (char far*) NULL)
    {
    if (!scr_adr)
        {
        scr_adr_c = NEW (char far [SCR_ROW*SCR_STRING*2]);
        real_video = TRUE;
        }
    else
        {
        scr_adr_c = scr_adr;
        real_video = FALSE;
        }
    text_color_c = 8; backgnd_color_c = 0;
    }

void CScreen::SetBaseAdress (char far* scr_adr = (char far*) NULL)
    {
    scr_adr_c = scr_adr;
    real_video = FALSE;
    }

void CScreen::Put (int x = 0, int y = 0, char c = ' ', int text_color=79, int backgnd_color=79)
    {
    *(get_adr(x, y)) = c;
    SetColor(x, y, text_color, backgnd_color);
    }

void CScreen::Clr ()
    {
    for (int i = 0; i < SCR_ROW; i++)
        {
        for (int j = 0; j < SCR_STRING; j++)
            {
            Put (i,j);
            }
        }
    }

void CScreen::SetColor (int x = 0, int y = 0, int text_color = 79, int backgnd_color = 79)
    { TextColor (text_color);
      BackColor (backgnd_color);
      *(get_adr(x, y)+1) = text_color_c + backgnd_color_c * 16;}

void CScreen::SetCur (char x = 0, char y = 0, int type = 0)
    {
    asm {
        mov ah,0x0f
        int 0x10
        mov dh,y
        mov dl,x
        mov ah,0x02
        int 0x10
        }

    if ( type == NO_CUR)
```

```
        {
        asm {
            mov ah,0x01
            mov cx,0x0100
            int 0x10
            }
        return;
        }

    if ( type == NOR_CUR)
        {
        asm {
            mov ah,0x01
            mov cx,0x0607
            int 0x10
            }
        return;
        }

    if ( type == FULL_CUR)
        {
        asm {
            mov ah,0x01
            mov cx,0x0007
            int 0x10
            }
        }
    }

int CScreen::real_scr (int far* scr_adr = (int far*) NULL)
    {
    if (( (int) scr_adr_c & 0xffff0000l) == 0xb8000000l) return TRUE;
    else return FALSE;
    }

char far* CScreen::get_adr (int x = 0, int y = 0)
    {
    if (gotoxy (x,y))
        {
        return scr_adr_c + (2 * ((y_c * SCR_ROW) + x_c));
        }
    return scr_adr_c;
    }

int CScreen::gotoxy (int x = 0, int y = 0)
    {
    x_c = x;
    y_c = y;
    if (x > SCR_ROW || y > SCR_STRING || x < 0 || y < 0) return FALSE;
    else  return TRUE;
    }

void CScreen::PutStr (char* str, int x = 0, int y = 0, int bg_col = 79, int text_col = 79)
    {
    int i = 0;
    while (str[i] != 0)
        {
        Put(x+i, y, str[i], bg_col, text_col);
        i++;
        }
    }

//EOF
#endif  __CSCREEN_H__
```

File Core\CSwap.h

```
#ifndef __CSWAP_H__
#define __CSWAP_H__

//===INCLUDES=============================================================
#include <assert.h>
#include "CSwapMng.h"

#define NULL 0
#define TRUE 1
#define FALSE 0
#define MAX_LOADED 100
//===DEFINITION===========================================================
#ifndef element_c
struct element_c
    {
    int  index;
```

```
    void *offset;
    };
#endif element_c

template <class T> class CSwap
{

//---INTERFACE---
public:
      CSwap        ();
      CSwap       (int max_lgh, int max_loaded);
void  SetOffset   (T& dist,        void* scr,        int del = 1);
void  SetOffset1  (void* dist,     void* scr = NULL, int del = 1);
void  SetOffset2  (int index1 = 0, int index2 = 0,   int del = 1);
void  SetOffset3  (int index1 = 0, void* scr = NULL, int del = 1);

T&    operator[]  (int index);
void  SwapMin     ();
      ~CSwap      ();

//---ATRIBUTES---
private:
unsigned long  file_offset_c;
int            max_lgh_c,
               loaded_c,
               max_loaded_c,
               last_loaded_c;
unsigned int  far* cach_table_c;
element_c     far* incach_c;
};

//--REALISATION---
template<class T> CSwap<T>::CSwap ()
    {
    max_lgh_c     = 1000;
    file_offset_c = SwapMng.NewSwap(max_lgh_c, sizeof(T));
    loaded_c      = 0;
    max_loaded_c  = 50;
    cach_table_c  = new unsigned int far[max_lgh_c];
    incach_c      = new element_c far[max_loaded_c];
    assert(cach_table_c && incach_c);
    for (int i = 0; i < max_lgh_c; i++) cach_table_c[i] = 0;
    for (i = 0; i < max_loaded_c; i++)
        {
        incach_c[i].offset = NULL;
        incach_c[i].index  = 0;
        }
    }

template<class T> CSwap<T>::CSwap (int max_lgh, int max_loaded)
    {
    last_loaded_c = 0;
    max_lgh_c     = max_lgh;
    file_offset_c = SwapMng.NewSwap(max_lgh_c, sizeof(T));
    loaded_c      = 0;
    max_loaded_c  = max_loaded;
    cach_table_c  = new unsigned int far[max_lgh];
    incach_c      = new element_c far[max_loaded];
    assert(cach_table_c && incach_c);
    for (int i = 0; i < max_lgh; i++) cach_table_c[i] = 0;
    for (i = 0; i < max_loaded; i++)
        {
        incach_c[i].offset = NULL;
        incach_c[i].index  = 0;
        }
    }

template<class T> void CSwap<T>::SetOffset (T& dist, void* scr = NULL, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].offset == (void*)&dist)
            {
            if(del)
                delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
            }
    }

template<class T> void CSwap<T>::SetOffset3 (int index1 = 0, void* scr, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].index == index1)
            {
            if (del) delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
```

```
            }
    }

template<class T> void CSwap<T>::SetOffset2 (int index1 = 0, int index2 = 0, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].index == index1)
            {
            for (int j = 0; j < loaded_c; j++)
            if(incach_c[j].index == index2)
                {
                if (del) delete incach_c[i].offset;
                incach_c[i].offset = incach_c[j].offset;
                return;
                }
            }
    }

template<class T> void CSwap<T>::SetOffset1 (void* dist, void* scr = NULL, int del = 1)
    {
    for (int i = 0; i < loaded_c; i++)
        if(incach_c[i].offset == (void*)dist)
            {
            if(del) delete incach_c[i].offset;
            incach_c[i].offset = scr;
            return;
            }
    }

template<class T> T& CSwap<T>::operator[] (int index)
    {
    if (cach_table_c[index] < 0xfffe) cach_table_c[index]++;
    for (int count = 0; count < loaded_c; count++)
        {
        if(incach_c[count].index == index)
            {
            T& ret = *((T*)incach_c[count].offset);
            return ret;
            }
        }
    void *temp = new T;
    while (!temp && !loaded_c)
        {
        SwapMin();
        temp = new T;
        }
    while (loaded_c >= max_loaded_c) SwapMin();
    assert(SwapMng.InMem(temp, 1, file_offset_c + (long)index * (long)sizeof(T), sizeof(T)));
    incach_c[loaded_c].index  = index;
    last_loaded_c             = index;
    incach_c[loaded_c].offset = temp;
    loaded_c++;
    T& ret = *((T*)temp);
    return ret;
    }

template<class T> CSwap<T>::~CSwap()
    {
    SwapMng.DelSwap(file_offset_c);
    for (int i = 0; i < loaded_c; i++)
        {
        delete incach_c[i].offset;
        }
    delete cach_table_c;
    delete incach_c;
    }

template<class T> void CSwap<T>::SwapMin()
    {
    int min     = 0xffff;
    int min_num = 0;
    for (int i = 0; i < loaded_c; i++)
        {
        if (cach_table_c[incach_c[i].index] < min)
            {
            if (incach_c[i].index == last_loaded_c)
                continue;
            min = cach_table_c[incach_c[i].index];
            min_num = i;
            }
        }
    assert(SwapMng.Swap(incach_c[min_num].offset, 1,
        file_offset_c + (long)incach_c[min_num].index * (long)sizeof(T), sizeof(T)));
    delete incach_c[min_num].offset;
    for (i = min_num;i < loaded_c; i++) incach_c[i] = incach_c[i+1];
    loaded_c--;
    incach_c[loaded_c].offset = NULL;
```

```
    incach_c[loaded_c].index  = 0;
    }

//EOF
#endif __CSWAP_H__
```

File Core\CSwapMng.h

```
#ifndef __CSWAPMNG_H__
#define __CSWAPMNG_H__
//===REMARCS=================================================================
//errorlevel 1 - can not create swap file
#define er1 1

//===INCLUDES================================================================
#include <io.h>
#include <dos.h>
#include <fcntl.h>
#include <process.h>
#include <stdio.h>
#include <SYS\STAT.H>
#ifndef NULL
#define NULL 0
#endif

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

//===DEFINITION==============================================================
struct swap_node
    {
    swap_node* next;
    long       offset;
    long       data_lgh;
    };

class CSwapMng
{

//---INTERFACE---
public:
      CSwapMng       (const char *filename);
long  NewSwap        (unsigned int lgh, unsigned int size);
int   Swap           (void far *obj, int lgh, long pointer, size_t size);
int   InMem          (void far *buf, int lgh, long pointer, size_t size);
void  DelSwap        (long file_offset);
      ~CSwapMng      ();

//---ATRIBUTES---
private:
swap_node      *head_c;
const char     *filename_c;
int            swap_handler_c;
unsigned long  last_pointer_c;
};

//--REALISATION---
CSwapMng::CSwapMng (const char* filename = "swap.swp")
    {
    last_pointer_c = 0;
    filename_c = filename;

    remove(filename);
    if (_dos_creat(filename, _A_NORMAL, &swap_handler_c) != 0)
        {
        printf ("Can not create swap file\n");
        exit(er1);
        }

    head_c           = new swap_node;
    head_c->next     = 0;
    head_c->offset   = 0;
    head_c->data_lgh = 0;
    }

void CSwapMng::DelSwap (long file_offset)
    {
    swap_node* node  = head_c;
```

```
    swap_node* node1 = head_c;

    while (node->offset != file_offset) node = node->next;
    while (node1->next != node) node1 = node1->next;
    node1->next = node->next;
    delete node;
    }

long CSwapMng::NewSwap (unsigned int lgh = 0, unsigned int size = 0)
    {
    swap_node* node = head_c;
    long data_lgh   = (long)lgh*(long)size;
    while (node->next != 0)
        {
        if (node->next->offset - (node->offset + node->data_lgh) >= data_lgh)
            break;
        node = node->next;
        }
    swap_node* new_node = new swap_node;
    new_node->offset    = node->offset + node->data_lgh + 1;
    new_node->data_lgh  = data_lgh;
    new_node->next      = node->next;
    node->next          = new_node;
    return new_node->offset;
    }

int CSwapMng::Swap (void far* obj = NULL, int lgh = 0, long pointer = 0, size_t size = 0)
    {
    unsigned num = 0;
    if (lseek      (swap_handler_c, pointer, SEEK_SET) == -1l) return FALSE;
    if (_dos_write (swap_handler_c, obj, lgh*size, &num) != 0) return FALSE;
    return TRUE;
    }

int CSwapMng::InMem (void far* buf = NULL, int lgh = 0, long pointer = 0, size_t size = 0)
    {
    unsigned num = 0;
    if (lseek (swap_handler_c, pointer, SEEK_SET) == -1l) return FALSE;
    if (_dos_read (swap_handler_c, buf, lgh*size, &num) != 0)  return FALSE;
    return TRUE;
    }

CSwapMng::~CSwapMng()
    {
    lseek  (swap_handler_c, last_pointer_c, SEEK_SET);
    _write (swap_handler_c, "\0", 0);
    close  (swap_handler_c);
    remove (filename_c);
    swap_node* node  = head_c;
    swap_node* node1 = head_c;
    while(node != 0)
        {
        node1 = node->next;
        delete node;
        node  = node1;
        }
    }

CSwapMng SwapMng/*("core00.swp")*/;
//EOF
#endif __CSWAPMNG_H__
```

File Core\DebugWnd.cpp

```
#include <SYS\STAT.H>
//=============================================================================
class CEditData : public CChildWnd
{
private:
char *buf_c;
int  x_start_c, y_start_c, cur_pos_c;

public:
int     size_c;
        CEditData       (char*, int, int, int);
void    ReDraw          (int x, int y);
void    ReDrawAct       (int x, int y);
void    OnKeyDo     (int c);
char*GetEdit    ();
};
//=============================================================================
class CCompactList : public CChildWnd
{
```

```
protected:
CButton* element_c [MAX_LIST_COL] [SCR_ROW];
int      x_start_c, y_start_c,
          count_c,   max_y_c,
          size_c,
          max_x_c,   pg_f_c;
CWindow* parent_c;

public:
int   current_c;

      CCompactList  (int,int,int,int,int,CWindow*,int);
void  Clear         ();
void  Add           (char*,int);
void  OnKeyDo       (int);
void  ReDrawAct     (int,int);
void  ReDraw        (int,int);
char  *GetCurrent   ();
      ~CCompactList ();
};
//=============================================================================
class CDebugWnd : public CContainer
{
private:
CCompactList *code_c,
              *sp_reg_c,
              *reg_c,
              *stak_c,
              *eeprom_c;

int           ip_c,
              drawed_c,
              next_c,
              prog_pointer,
              file_c,
              dis_c,
              cod_c,
              step_over_c,
              work_c;

char         *file_name_c,
             *dis_name_c,
             *cod_name_c;

public:
void edit_data     (const void* that, void* but, int x, int y, int c);
void nextpg       ();
void prevpg       ();
void WndFrameDraw ();
void OnKeyDo      (int c);
void ReDraw       ();
     CDebugWnd    (char *file);
     ~CDebugWnd   ();

private:
void FullRegs     (int handler);
void FullSpRegs   (int handler);
void FullEeprom   (int handler);
void FullStack    (int handler);
void FullWnd      ();
void Work          ();
void Ready        ();
};

//=============================================================================
void CDebugWnd::Work()
    {
    work_c++;
    char work_str[] =  "  RUN  ";
    Scr.PutStr(work_str, SCR_ROW-7, 1, 15, 4);
    }

void CDebugWnd::Ready()
    {
    work_c--;
    char ready_str[] = " READY ";
    if (!work_c) Scr.PutStr(ready_str, SCR_ROW-7, 1, 15, 2);
    }

void CDebugWnd::edit_data (const void* that, void* but, int x, int y, int c)
    {
    CButton* button = (CButton*) but;
    if (that == code_c) return;
    CEditData* wnd = NULL;

    if (that == sp_reg_c)
        {
```

```
        wnd = NEW (CEditData(button->string_c+8, x+8, y, 2));
        }

    if (that == reg_c || that == eeprom_c)
        {
        if (y == 14 && x >= 10 && x <= 21) return;
        wnd = NEW (CEditData(button->string_c+1, x+1, y, 2));
        }

    if (that == stak_c)
        {
        wnd = NEW (CEditData(button->string_c, x, y, 4));
        }
    Add(wnd);
    ChildReDraw();
    if (c == KBD_ENTER) c = 's';
    while(1)
        {
        switch (c)
            {
            case KBD_ENTER: case KBD_DOWN: case KBD_UP:
            case KBD_PGUP:  case KBD_PGDOWN:

                char* buf = wnd->GetEdit();
                char* end = buf + wnd->size_c;
                int num = strtol(buf, &(end), 16);
                int out = open("out.sim", O_RDWR);
                int pos = 0;
                if (that == sp_reg_c)
                    {
                    pos = (y - 1);

                    if (pos == 0  || pos == 2 || pos == 3 ||
                        pos == 10 || pos == 9 || pos == 4)
                        {
                        pos += 24;
                        lseek(out, pos, SEEK_SET);
                        _write(out, &num, wnd->size_c/2);
                        pos += 71 - 24;
                        lseek(out, pos, SEEK_SET);
                        _write(out, &num, wnd->size_c/2);
                        pos -= 71;
                        }

                    if (x >= 68 && pos != 11) pos += 71;
                    else if (pos != 11) pos += 24;
                    if (pos == 11 && x >= 68) pos = 5;
                    if (pos == 11 && x < 68) pos = 6;
                    }
                if (that == reg_c)
                    {
                    pos = (x - 10)/3 + (y - 14)*8 - 4;
                    if (pos > 34) pos += 82 - 34;
                    else pos += 36;
                    }
                if (that == eeprom_c)
                    {
                    pos = 122 + (y - 14)*8 + (x - 43)/3;
                    }
                if (that == stak_c)
                    {
                    pos = (y - 14)*2 + 7;
                    }
                if (out != -1)
                    {
                    lseek(out, pos, SEEK_SET);
                    _write(out, &num, wnd->size_c/2);
                    close(out);
                    }
                Del(wnd);
                DEL(wnd);
                Scr.SetCur();
                return;

            default: wnd->OnKeyDo(c);break;
            }
        wnd->ReDraw(x_c, y_c);
        c = Kbd.WaitKey();
        }

    }

CDebugWnd::~CDebugWnd ()
    {
    DEL (code_c);
    DEL (sp_reg_c);
    DEL (reg_c);
```

```
    DEL (eeprom_c);
    DEL (stak_c);
    remove ("command.");
    remove ("in.sim");
    remove ("out.sim");
    remove (dis_name_c);
    remove (cod_name_c);
    DEL (dis_name_c);
    DEL (cod_name_c);
    DEL (title_c);
    }

void CDebugWnd::OnKeyDo (int c = 0)
    {
    switch (c)
        {
        case KBD_F7:    Work();
                        remove ("in.sim");
                        rename ("out.sim", "in.sim");
                        lseek (cod_c, ip_c*2, SEEK_SET);
                        int com = 0;
                        _read (cod_c, &com, 2);
                        int handler = open("command." ,O_CREAT|O_BINARY|O_RDWR, S_IREAD|S_IWRITE);
                        lseek (handler, 0, SEEK_SET);
                        _write (handler, &com, 2);
                        close(handler);
                        if ((spawnlp(P_WAIT, "sim.com", NULL)) == -1)
                            {
                            CWindow* wnd = NEW (CWarning("      Can not exec SIM.COM      "," Error "));
                            prog->AddWnd(wnd);
                            }
                        handler = open("out.sim", O_RDWR);
                        int ip = 0;
                        if (handler != -1)
                            {
                            lseek (handler, 1, SEEK_SET);
                            _read (handler, &ip, 2);
                            ip_c = ip;
                            close(handler);
                            }
                        nextpg();
                        while (ip_c > (drawed_c-12)) nextpg();
                        while (ip_c < (drawed_c-12)) prevpg();
                        while (ip_c > (drawed_c-12+code_c->current_c))
                            code_c->OnKeyDo(KBD_DOWN);
                        FullWnd();
                        Ready();
                        if (step_over_c)
                            {
                            char is_key = 0;
                            asm {    mov ah,0x01
                                     int 0x16
                                     jz  end_
                                     mov is_key,0x01 }
end:                        if (is_key)
                                {
                                int key = Kbd.WaitKey();
                                if(key == KBD_ESC) step_over_c = 0;
                                }
                            }
                        return;

        case KBD_F8:    lseek (dis_c, ip_c*16, SEEK_SET);
                        char* comm = NEW(char[16]);
                        _read (dis_c, comm, 16);
                        if (strstr(comm,"return") != 0 ||
                            strstr(comm,"goto") != 0 ||
                            strstr(comm,"btfss") != 0 ||
                            strstr(comm,"btfsc") != 0)
                                {
                                OnKeyDo(KBD_F7);
                                DEL(comm);
                                break;
                                }
                        DEL(comm);
                        Work();
                        Scr.SetBaseAdress((char far*)0xba000000L);
                        ip = ip_c+1;
                        step_over_c = 1;
                        while (ip_c != ip && step_over_c) OnKeyDo(KBD_F7);
                        step_over_c = 0;
                        prog->ReDraw();
                        Ready();
                        break;

        case KBD_F4:    Work();
                        step_over_c = 1;
```

```
                        Scr.SetBaseAdress((char far*)0xba000000L);
                        ip = code_c->current_c + drawed_c - 12;
                        while (ip_c != ip && step_over_c) OnKeyDo(KBD_F7);
                        step_over_c = 0;
                        prog->ReDraw();
                        Ready();
                        break;

        case KBD_TAB:   CChildWnd* temp = child_c [0];
                        for (int i = 0; i <= count_c-1; i++)
                            child_c [i] = child_c [i+1];
                        child_c [count_c] = temp;
                        break;

        default:     child_c[0]->OnKeyDo(c);break;
        }
    if (child_c[0]) ChildReDraw();
    }


void CDebugWnd::nextpg ()
    {
    code_c->Clear();
    char* command = NEW (char [56]);
    char* com = NEW (char [15]);
    char* str = NEW (char [5]);
    for (int i = 0; i <= 12 && drawed_c+i < 0x400; i++)
        {
        *command = 0;
        if (i+drawed_c == ip_c) strcat(command, " □");
        else strcat(command, "  ");
        strcat(command, "  ");
        itoa (i+drawed_c, str, HEX);
        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int l = 0; l <= k; l++) command[5-l] = str[k-l];
        for (;l < 4; l++) command[5-l] = '0';
        command[6] = 0;
        lseek (dis_c, 16*(i+drawed_c), 0);
        if (_read (dis_c, com, 15) == -1) strcat(command,":          nop             ");
        else
            {
            com[15] = 0;
            strcat(command,":        ");
            strcat(command,com);
            }
        code_c->Add(command ,27);
        }
    int j = 0;
    for (; i <= 12; i++,j++)
        {
        *command = 0;
        if (j == ip_c) strcat(command, " □");
        else strcat(command, "  ");

        itoa (j, str, HEX);
        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int l = 0; l <= k; l++) command[5-l] = str[k-l];
        for (;l < 4; l++) command[5-l] = '0';
        command[6] = 0;
        lseek (dis_c, 16*(j), 0);
        if (_read (dis_c, com, 15) == -1) strcat(command,":          nop             ");
        else
            {
            *(com+15) = 0;
            strcat(command,":        ");
            strcat(command,com);
            }
        code_c->Add(command ,27);
        drawed_c = j-12;
        }
    DEL(str);
    DEL(command);
    DEL(com);
    drawed_c += 12;
    if (drawed_c > 0x3ff) drawed_c = 0;
    }

void CDebugWnd::prevpg ()
    {
    code_c->Clear();
    char* command = NEW (char [56]);
    char* com     = NEW (char [15]);
    char* str      = NEW (char [5]);
    for (int i = 0; drawed_c+i-24 < 0; i++)
        {
```

```
        *command = 0;
        if (0x400-(24-drawed_c)+i == ip_c) strcat(command, " □");
        else strcat(command, "  ");
        strcat(command, "  ");
        itoa (0x400-(24-drawed_c)+i, str, HEX);
        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int l = 0; l <= k; l++) command[5-l] = str[k-l];
        for (;l < 4; l++) command[5-l] = '0';
        command[6] = 0;
        lseek (dis_c, 16*(0x400-(24-drawed_c)+i), 0);
        if (_read (dis_c, com, 15) == -1) strcat(command,":          nop            ");
        else
            {
            *(com+15) = 0;
            strcat(command,":        ");
            strcat(command,com);
            }
        code_c->Add(command ,27);
        if (i > 12) break;
        }
    for (; i <= 12; i++)
        {
        *command = 0;
        if (i+drawed_c-24 == ip_c) strcat(command, " □");
        else strcat(command, "  ");
        itoa (i+drawed_c-24, str, HEX);
        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int l = 0; l <= k; l++) command[5-l] = str[k-l];
        for (;l < 4; l++) command[5-l] = '0';
        command[6] = 0;
        lseek (dis_c, 16*(i+drawed_c-24), 0);
        if (_read (dis_c, com, 15) == -1) strcat(command,":          nop            ");
        else
            {
            *(com+15) = 0;
            strcat(command,":        ");
            strcat(command,com);
            }
        code_c->Add(command ,27);
        }
    drawed_c -= 12;
    if (drawed_c < 0) drawed_c = 0x3ff+drawed_c+1;
    DEL(com);
    DEL(command);
    DEL(str);
    }

CDebugWnd::CDebugWnd (char* file) : CContainer(0, 1, SCR_ROW-1, SCR_STRING-1)
    {
    title_c = NEW(char[10]);
    strcat(title_c,"Debug PIC\0");
    step_over_c = 0;
    file_name_c = file;
    file_c      = _open (file,O_RDONLY);
    ip_c        = 0;
    drawed_c    = 0;
    work_c      = 0;

    OutText ("¡----PIC16F84-04-----------------------------------------------------------------“",0,0,SCR_ROW);
    OutText ("'                                        ‹ indr   00  indr   00 '",0,1,SCR_ROW);
    OutText ("'                                        ‹ rtcc   00  option 00 '",0,2,SCR_ROW);
    OutText ("'                                        ‹ pcl    00  pcl    00 '",0,3,SCR_ROW);
    OutText ("'                                        ‹ status 00  status 00 '",0,4,SCR_ROW);
    OutText ("'                                        ‹ fsr    00  fsr    00 '",0,5,SCR_ROW);
    OutText ("'                                        ‹ porta  00  trisa  00 '",0,6,SCR_ROW);
    OutText ("'                                        ‹ portb  00  trisb  00 '",0,7,SCR_ROW);
    OutText ("'                                        ‹ eedata 00  eecon1 00 '",0,8,SCR_ROW);
    OutText ("'                                        ‹ eeadr  00  eecon2 00 '",0,9,SCR_ROW);
    OutText ("'                                        ‹ pch    00  pch    00 '",0,10,SCR_ROW);
    OutText ("'                                        ‹ intcon 00  intcon 00 '",0,11,SCR_ROW);
    OutText ("'                                        ‹ W      00            '",0,12,SCR_ROW);
    OutText ("'regs-----------------------------------rom-----------šstack---6ə",0,13,SCR_ROW);
    OutText ("' 08(88):  FF FF FF FF FF FF FF FF   ‹ 00:  FF FF FF FF FF FF FF FF   ‹       '",0,14,SCR_ROW);
    OutText ("' 10(90):  FF FF FF FF FF FF FF FF   ‹ 08:  FF FF FF FF FF FF FF FF   ‹       '",0,15,SCR_ROW);
    OutText ("' 18(98):  FF FF FF FF FF FF FF FF   ‹ 10:  FF FF FF FF FF FF FF FF   ‹       '",0,16,SCR_ROW);
    OutText ("' 20(A0):  FF FF FF FF FF FF FF FF   ‹ 18:  FF FF FF FF FF FF FF FF   ‹       '",0,17,SCR_ROW);
    OutText ("' 28(A8):  FF FF FF FF FF FF FF FF   ‹ 20:  FF FF FF FF FF FF FF FF   ‹       '",0,18,SCR_ROW);
    OutText ("' 30(B0):  FF FF FF FF FF FF FF FF   ‹ 28:  FF FF FF FF FF FF FF FF   ‹       '",0,19,SCR_ROW);
    OutText ("' 38(B8):  FF FF FF FF FF FF FF FF   ‹ 30:  FF FF FF FF FF FF FF FF   ‹       '",0,20,SCR_ROW);
    OutText ("' 40(C0):  FF FF FF FF FF FF FF FF   ‹ 38:  FF FF FF FF FF FF FF FF   ‹       '",0,21,SCR_ROW);
    OutText ("' 48(C8):  FF FF FF FF FF FF FF FF   ‹                                 ‹       '",0,22,SCR_ROW);
    OutText ("-------------------------------------------------------------------""",0,23,SCR_ROW);

    code_c   = NEW (CCompactList (1, 1, 12,55,1,this,1));
    sp_reg_c = NEW (CCompactList (57,1, 12,11,2,this,0));
```

```
    reg_c    = NEW (CCompactList (10,14,9, 3, 8,this,0));
    eeprom_c = NEW (CCompactList (43,14,8, 3, 8,this,0));
    stak_c   = NEW (CCompactList (73,14,8, 4, 1,this,0));
    Add(code_c);
    Add(sp_reg_c);
    Add(reg_c);
    Add(eeprom_c);
    Add(stak_c);

    spawnlp (P_WAIT, "sim_dis.exe", "", file, NULL);
    char* dis_f = NEW (char [MAX_FILE_NAME]);
    for (int i = 0; i < (MAX_FILE_NAME-3); i++)
        {
        *(dis_f + i) = *(file + i);
        if (*(file + i) == '.' || *(file + i) == 0) break;
        }
    if (*(dis_f + i) == 0) *(dis_f + i++) = '.';
    else i++;
    *(dis_f + i++) = 'd';
    *(dis_f + i++) = 'i';
    *(dis_f + i++) = 's';
    *(dis_f + i++) = 0;
    dis_c = _open (dis_f,O_RDONLY);
    dis_name_c = NEW (char [MAX_FILE_NAME]);
    cod_name_c = NEW (char [MAX_FILE_NAME]);

    for (i = 0; dis_f[i-1] != 0; i++) dis_name_c[i] = dis_f[i];
    for (i = 0; i < (MAX_FILE_NAME-3); i++)
        {
        *(dis_f + i) = *(file + i);
        if (*(file + i) == '.' || *(file + i) == 0) break;
        }
    if (*(dis_f + i) == 0) *(dis_f + i++) = '.';
    else i++;
    *(dis_f + i++) = 'c';
    *(dis_f + i++) = 'o';
    *(dis_f + i++) = 'd';
    *(dis_f + i++) = 0;
    cod_c = _open (dis_f,O_RDONLY);
    for (i = 0; dis_f[i-1] != 0; i++) cod_name_c[i] = dis_f[i];

    int in = open ("out.sim", O_CREAT | O_BINARY | O_RDWR, S_IREAD | S_IWRITE);
    if (in != -1)
        {
        char  init_data[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0xff, 0x00, 0x18, 0x00, 0xff, 0x1f, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
        if (lseek(cod_c, 0x2007, SEEK_SET) != -1l)
            {
            _read(cod_c, &init_data[3], 1);
            _read(cod_c, &init_data[4], 1);
            }
        if (lseek(cod_c, 0x2100, SEEK_SET) != -1l)
            {
            for (int i = 0; i < 64; i++)
                {
                char c = 0;
                if (_read(cod_c, &c, 1) <= 0) break;
                init_data[122 + i] = c;
                }
            }
        _write (in, &init_data[0], 189);
        _write (in, 0, 0);
        close (in);
        }
    nextpg();
    FullWnd();
    }

void CDebugWnd::FullWnd()
    {
```

```
    int handler = 0;
    handler = open("out.sim", O_RDWR);
    FullRegs  (handler);
    FullSpRegs(handler);
    FullEeprom(handler);
    FullStack (handler);
    if (handler != -1) close (handler);
    ChildReDraw();
    }

void CDebugWnd::ReDraw ()
    {
    WndFrameDraw();
    DrawText();
    if (child_c[0] != NULL) ChildReDraw();
    work_c++;
    Ready();
    Scr.TextColor (0);
    Scr.BackColor (3);
    }

void CDebugWnd::FullRegs(int handler)
    {
    reg_c->Clear();
    if (handler == -1)
        {
        for (int i = 0; i < 72; i++) reg_c->Add(" FF",3);
        return;
        }
    short num    = 0;
    char  *str   = NEW(char[3]);
    char  *str_t = NEW(char[4]);
    int adr0 = 36;
    for (int i = 0; i < 72; i++)
        {
        if ((i%9*8 + i/9-4) > 34) adr0 = (82-34);
        else adr0 = 36;
        if (i%9*8 + i/9 - 4 < 0)
            {
            reg_c->Add (" - ", 3);
            continue;
            }
        lseek (handler, (adr0 + i%9*8 + i/9 - 4), SEEK_SET);
        _read (handler, &num, 1);
        *str = *str_t = 0;
        itoa (num, str, HEX);
        for (int k = 0; k < 2; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[2-j] = str[k-j];
        for (;j < 2; j++) str_t[2 - j] = '0';
        str_t[0] = ' ';
        reg_c->Add (str_t, 3);
        }
    DEL (str);
    DEL (str_t);
    }

void CDebugWnd::FullSpRegs(int handler)
    {
    sp_reg_c->Clear();

    char* sp_reg[] = {
    "indr   ",
    "rtcc   ",
    "pcl    ",
    "status ",
    "fsr    ",
    "porta  ",
    "portb  ",
    "eedata ",
    "eeadr  ",
    "pch    ",
    "intcon ",
    "W      ",
    "indr   ",
    "option ",
    "pcl    ",
    "status ",
    "fsr    ",
    "trisa  ",
    "trisb  ",
    "eecon1 ",
    "eecon2 ",
    "pch    ",
    "intcon ",
    "mlsr   ",
    };
```

```
    if (handler == -1)
        {
        char str [12];
        for (int i = 0; i < 24; i++)
            {
            strcpy (str, sp_reg[i]);
            strcat (str, " 00 ");
            sp_reg_c->Add(str,10);
            }
        return;
        }
    char  str_  [12];
    short num   = 0;
    char  str   [3];
    char  str_t [4];
    int   adr0  = 24;
    for (int i = 0; i < 24; i++)
        {
        strcpy (str_, sp_reg[i]);
        strcat (str_, " ");
        if (i >  6)  adr0 = (32 - 7);
        if (i == 11) adr0 = (6  - 11);
        if (i >  11) adr0 = (71 - 12);
        if (i >  18) adr0 = (78 - 18);
        if (i == 23) adr0 = (5  - 23);
        lseek (handler, (adr0 + i), SEEK_SET);
        _read (handler, &num, 1);
        itoa (num, str, HEX);
        for (int k = 0; k < 2; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[1-j] = str[k-j];
        for (;j < 2; j++) str_t[1 - j] = '0';
        str_t[2] = 0;
        strcat (str_, str_t);
        strcat (str_, " ");
        sp_reg_c->Add(str_, 10);
        }
    }

void CDebugWnd::FullEeprom(int handler)
    {
    eeprom_c->Clear();
    if (handler == -1)
        {
        for (int i = 0; i < 64; i++) eeprom_c->Add(" FF",3);
        return;
        }
    short num    = 0;
    char  *str   = NEW(char[3]);
    char  *str_t = NEW(char[4]);
    for (int i = 0; i < 64; i++)
        {
        lseek (handler, (122 + i%8*8 + i/8), SEEK_SET);
        _read (handler, &num, 1);
        *str = *str_t = 0;
        itoa (num, str, HEX);

        for (int k = 0; k < 2; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[2-j] = str[k-j];
        for (;j < 2; j++) str_t[2 - j] = '0';
        str_t[0] = ' ';

        eeprom_c->Add (str_t, 3);
        }
    DEL (str);
    DEL (str_t);
    }

void CDebugWnd::FullStack(int handler)
    {
    stak_c->Clear();
    if (handler == -1)
        {
        for (int i = 0; i < 8; i++)
            {
            stak_c->Add("0000",4);
            OutText(" ",72,14+i,1);
            }
        OutText("□",72,14,1);
        return;
        }
    lseek (handler, 7l, SEEK_SET);
    char *str   = NEW (char[5]);
    char *str_t = NEW (char[5]);
    int adr     = 0;
```

```
    for (int i = 0; i < 8; i++)
        {
        _read (handler, &adr, 2);
        *str = *str_t = 0;
        itoa (adr, str, HEX);

        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[3-j] = str[k-j];
        for (;j < 4; j++) str_t[3 - j] = '0';
        str_t[4] = 0;
        stak_c->Add(str_t ,4);

        OutText(" ",72,14+i,1);
        }
    char pos = 0;
    lseek (handler, 23, SEEK_SET);
    _read (handler, &pos, 1);
    if (pos <= 8 && pos > 0) OutText("□",72,13+pos,1);
    if (pos < 0)
        {
        lseek (handler, 23, SEEK_SET);
        pos = 0;
        _write(handler, &pos, 1);
        CWindow* wnd = NEW (CWarning("          Stack underflow         "," Error "));
        prog->AddWnd(wnd);
        }
    if (pos > 8)
        {
        lseek (handler, 23, SEEK_SET);
        pos = 8;
        _write(handler, &pos, 1);
        CWindow* wnd = NEW (CWarning("          Stack overflow          "," Error "));
        prog->AddWnd(wnd);
        }
    DEL (str_t);
    DEL (str);
    }

void CDebugWnd::WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, TRUE, TRUE, 0 , 7, 0);
    Scr.TextColor (0);
    Scr.BackColor (3);
    for (int i = x_c+1; i < x_end_c; i++)
        {
        for (int j = y_c+1; j < y_end_c; j++)
            {
            Scr.Put (i,j);
            }
        }
    }
//=============================================================================
CCompactList::CCompactList (int x_start = 0, int y_start = 0, int max_y = 0, int size = 0,
    int max_x = 0, CWindow* parent = NULL,int pg_f = 1)
    {
    parent_c  = parent;
    count_c   = -1;
    current_c = 0;
    x_start_c = x_start;
    y_start_c = y_start;
    max_y_c   = max_y;
    max_x_c   = max_x;
    size_c    = size;
    pg_f_c    = pg_f;
    }

void CCompactList::Clear ()
    {
    for (int i = 0; i <= (count_c % max_y_c); i++)
        {
        DEL (element_c [count_c / max_y_c] [i]->string_c);
        DEL (element_c [count_c / max_y_c] [i]);
        }
    for (i = 0; i <= ((count_c / max_y_c) - 1); i++)
        for (int j = 0; j < max_y_c; j++)
            {
            DEL (element_c [i] [j]->string_c);
            DEL (element_c [i] [j]);
            }
    count_c   = -1;
    current_c = 0;
    }

void CCompactList::Add (char* where = NULL, int count = 0)
    {
    count_c++;
```

```
    char* str = NEW (char [size_c]);
    count = count > size_c ? size_c : count;
    for (int i = 0; i < count; i++) *(str + i) = *(where + i);
    if (i < size_c) for (; i <= size_c; i++) *(str + i) = ' ';
    element_c [count_c / max_y_c] [count_c % max_y_c] =
    NEW (CButton ((count_c / max_y_c) * size_c, count_c % max_y_c,
    size_c, str,0,0,0,3,15,15,2 ));
    if (element_c [count_c / max_y_c] [count_c % max_y_c] == NULL)
        {
        count_c--;
        }
    }

void CCompactList::OnKeyDo (int c = 0)
    {
    switch (c)
        {
        case KBD_DOWN  : current_c++;break;
        case KBD_UP    : current_c--;break;
        case KBD_RIGHT : current_c += max_y_c;break;
        case KBD_LEFT  : current_c -= max_y_c;break;
        case KBD_PGUP  : current_c -= max_y_c*max_x_c;break;
        case KBD_PGDOWN: current_c += max_y_c*max_x_c;break;
        case '1': case '2': case '3': case '4': case '5': case '6':
        case '7': case '8': case '9': case '0': case 'a': case 'b':
        case 'c': case 'd': case 'e': case 'f': case 'A': case 'B':
        case 'C': case 'D': case 'E': case 'F': case KBD_ENTER:
        parent_c->edit_data(this, element_c[current_c/max_y_c][current_c%max_y_c],
            element_c[current_c/max_y_c][current_c%max_y_c]->x_start_c+x_start_c,
            element_c[current_c/max_y_c][current_c%max_y_c]->y_start_c+y_start_c, c);break;
        default : break;
        }
    if (current_c > count_c) current_c = count_c;
    if (current_c - (max_y_c*max_x_c) >= 0 && pg_f_c)
         {
         parent_c->nextpg();
         return;
         }
    if (current_c < 0 && pg_f_c)
        {
        parent_c->prevpg();
        return;
        }
    if (current_c < 0) current_c = 0;
    }

void CCompactList::ReDrawAct (int x = 0, int y = 0)
    {
    if (count_c == -1) return;
    for (int i = 0; i < max_y_c*max_x_c; i++)
        {
        if (i > count_c)
            {
            Scr.BackColor(3);
            for (;i < max_y_c*max_x_c; i++)
                {
                for (int j = 0; j < size_c; j++) Scr.Put(x+x_start_c+(i /
    max_y_c)*size_c+j,y+y_start_c+(i % max_y_c));
                }
            Scr.BackColor(7);
            break;
            }
        element_c [i / max_y_c] [i % max_y_c]->ReDraw(x+x_start_c,y+y_start_c);
        }
    element_c [current_c / max_y_c] [current_c % max_y_c]->ReDrawAct(x+x_start_c,y+y_start_c);
    }

void CCompactList::ReDraw (int x = 0, int y = 0)
    {
    if (count_c == -1) return;
    for (int i = 0; i < size_c; i++)
        {
        Scr.SetColor(i+x+x_start_c+(current_c / max_y_c)*size_c, y+y_start_c+(current_c %
    max_y_c),14,3);
        }
    for (i = 0; i < max_y_c*max_x_c; i++)
        {
        if (i > count_c)
            {
            Scr.BackColor(3);
            for (;i < max_y_c*max_x_c; i++)
                {
                for (int j = 0; j < size_c; j++) Scr.Put(x+x_start_c+(i /
    max_y_c)*size_c+j,y+y_start_c+(i % max_y_c));
                }
            Scr.BackColor(7);
            break;
```

```
            }
        element_c [i / max_y_c] [i % max_y_c]->ReDraw(x+x_start_c,y+y_start_c);
        }
    }

char* CCompactList::GetCurrent () { return element_c [current_c / max_y_c] [current_c %
    max_y_c]->string_c; }

CCompactList::~CCompactList()
    {
    for (int i = 0; i <= (count_c % max_y_c); i++)
        {
        DEL ((*element_c [count_c / max_y_c] [i]).string_c);
        DEL (element_c [count_c / max_y_c] [i]);
        }

    for (i = 0; i <= ((count_c / max_y_c) - 1); i++)
        for (int j = 0; j < max_y_c; j++)
            {
            DEL ((*element_c [i] [j]).string_c);
            DEL (element_c [i] [j]);
            }
    }
//=========================================================================
CEditData::CEditData (char* buf = NULL, int x_start = 0, int y_start = 0, int size = 2)
    {
    buf_c     = buf;
    cur_pos_c = 0;
    x_start_c = x_start;
    y_start_c = y_start;
    size_c    = size;
    }

void CEditData::ReDraw (int x, int y)
    {
    Scr.BackColor (1);   //!
    Scr.TextColor (15);  //!
    for (int i = 0; i < size_c; i++)
        {
        Scr.Put (x+x_start_c+i,y+y_start_c,buf_c[i]);
        }
    Scr.SetCur (x + x_start_c + cur_pos_c, y+y_start_c, 1);
    }

void CEditData::ReDrawAct (int x, int y)
    {
    ReDraw(x,y);
    }

void CEditData::OnKeyDo (int c)
    {
    switch (c)
        {
        case KBD_LEFT  : cur_pos_c--;break;
        case KBD_RIGHT : cur_pos_c++;break;
        case 'A': case 'B': case 'C': case 'D': case 'E':
        case 'F': case '0': case '1': case '2': case '3':
        case '4': case '5': case '6': case '7': case '8':
        case '9': case 'a': case 'b': case 'c': case 'd':
        case 'e': case 'f':
        *(buf_c + cur_pos_c) = c;cur_pos_c++; break;
        default: break;
        }
    if (cur_pos_c >= size_c) {cur_pos_c = size_c-1;}
    if (cur_pos_c < 0) {cur_pos_c = 0;}
    }

char* CEditData::GetEdit()
    {
    return buf_c;
    }
//=========================================================================
```

File Core\IO.h

```
#ifndef __IO_H__
#define __IO_H__

//===CONSTANTS============================================================
#define NULL    0
#define TRUE    1
#define FALSE   0
#define IO_VER  1
```

```
#define HEX     16
#define BINARY  2

//===INCLUDES=========================================================================
#include "CScreen.h"
CScreen Scr ((char far*) 0xb8000000L);
#include "CKbd.h"
#include "CDraw.h"

//===DEFINITION=======================================================================
CDraw   Draw (&Scr);
CKbd    Kbd;

//EOF
#endif __IO_H__
```

File Core\W.cpp

```
#include "io.h"

//============================================================================
//               Class Tree
//V      CScreen
//V      CDraw
//V      CKbd
//
//V      CProg
//V      CManager
//V      CWindow
//V       |-- CMenu
//V      |-- CContainer
//V       +-- CChildWnd
//V           |------ CScroll
//V           |------ CButton
//V           |------ CList
//V           |------ CEdit
//V           +------ CEditLine
//
//============================================================================
//----------------------------------------------------------------------------
const MOVE_LEFT     = 1;
const MOVE_RIGHT    = 2;
const MOVE_UP       = 3;
const MOVE_DOWN     = 4;
const BIGGER_LEFT   = 5;
const BIGGER_RIGHT  = 6;
const BIGGER_UP     = 7;
const BIGGER_DOWN   = 8;
const CLOSE         = 9;
const FULL_SCR      = 10;

const MAX_FILE_NAME = 13;
const MAX_STR       = 100;
const MAX_COL       = SCR_ROW;
const MAX_EDIT_STR  = 1000;
const MAX_EDIT_SYM  = 256;
const MAX_WND       = 10;
const MAX_MES       = 256;
const MAX_LIST_COL  = 15;

int InsFlag         = 0;
int TAB_LGH         = 8;
//----------------------------------------------------------------------------
void SetPage1 ()
    {
    asm { mov ah,0x05
          mov al,0x01
          int 0x10
        }
    }

void SetPage0 ()
    {
    asm { mov ah,0x05
          mov al,0x00
          int 0x10
        }
    }

//============================================================================
class CWindow
{
public:
```

```
int x_c,     y_c,
    x_end_c, y_end_c, str_c,
    x_old_c, y_old_c,
    x_end_old_c, y_end_old_c;
char *title_c,
     *text_c [MAX_STR];

               CWindow        (int,int,int,int);
void virtual   WndFrameDraw   ();
void virtual   ReDraw      ();
void           DrawText       ();
void           OutText        (char*,int,int,int);
int            testxy      ();
int virtual    ManMes      (int);
void           clear_str   (char*);
void virtual   OnKeyDo         (int) {}
virtual        ~CWindow       ();

int virtual    Save           (char*) {return FALSE;}
void virtual   nextpg      () {}
void virtual   prevpg      () {}
void virtual   edit_data     (const void* that, void* but, int x, int y, int c) {}

};
//---------------------------------------------------------------------------
CWindow::CWindow (int x = 0, int y = 1, int x_end = SCR_ROW, int y_end = SCR_STRING)
    {
    title_c = "";
    x_old_c = 0;
    y_old_c = 1;
    x_end_old_c = SCR_ROW - 1;
    y_end_old_c = SCR_STRING - 1;
    text_c [0] = NEW (char [MAX_COL]);
    clear_str (text_c [0]);
    str_c = 0;
    if ( x >= 0 )     x_c = x;         else x_c = 0;
    if ( y >= 1 )     y_c = y;         else y_c = 1;
    if ( x_end >= 1 ) x_end_c = x_end; else x_end_c = 1;
    if ( y_end >= 2 ) y_end_c = y_end; else y_end_c = 2;
    Scr.SetCur();
    }

void CWindow::WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, TRUE, TRUE, 1 ,7, 1);
    Scr.BackColor (7);
    for (int i = x_c+1; i < x_end_c; i++)
        {
        for (int j = y_c+1; j <y_end_c; j++)
            {
            Scr.Put (i,j);
            }
        }
    }

void CWindow::ReDraw ()
    {
    WndFrameDraw();
    DrawText();
    }

void CWindow::DrawText ()
    {
    int k;
    if (str_c >= (y_end_c - y_c + 1)) k = y_end_c - y_c + 1; else k = str_c;
    for (int i = 0; i <= k; i++)
        {
        if (y_c + i <= y_end_c)
        for (int j = 0; j <=(MAX_COL); j++)
            {
            if ( *(text_c [i] + j) != '□' && x_end_c >= (x_c + j))
                {
                Scr.Put (x_c+j, y_c+i, *(text_c [i] + j));
                }
            }
        }
    }

void CWindow::OutText (char* where, int x, int y, int count)
    {
    while (str_c < y)
        {
        str_c++;
        text_c [str_c] = NEW (char [MAX_COL]);
        clear_str (text_c [str_c]);
        }
```

```
    char* temp = text_c [y] + x;
    count--;
    for (int i = 0; i <= count; i++)
        {
        *(temp + i) = *(where + i);
        }
    }

int CWindow::testxy ()
    {
    if ( x_c     >= 0          &&
         y_c     >= 1          &&
         x_end_c <  SCR_ROW    &&
         y_end_c <  SCR_STRING &&
         x_end_c >  x_c        &&
         y_end_c >  y_c )
        return TRUE;
         else
        return FALSE;
    }

int CWindow::ManMes (int mes)
    {
    switch (mes)
        {
        case MOVE_LEFT:    x_c--; x_end_c--;if (testxy () == TRUE) return TRUE; else x_c++;
    x_end_c++;break;
        case MOVE_RIGHT:   x_c++; x_end_c++;if (testxy () == TRUE) return TRUE; else x_c--;
    x_end_c--;break;
        case MOVE_UP:      y_c--; y_end_c--;if (testxy () == TRUE) return TRUE; else y_c++;
    y_end_c++;break;
        case MOVE_DOWN:    y_c++; y_end_c++;if (testxy () == TRUE) return TRUE; else y_c--;
    y_end_c--;break;
        case BIGGER_LEFT:  x_end_c--; if (testxy () == TRUE) return TRUE; else x_end_c++;break;
        case BIGGER_RIGHT: x_end_c++; if (testxy () == TRUE) return TRUE; else x_end_c--;break;
        case BIGGER_UP:    y_end_c--; if (testxy () == TRUE) return TRUE; else y_end_c++;break;
        case BIGGER_DOWN:  y_end_c++; if (testxy () == TRUE) return TRUE; else y_end_c--;break;
        default:           return FALSE;
        }
    return FALSE;
    }

void CWindow::clear_str (char* str = NULL)
    {
    if (str) for (int i = 0; i < MAX_COL; i++) str [i] = '□';
    }

CWindow::~CWindow()
    {
    for (int i = 0; i <= str_c; i++)
        {
        DEL (text_c [i]);
        }
    }
//============================================================================
class CChildWnd
{
public:
void virtual OnKeyDo   (int)      {}
void virtual ReDrawAct (int, int) {}
void virtual ReDraw    (int, int) {}
void virtual Run       ()         {}
int  virtual HotKey    (int)      {return FALSE;}
virtual      ~CChildWnd()         {}
};
//============================================================================
class CContainer : public CWindow
{
public:

int count_c;
CChildWnd* child_c [MAX_WND];
CContainer () {}
CContainer (int x = 0, int y = 0, int x_end = 0, int y_end = 0) : CWindow (x,y,x_end,y_end)
    {
    for (int i = 0; i < MAX_WND; i++)
        {
        child_c [i] = NULL;
        }
    count_c = -1;
    }

int virtual CContainer::ManMes (int mes)
    {
    switch (mes)
        {
```

```
        case MOVE_LEFT  : x_c--; x_end_c--;if (testxy () == TRUE) return TRUE; else x_c++;
    x_end_c++;break;
        case MOVE_RIGHT : x_c++; x_end_c++;if (testxy () == TRUE) return TRUE; else x_c--;
    x_end_c--;break;
        case MOVE_UP    : y_c--; y_end_c--;if (testxy () == TRUE) return TRUE; else y_c++;
    y_end_c++;break;
        case MOVE_DOWN  : y_c++; y_end_c++;if (testxy () == TRUE) return TRUE; else y_c--;
    y_end_c--;break;
        default:        return FALSE;
        }
    return FALSE;
    }

int Add (CChildWnd* child)
    {
    if ( count_c < MAX_WND )
        {
        count_c++;
        child_c [count_c] = child;
        return TRUE;
        }
    else return FALSE;
    }

void Del (CChildWnd* child)
    {
    for (int i = 0; i <= count_c; i++)
        {
        if (child_c [i] == child)
            {
            for (int j = i; j <= count_c; j++)
                {
                child_c [j] = child_c [j+1];
                }
            }
        }
    count_c--;
    }

void virtual CContainer::ReDraw ()
    {
    WndFrameDraw();
    DrawText();
    if (child_c[0] != NULL) ChildReDraw();
    }

void ChildReDraw ()
    {
    for (int i = count_c; i >= 1; i--) child_c [i]->ReDraw(x_c, y_c);
    child_c[0]->ReDrawAct (x_c, y_c);
    }

void virtual CContainer::OnKeyDo(int c)
    {
    switch (c)
        {
        case KBD_TAB: CChildWnd* temp = child_c [0];
                  for (int i = 0; i <= count_c-1; i++)
                {
                int k = i + 1;
                child_c [i] = child_c [k];
                }
                  child_c [count_c] = temp;
                  break;
        case KBD_ENTER: child_c[0]->Run();break;
        default: child_c[0]->OnKeyDo(c);break;
        }
    if (child_c[0] != NULL) ChildReDraw();
    }

virtual ~CContainer() {}
};
//=========================================================================
class CButton : public CChildWnd
{

//int x_start_c, y_start_c,
int frame_c;
int text_color_c,     s_text_color_c,     back_color_c,
    text_color_act_c, s_text_color_act_c, back_color_act_c;

public:
int x_start_c, y_start_c;
int   count_c;
char* string_c;

CButton (int x_start = 0, int y_start = 0, int count = 0, char* string = NULL, int frame = 1,
```

```
    int text_color     = 0, int s_text_color     = 4, int back_color = 7,
    int text_color_act = 0, int s_text_color_act = 4, int back_color_act = 2)
    {
    x_start_c = x_start;
    y_start_c = y_start;
    count_c   = count;
    string_c  = string;
    frame_c   = frame;

    text_color_c       = text_color;
    s_text_color_c     = s_text_color;
    back_color_c       = back_color;
    text_color_act_c   = text_color_act;
    s_text_color_act_c = s_text_color_act;
    back_color_act_c   = back_color_act;
    }

void ReDraw (int x = 0, int y = 0)
    {
    int back_color = Scr.GetBackColor();
    Scr.BackColor(back_color_c);
    !frame_c ? Draw.Button (x+x_start_c, y+y_start_c, string_c, count_c, text_color_c,
    s_text_color_c) : Draw.FrameButton (x+x_start_c, y+y_start_c,  string_c, count_c, 0,
    text_color_c, s_text_color_c);
    Scr.BackColor(back_color);
    }

void ReDrawAct (int x = 0, int y = 0)
    {
    int back_color = Scr.GetBackColor();
    Scr.BackColor(back_color_act_c);
    !frame_c ? Draw.Button (x+x_start_c, y+y_start_c, string_c, count_c, text_color_act_c,
    s_text_color_act_c) : Draw.FrameButton (x+x_start_c, y+y_start_c,  string_c, count_c, 0,
    text_color_act_c, s_text_color_act_c);
    Scr.BackColor (back_color);
    }

int HotKey (int c)
    {
    for (int i = 0; i <= count_c; i++) if (*(string_c + i) == '~') break;
    if (c == *(string_c + i + 1) || c == *(string_c + i + 1) + 0x20) return 1;
    return 0;
    }
};
//============================================================================
//============================================================================
class CFButton //: public CChildWnd
{

//int x_start_c, y_start_c,
int frame_c;
int text_color_c,     s_text_color_c,     back_color_c,
    text_color_act_c, s_text_color_act_c, back_color_act_c;

public:
int x_start_c, y_start_c;
int   count_c;
char* string_c;

CFButton (int x_start = 0, int y_start = 0, int count = 0, char* string = NULL, int frame = 1,
    int text_color     = 0, int s_text_color     = 4, int back_color = 7,
    int text_color_act = 0, int s_text_color_act = 4, int back_color_act = 2)
    {
    x_start_c = x_start;
    y_start_c = y_start;
    count_c   = count;
    string_c  = string;
    frame_c   = frame;

    text_color_c       = text_color;
    s_text_color_c     = s_text_color;
    back_color_c       = back_color;
    text_color_act_c   = text_color_act;
    s_text_color_act_c = s_text_color_act;
    back_color_act_c   = back_color_act;
    }

void far ReDraw (int x = 0, int y = 0)
    {
    int back_color = Scr.GetBackColor();
    Scr.BackColor(back_color_c);
    !frame_c ? Draw.Button (x+x_start_c, y+y_start_c, string_c, count_c, text_color_c,
    s_text_color_c) : Draw.FrameButton (x+x_start_c, y+y_start_c,  string_c, count_c, 0,
    text_color_c, s_text_color_c);
    Scr.BackColor(back_color);
    }
```

```
void far ReDrawAct (int x = 0, int y = 0)
    {
    int back_color = Scr.GetBackColor();
    Scr.BackColor(back_color_act_c);
    !frame_c ? Draw.Button (x+x_start_c, y+y_start_c, string_c, count_c, text_color_act_c,
    s_text_color_act_c) : Draw.FrameButton (x+x_start_c, y+y_start_c,  string_c, count_c, 0,
    text_color_act_c, s_text_color_act_c);
    Scr.BackColor (back_color);
    }

int far HotKey (int c)
    {
    for (int i = 0; i <= count_c; i++) if (*(string_c + i) == '~') break;
    if (c == *(string_c + i + 1) || c == *(string_c + i + 1) + 0x20) return 1;
    return 0;
    }
};
//=============================================================================
//=============================================================================
class CScrollBar : public CChildWnd
{

int x_start_c, y_start_c,
    count_c,   max_c,
    current_c, dir_c;

public:

CScrollBar (int x_start = 0, int y_start = 0, int count = 0, int dir = HOR, int max = 0, int
    current = 0)
    {
    x_start_c = x_start;
    y_start_c = y_start;
    count_c   = count;
    max_c     = max;
    current_c = current;
    dir_c     = dir;
    }

void Set (int x_start = 0, int y_start = 0, int count = 0, int dir = HOR)
    {
    x_start_c = x_start;
    y_start_c = y_start;
    count_c   = count;
    dir_c     = dir;
    }

void ReDraw (int x = 0, int y = 0)
    {
    Scr.BackColor(3);        //!
    Scr.TextColor(1);       //!
    Draw.Scroll (x+x_start_c, y+y_start_c, count_c, dir_c, max_c, current_c);
    }

void ReDrawAct (int x = 0, int y = 0)
    {
    ReDraw(x,y);
    }

void SetScroll (int current = 0)
    {
    if (current <= max_c) current_c = current;
    }

void SetScrollMax (int max = 0)
    {
    max_c = max;
    }
};
//=============================================================================
//=============================================================================
class CList : public CChildWnd
{
CSwap<CButton> element_c;

int x_start_c, y_start_c,
    count_c,   max_y_c,
    size_c,    current_c,
    max_x_c,   hide_c;

public:

CList (int x_start = 0, int y_start = 0, int max_y = 0, int size = 0, int max_x = 0)
        : element_c(SCR_ROW*MAX_STR,max_y*max_x)
    {
    count_c   = -1;
    current_c = 0;
```

```
    hide_c    = 0;
    x_start_c = x_start;
    y_start_c = y_start;
    max_y_c   = max_y;
    max_x_c   = max_x;
    size_c    = size;
    }

void Clear ()
    {
    if (count_c >= 0)
        {
        for (int i = 0; i <= count_c; i++)
            {
            DEL (element_c[i].string_c);
            }
        }
    count_c   = -1;
    current_c = 0;
    hide_c    = 0;
    }

void Add (char* where = NULL, int count = 0)
    {
    count_c++;
    char* str = NEW (char [size_c]);
    count = count > size_c ? size_c : count;
    for (int i = 0; i < count; i++) *(str + i) = *(where + i);
    if (i < size_c) for (; i <= size_c; i++) *(str + i) = ' ';
    CButton* but = NEW (CButton((count_c / max_y_c) * size_c, count_c % max_y_c,
    size_c, str,0,0,0,3,15,15,2 ));
    element_c.SetOffset(element_c[count_c],but,1);
    }

void OnKeyDo (int c = 0)
    {
    switch (c)
        {
        case KBD_DOWN  : current_c++;break;
        case KBD_UP    : current_c--;break;
        case KBD_RIGHT : current_c += max_y_c;break;
        case KBD_LEFT  : current_c -= max_y_c;break;
        case KBD_PGUP  : current_c -= max_y_c*max_x_c;break;
        case KBD_PGDOWN: current_c += max_y_c*max_x_c;break;
        default : break;
        }
    if (current_c > count_c) current_c = count_c;
    if (current_c < 0) current_c = 0;
    for (int i = 0; i < max_x_c; i++) if ((current_c - (max_y_c*hide_c) - (max_y_c*max_x_c)) >=
    0) hide_c++;
    for (i = 0; i < max_x_c; i++) if ((current_c - (max_y_c*hide_c)) < 0) hide_c--;
    }

void ReDrawAct (int x = 0, int y = 0)
    {
    if (count_c == -1) return;
    for (int i = hide_c*max_y_c; i < max_y_c*(max_x_c+hide_c); i++)
        {
        if (i > count_c)
            {
            Scr.BackColor(3);
            for (;i < max_y_c*(max_x_c+hide_c); i++)
                {
                for (int j = 0; j < size_c; j++) Scr.Put(x+x_start_c+(i / max_y_c -
    hide_c)*size_c+j,y+y_start_c+(i % max_y_c));
                }
            Scr.BackColor(7);
            break;
            }
        element_c[i].ReDraw(x-(size_c*hide_c)+x_start_c,y+y_start_c);
        }
    element_c[current_c].ReDrawAct(x-(size_c*hide_c)+x_start_c,y+y_start_c);
    }

void ReDraw (int x = 0, int y = 0)
    {
    if (count_c == -1) return;
    for (int i = hide_c*max_y_c; i < max_y_c*(max_x_c+hide_c); i++)
        {
        if (i > count_c)
            {
            Scr.BackColor(3);
            for (;i < max_y_c*(max_x_c+hide_c); i++)
                {
                for (int j = 0; j < size_c; j++) Scr.Put(x+x_start_c+(i / max_y_c -
    hide_c)*size_c+j,y+y_start_c+(i % max_y_c));
                }
```

```
            Scr.BackColor(7);
            break;
            }
        element_c[i].ReDraw(x-(size_c*hide_c)+x_start_c,y+y_start_c);
        }
    for (i = 0; i < size_c; i++)
        {
        Scr.SetColor(i+x-(size_c*hide_c)+x_start_c+(current_c / max_y_c)*size_c,
    y+y_start_c+(current_c % max_y_c),14,3);
        }
    }

char* GetCurrent ()
    {
    return element_c [current_c].string_c;
    }

~CList()
    {
    if (count_c >= 0)
        {
        for (int i = 0; i <= count_c; i++)
            {
            DEL (element_c[i].string_c);
            }
        }
    }
};
//============================================================================
class CEdit : public CChildWnd
{
int   count_c [MAX_EDIT_STR], count_y_c,
      x_start_c, y_start_c,
      size_x_c,  size_y_c,
      cur_x_c,   cur_y_c,
      str_x_c,   str_y_c,
      sym_cor_c;
CFSwap<char[MAX_EDIT_SYM]> buf_c;

public:

CEdit (int x_start = 1, int y_start = 1, int size_x = 0, int size_y = 0)
    {
    cur_x_c    = 0;
    cur_y_c    = 0;
    str_x_c    = 0;
    str_y_c    = 0;
    count_c[0] = -1;
    count_y_c  = 0;
    x_start_c  = x_start;
    y_start_c  = y_start;
    size_x_c   = size_x;
    size_y_c   = size_y;
    sym_cor_c  = 0;
    for (int i = 0; i < MAX_EDIT_SYM; i++) *(buf_c[0] + i) = ' ';
    }

void SetSize (int size_x, int size_y)
    {
    size_x_c = size_x;
    size_y_c = size_y;
    }

void ReDrawAct (int x, int y)
    {
    ReDraw(x,y);
    Scr.SetCur (x+x_start_c+cur_x_c,y+y_start_c+cur_y_c, InsFlag==1?2:1);
    }

int GetX () { return (str_x_c+cur_x_c); }
int GetY () { return (str_y_c+cur_y_c); }
int GetMaxX () { return (count_c[str_x_c+cur_x_c]); }
int GetMaxY () { return (count_y_c); }

int Save (char* file)
    {
    FILE* save_file = NULL;
    if ((save_file = fopen(file, "w+")) == NULL) return FALSE;
    for (int i = 0; i < count_y_c; i++)
            {
            char c = 0;
            for (int j = 0; j <= count_c[i]; j++) fputc (*(buf_c[i]+j),save_file);
            if (fputc ('\n', save_file) == EOF) return FALSE;
            }
    for (int j = 0; j <= count_c[i]; j++) fputc (*(buf_c[i]+j),save_file);
    fclose (save_file);
    return TRUE;
```

```
    }

~CEdit()
    {
    Scr.SetCur();
    }

void ReDraw (int x, int y)
    {
    Scr.BackColor (1);   //!
    Scr.TextColor (14);  //!
    sym_cor_c = 0;
    for (int j = str_y_c; j < str_y_c + size_y_c; j++)
        {
        int drawed   = 0,
            undrawed = 0,
            pnt      = 0,
            sym_cor  = 0;
        while (undrawed < str_x_c)
            {
            if (*(buf_c[j] + pnt) == KBD_TAB)
                {
                sym_cor  += TAB_LGH - (undrawed % TAB_LGH);
                sym_cor--;
                undrawed += TAB_LGH - (undrawed % TAB_LGH);
                }
            undrawed++;
            pnt++;
            }
        if (undrawed > str_x_c)
            {
            drawed = undrawed - str_x_c;
            for (int i = 0; i < drawed; i++) Scr.Put(x+x_start_c+i,y+y_start_c+j-str_y_c);
            }
        while (drawed < size_x_c)
            {
            if (*(buf_c[j] + pnt) == KBD_TAB)
                {
                sym_cor += TAB_LGH - ((str_x_c+drawed) % TAB_LGH);
                sym_cor--;
                for (int i = 0; (i < (TAB_LGH - ((str_x_c+drawed) % TAB_LGH))) &&
                                                ((drawed+i) < (size_x_c/*-1*/)); i++)
                    Scr.Put(x+x_start_c+i+drawed,y+y_start_c+j-str_y_c);
                drawed += i;
                pnt++;
                continue;
                }
            if (pnt > count_c[j]) Scr.Put (x+x_start_c+drawed,y+y_start_c+j-str_y_c);
            else
                {
                Scr.Put (x+x_start_c+drawed,y+y_start_c+j-str_y_c,*(buf_c[j]+pnt));
                pnt++;
                }
            drawed++;
            }
        while (pnt < count_c[j])
            {
            if (*(buf_c[j] + pnt) == KBD_TAB)
                {
                sym_cor  += TAB_LGH - ((str_x_c+drawed+undrawed) % TAB_LGH);
                sym_cor--;
                undrawed += TAB_LGH - ((str_x_c+drawed+undrawed) % TAB_LGH);
                }
            undrawed++;
            pnt++;
            }
        if (j == (cur_y_c+str_y_c)) sym_cor_c = sym_cor;
        if (j >= count_y_c) break;
        }
    j++;
    for (; j < str_y_c + size_y_c; j++)
        {
        for (int i = 0; i < size_x_c; i++) Scr.Put(x+x_start_c+i, y+y_start_c+j-str_y_c,' ');
        }
    }

void InsSim (char c)
    {
    if (!InsFlag || (cur_x_c + str_x_c - CurCor()) >= count_c[cur_y_c+str_y_c]+1)
        {
        if (count_c[cur_y_c+str_y_c] == MAX_EDIT_SYM) return;
        if ((cur_x_c + str_x_c - CurCor()) >= count_c[cur_y_c+str_y_c]+1)
            count_c[cur_y_c+str_y_c] = cur_x_c + str_x_c - 1 - CurCor();
        count_c[cur_y_c+str_y_c]++;
        for (int i = count_c[cur_y_c+str_y_c]-1; i >= (cur_x_c + str_x_c - CurCor()); i--)
            *(buf_c[cur_y_c+str_y_c] + i + 1) = *(buf_c[cur_y_c+str_y_c] + i);
        *(buf_c[cur_y_c+str_y_c] + count_c[cur_y_c+str_y_c] + 1) = ' ';
```

```
        }
    *(buf_c[cur_y_c+str_y_c] + cur_x_c + str_x_c - CurCor()) = c;
    }

int CurCor ()
    {
    int cor = 0,
        i   = 0,
        n   = 0,
        pnt = 0;
    while (i < (cur_x_c+str_x_c))
        {
        n++;
        i++;
        if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
            {
            cor += (TAB_LGH - (i % TAB_LGH));
            i   += (TAB_LGH - (i % TAB_LGH));
            }
//      i++;
        pnt++;
        }
    return cor;
    }

int CurCor1 ()
    {
    int cor = 0,
        i   = 0,
        n   = 0,
        pnt = 0;
    while (i < (cur_x_c+str_x_c))
        {
        n++;
        i++;
        if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
            {
            cor += (TAB_LGH - (i % TAB_LGH));
            i   += (TAB_LGH - (i % TAB_LGH));
            cor --;
            }
        pnt++;
        }
    return cor;
    }

void DelSim (int dir = 0)
    {
    if (dir && cur_x_c == 0 && str_x_c == 0)
        {
//BS
        if ((count_c[str_y_c+cur_y_c]+count_c[str_y_c+cur_y_c-1]) >= MAX_EDIT_SYM) return;
        if (cur_y_c+str_y_c == 0) return;
        for (int i = 0; i < (count_c[str_y_c+cur_y_c-1]); i++)
            {
            if (*(buf_c[str_y_c+cur_y_c-1]+i) == KBD_TAB)
                {
                cur_x_c += TAB_LGH - (cur_x_c % TAB_LGH);
                cur_x_c--;
                }
            cur_x_c++;
            }
        if(count_c[str_y_c+cur_y_c-1] >= 0) cur_x_c++;
        for (i = count_c[cur_y_c+str_y_c-1]+1; i < (count_c[cur_y_c+str_y_c-
    1])+count_c[cur_y_c+str_y_c]+2 && i < MAX_EDIT_SYM; i++)
            {
            *(buf_c[cur_y_c+str_y_c-1] + i) = *(buf_c[cur_y_c+str_y_c] + i -
    (count_c[cur_y_c+str_y_c-1]+1));
            }
        count_c[cur_y_c+str_y_c-1] += count_c[cur_y_c+str_y_c]+1;
        buf_c.SetOffset2(cur_y_c+str_y_c,cur_y_c+str_y_c+1,1);
        for (i = (cur_y_c + str_y_c); i <= (count_y_c-1); i++)
            {
            char t = *buf_c[i];
            t = *buf_c[i+1];
            buf_c.SetOffset2(i,i+1,0);
            count_c[i] = count_c[i+1];
            }
        buf_c.SetOffset3(count_y_c ,NULL, 0);
        cur_y_c--;
        count_y_c--;
        return;
        }

    if (!dir && (cur_x_c + str_x_c) >= count_c[cur_y_c+str_y_c]+1 && (cur_y_c+str_y_c !=
    count_y_c))
        {
```

```
//Del
        if ((count_c[str_y_c+cur_y_c]+count_c[str_y_c+cur_y_c+1]) >= MAX_EDIT_SYM) return;
        if (cur_y_c+str_y_c >= count_y_c) return;
        count_c[cur_y_c+str_y_c] = cur_x_c + str_x_c - 1;
        for (int i = count_c[cur_y_c+str_y_c]+1; i <
    (count_c[cur_y_c+str_y_c]+1)+count_c[cur_y_c+str_y_c+1]+1 && i < MAX_EDIT_SYM; i++)
            {
            *(buf_c[cur_y_c+str_y_c] + i) = *(buf_c[cur_y_c+str_y_c+1] + i -
    (count_c[cur_y_c+str_y_c]+1));
            }
        count_c[cur_y_c+str_y_c] += count_c[cur_y_c+str_y_c+1] + 1;
        buf_c.SetOffset2(cur_y_c+str_y_c+1,cur_y_c+str_y_c+2,1);
        for (i = (cur_y_c + str_y_c + 1); i <= (count_y_c-1); i++)
            {
            char t = *buf_c[i];
            t = *buf_c[i+1];
            buf_c.SetOffset2(i,i+1,0);
            count_c[i] = count_c[i+1];
            }
        buf_c.SetOffset3(count_y_c ,NULL, 0);
        count_y_c--;
        return;
        }

    if (dir || (cur_x_c+str_x_c-CurCor() > count_c[cur_y_c+str_y_c])) /*BkSp*/
        {
        cur_x_c--;
        if (*(buf_c[cur_y_c+str_y_c] + cur_x_c+str_x_c - CurCor()) == KBD_TAB)
                    {
                    int i = 0,
                      pnt = 0,
                        n = 0;
                    while (i < (cur_x_c+str_x_c))
                        {
                        i++;
                        if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
                            {
                            n  = (TAB_LGH - (i % TAB_LGH));
                            i += (TAB_LGH - (i % TAB_LGH));
                            }
                        pnt++;
                        }
                    cur_x_c -= n;
                    }
        if (cur_x_c < 0)
            {
            cur_x_c = 0;
            str_x_c--;
            if (str_x_c < 0)
                {
                str_x_c = 0;
                return;
                }
            }
        if ((cur_x_c + str_x_c - CurCor()) > count_c[cur_y_c+str_y_c]) return;
        }
    for (int i = (cur_x_c + str_x_c - CurCor()); i < count_c[cur_y_c+str_y_c]+1; i++)
    *(buf_c[cur_y_c+str_y_c] + i) = *(buf_c[cur_y_c+str_y_c] + i + 1);
    *(buf_c[cur_y_c+str_y_c] + count_c[cur_y_c+str_y_c]) = ' ';
    count_c[cur_y_c+str_y_c]--;
    if (count_c[cur_y_c+str_y_c] < -1) count_c[cur_y_c+str_y_c] = -1;
    }

void OnKeyDo (int c)
    {
    switch (c)
        {
         case KBD_ENTER:if ((cur_x_c + str_x_c - CurCor()) >= count_c[cur_y_c+str_y_c]+1)
    count_c[cur_y_c+str_y_c] = cur_x_c + str_x_c - 1 - CurCor();
                int temp_count = -1;
                char far *temp_p = NEW (char far [MAX_EDIT_SYM]);
                for (int i = 0; i < MAX_EDIT_SYM; i++) temp_p[i] = ' ';
                if (temp_p == NULL) return;
                for (i = (cur_x_c + str_x_c - CurCor()); i <= count_c[cur_y_c+str_y_c]; i++)
                     {
                     temp_count++;
                     *(temp_p + temp_count) = *(buf_c[cur_y_c+str_y_c] + i);
                     *(buf_c[cur_y_c+str_y_c] + i) = ' ';
                     }
                count_c[cur_y_c+str_y_c] -= temp_count+1;
                if (count_y_c == MAX_EDIT_STR) break;
                count_y_c++;
                char t = *buf_c[count_y_c-1];
                t = *buf_c[count_y_c];
                buf_c.SetOffset2(count_y_c,count_y_c-1,1);
                for (i = (count_y_c-1); i > (cur_y_c + str_y_c); i--)
                    {
```

```
            if (count_y_c == 0) break;
            t = *buf_c[i+1];
            t = *buf_c[i];
            buf_c.SetOffset2(i+1,i,0);
            count_c[i+1] = count_c[i];
            }
        cur_y_c++;
        t = *buf_c[cur_y_c+str_y_c];
        buf_c.SetOffset3(cur_y_c+str_y_c,temp_p,0);
        count_c[cur_y_c+str_y_c] = temp_count;
        cur_x_c = str_x_c = 0;
        break;
case KBD_UP   :cur_y_c--;break;
case KBD_DOWN :cur_y_c++;break;
case KBD_LEFT :cur_x_c--;
if (*(buf_c[cur_y_c+str_y_c] + cur_x_c+str_x_c - CurCor()) == KBD_TAB)
            {
            int i = 0,
              pnt = 0,
                n = 0;
            while (i < (cur_x_c+str_x_c))
                {
                i++;
                if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
                    {
                    n  = TAB_LGH - (i % TAB_LGH);
                    i += TAB_LGH - (i % TAB_LGH);
                    }
                pnt++;
                }
            cur_x_c -= n;
            }
        break;
case KBD_RIGHT:cur_x_c++;
        if (count_c[cur_y_c+str_y_c] == -1) break;
            i   = 0;
        int pnt = 0,
            n   = 0;
        while (i < (cur_x_c+str_x_c))
            {
            i++;
            if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
                {
                n  = (TAB_LGH - (i % TAB_LGH));
                i += (TAB_LGH - (i % TAB_LGH));
                }
            pnt++;
            }
        if (*(buf_c[cur_y_c+str_y_c] + cur_x_c+str_x_c - CurCor() + n - 1) == KBD_TAB)
            {
            cur_x_c += (TAB_LGH - ((cur_x_c+str_x_c) % TAB_LGH));
            }
        break;
case KBD_INS  :if (InsFlag == 1) InsFlag = 0; else InsFlag = 1;break;
case KBD_DEL  :DelSim();break;
case KBD_BKSP :DelSim(1);break;
case KBD_HOME :cur_x_c = 0;str_x_c = 0;break;
case KBD_END  :

   if ((count_c[cur_y_c+str_y_c]+sym_cor_c) < size_x_c)
        {
        str_x_c = 0;
        cur_x_c = count_c[cur_y_c+str_y_c]+1+sym_cor_c;
        break;
        }
   str_x_c = count_c[cur_y_c+str_y_c]+sym_cor_c - size_x_c + 1;
   cur_x_c = size_x_c;
   break;

case KBD_PGUP :cur_y_c -= (size_y_c - 1);
         if (cur_y_c < 0)
            {
            if (str_y_c != 0) str_y_c += cur_y_c;
            cur_y_c = 0;
            }
         break;
case KBD_PGDOWN:if ((cur_y_c+str_y_c) >= count_y_c) break;
         cur_y_c += (size_y_c - 1);
         if (cur_y_c >= size_y_c)
            {
            str_y_c += (cur_y_c - size_y_c);
            cur_y_c = size_y_c;
            if (str_y_c > count_y_c) str_y_c = count_y_c;
            }
          if ((cur_y_c+str_y_c) > count_y_c)
            {
            cur_y_c = count_y_c - str_y_c;
```

```
                    }
                  break;
          case 'A': case 'B': case 'C': case 'D': case 'E':
          case 'F': case 'G': case 'H': case 'I': case 'J':
          case 'K': case 'L': case 'M': case 'N': case 'O':
          case 'P': case 'Q': case 'R': case 'S': case 'T':
          case 'U': case 'V': case 'W': case 'X': case 'Y':
          case 'Z': case 'a': case 'b': case 'c': case 'd':
          case 'e': case 'f': case 'g': case 'h': case 'i':
          case 'j': case 'k': case 'l': case 'm': case 'n':
          case 'o': case 'p': case 'q': case 'r': case 's':
          case 't': case 'u': case 'v': case 'w': case 'x':
          case 'y': case 'z': case '0': case '1': case '2':
          case '3': case '4': case '5': case '6': case '7':
          case '8': case '9': case '`': case '~': case '!':
          case '@': case '#': case '$': case '%': case '^':
          case '&': case '*': case '(': case ')': case '-':
          case '_': case '=': case '+': case '[': case ']':
          case '{': case '}': case ';': case ':': case '"':
          case ',': case '<': case '.': case '>': case '/':
          case '?': case '\\': case ' ': case '\'': case KBD_TAB:
             if (count_c[cur_y_c+str_y_c] == (MAX_EDIT_SYM-1)) return;
             InsSim(c);
             cur_x_c++;
             if (count_c[cur_y_c+str_y_c] == -1) break;
             i   = 0;
             pnt = 0;
             n   = 0;
             while (i < (cur_x_c+str_x_c))
                 {
                 i++;
                 if (*(buf_c[cur_y_c+str_y_c] + pnt) == KBD_TAB)
                     {
                     n  = TAB_LGH - (i % TAB_LGH);
                     i += TAB_LGH - (i % TAB_LGH);
                     }
                 pnt++;
                 }
             if (*(buf_c[cur_y_c+str_y_c] + cur_x_c+str_x_c - CurCor() + n - 1) == KBD_TAB)
                 {
                 cur_x_c += (TAB_LGH - ((cur_x_c+str_x_c) % TAB_LGH));
                 }
             break;
          default: break;
          }
//y
     if ((cur_y_c+str_y_c) >= MAX_EDIT_STR) cur_y_c--;
     if ((cur_y_c+str_y_c) > count_y_c)
         {
         if (cur_y_c == size_y_c) str_y_c--;
         else cur_y_c--;
         }
     if (cur_y_c > size_y_c-1) {str_y_c++;cur_y_c = size_y_c-1;}
     if (cur_y_c < 0) {str_y_c--;cur_y_c = 0;}
     if (str_y_c < 0) str_y_c = 0;
     if (str_y_c > count_y_c) str_y_c = count_y_c;

//x
     if ((cur_x_c+str_x_c-CurCor()) > MAX_EDIT_SYM) cur_x_c--;
     while (cur_x_c < 0)
         {
         str_x_c--;
         cur_x_c++;
         }
     if (str_x_c < 0) str_x_c = 0;
     if ((cur_x_c+str_x_c-CurCor()) >= count_c[cur_y_c+str_y_c]+1) *(buf_c[cur_y_c+str_y_c] +
     cur_x_c + str_x_c - CurCor()) = ' ';
     if (cur_x_c > (size_x_c-1)) {str_x_c += cur_x_c - (size_x_c-1);cur_x_c = size_x_c-1;}
     }
};
//=============================================================================
//=============================================================================
class CEditLine : public CChildWnd
{

char* buf_c;
int count_c,   size_c,
     x_start_c, y_start_c,
     cur_pos_c, str_pos_c;

public:

CEditLine (int x_start = 0, int y_start = 0, int size = 0)
     {
     buf_c     = NEW (char [MAX_EDIT_SYM]);
     cur_pos_c = 0;
     str_pos_c = 0;
```

```
    count_c   = -1;
    x_start_c = x_start+1;
    y_start_c = y_start;
    size_c    = size;
    }

void Clear ()
    {
    cur_pos_c = 0;
    str_pos_c = 0;
    count_c   = -1;
    }
void ReDraw (int x, int y)
    {
    Scr.BackColor (1);   //!
    Scr.TextColor (15);  //!
    Scr.SetCur (x + x_start_c + cur_pos_c, y+y_start_c);
    if (str_pos_c > 0) Scr.Put(x+x_start_c-1, y+y_start_c, '□'); else Scr.Put(x+x_start_c-1,
    y+y_start_c, ' ');
    if (str_pos_c+size_c < count_c + 1) Scr.Put(x+x_start_c+size_c, y+y_start_c, '□'); else
    Scr.Put(x+x_start_c+size_c, y+y_start_c, ' ');
    if (str_pos_c != 0)
        {
        for (int i = str_pos_c; i < str_pos_c + size_c; i++)
            {
            Scr.Put (x + x_start_c + i - str_pos_c, y+y_start_c,*(buf_c + i));
            }
        }
    else
        {
        for (int i = 0; i < size_c; i++)
            {
            if (i <= count_c) Scr.Put (x+x_start_c+i,y+y_start_c,*(buf_c + i));
            if (i > count_c) Scr.Put (x+x_start_c+i,y+y_start_c);
            }
        }

    }

void ReDrawEdit (int x, int y)
    {
    Scr.BackColor (1);   //!
    Scr.TextColor (15);  //!
    if (str_pos_c > 0) Scr.Put(x+x_start_c-1, y+y_start_c, '□'); else Scr.Put(x+x_start_c-1,
    y+y_start_c, ' ');
    if (str_pos_c+size_c < count_c + 1) Scr.Put(x+x_start_c+size_c, y+y_start_c, '□'); else
    Scr.Put(x+x_start_c+size_c, y+y_start_c, ' ');
    if (str_pos_c != 0)
        {
        for (int i = str_pos_c; i < str_pos_c + size_c; i++)
            {
            Scr.Put (x + x_start_c + i - str_pos_c, y+y_start_c,*(buf_c + i));
            }
        }
    else
        {
        for (int i = 0; i < size_c; i++)
            {
            if (i <= count_c) Scr.Put (x+x_start_c+i,y+y_start_c,*(buf_c + i));
            if (i > count_c) Scr.Put (x+x_start_c+i,y+y_start_c);
            }
        }
    Scr.SetCur (x + x_start_c + cur_pos_c, y+y_start_c,InsFlag == 1? 2 : 1);
    }

void ReDrawAct (int x, int y)
    {
    ReDrawEdit (x,y);
    }

void OnKeyDo (int c)
    {
    switch (c)
        {
        case KBD_LEFT  : cur_pos_c--;break;
        case KBD_RIGHT : cur_pos_c++;break;
        case KBD_INS   : if (InsFlag == 1) InsFlag = 0;
                 else InsFlag = 1;break;
        case KBD_DEL   : DelSim();break;
        case KBD_BKSP  : DelSim(1);break;
        case KBD_HOME  : cur_pos_c = 0;str_pos_c = 0;break;
        case KBD_END   : str_pos_c = count_c + 1 - size_c;cur_pos_c = size_c;break;
        case 'A': case 'B': case 'C': case 'D': case 'E':
        case 'F': case 'G': case 'H': case 'I': case 'J':
        case 'K': case 'L': case 'M': case 'N': case 'O':
        case 'P': case 'Q': case 'R': case 'S': case 'T':
        case 'U': case 'V': case 'W': case 'X': case 'Y':
```

```
        case 'Z': case 'a': case 'b': case 'c': case 'd':
        case 'e': case 'f': case 'g': case 'h': case 'i':
        case 'j': case 'k': case 'l': case 'm': case 'n':
        case 'o': case 'p': case 'q': case 'r': case 's':
        case 't': case 'u': case 'v': case 'w': case 'x':
        case 'y': case 'z': case 'O': case '1': case '2':
        case '3': case '4': case '5': case '6': case '7':
        case '8': case '9': case '`': case '~': case '!':
        case '@': case '#': case '$': case '%': case '^':
        case '&': case '*': case '(': case ')': case '-':
        case '_': case '=': case '+': case '[': case ']':
        case '{': case '}': case ';': case ':': case '"':
        case ',': case '<': case '.': case '>': case '/':
        case '?': case '\\': InsSim(c); cur_pos_c++; break;
        default: break;
        }
    if (cur_pos_c > size_c-1) {str_pos_c++;cur_pos_c = size_c;}
    if (cur_pos_c+str_pos_c > count_c+1)
        {
        if (cur_pos_c == size_c) str_pos_c--;
        else cur_pos_c--;
        }
    if (cur_pos_c < 0) {{str_pos_c--;cur_pos_c = 0;}}
    if (str_pos_c < 0) str_pos_c = 0;
    if (str_pos_c > count_c + 1) str_pos_c = count_c + 1;
    }

void InsSim (char c)
    {
    if (!InsFlag)
        {
        count_c++;
        if (count_c > MAX_EDIT_SYM)
            {
            count_c--;
            return;
            }
        for (int i = count_c - 1; i >= (cur_pos_c + str_pos_c); i--) *(buf_c + i + 1) = *(buf_c
    + i);
        }
    *(buf_c + cur_pos_c + str_pos_c) = c;
    }

void DelSim (int dir = 0)
    {
    if (dir) /*BkSp*/
        {
        cur_pos_c--;
        if (cur_pos_c < 0)
            {
            cur_pos_c = 0;
            str_pos_c--;
            if (str_pos_c < 0)
                {
                str_pos_c = 0;
                return;
                }
            }
        }
    for (int i = (cur_pos_c + str_pos_c); i < count_c + 1; i++) *(buf_c + i) = *(buf_c + i +
    1);
    *(buf_c + count_c) = ' ';
    count_c--;
    if (count_c < -1) count_c = -1;
    }

char* GetEdit()
    {
    *(buf_c + count_c + 1) = '\0';
    return buf_c;
    }

~CEditLine()
    {
    DEL (buf_c);
    }
};
//============================================================================
//============================================================================
class CMenu : public CWindow
{

char     title_c   [SCR_ROW];
CButton* element_c [SCR_STRING];
void (*functions_c [SCR_STRING]) (void);
int count_c, title_lgh_c, title_x_c, next_c, prev_c;
```

```
public:

CMenu (int x = 0, char* title = NULL, int title_lgh = 0)
    {
    count_c     = -1;
    title_x_c   = x;
    y_end_c     = 2;
    x_end_c     = 0;
    x_c         = x;
    y_c         = 1;
    title_lgh_c = title_lgh;
    for (int i = 0; i <= title_lgh; i++) title_c [i] = *(title + i);
    }

void AddElement (char* where = NULL, int count = 0, void (*function)() = NULL)
    {
    count_c++;
    functions_c[count_c] = function;
    element_c [count_c] = NEW (CButton (1, y_end_c - 1 == str_c ? str_c + 1 : y_end_c - 1,
    count, where, 0));
    if ( count > x_end_c-1 ) x_end_c = x_c + count+1;
    if ((x_c + count + 2) >= (SCR_ROW - 1)) {x_end_c = SCR_ROW - 1;x_c = x_end_c - count - 1;}
    if (y_end_c - 1 == str_c) y_end_c += 1;
    y_end_c++;
    }

void ReDraw ()
    {
    Scr.SetCur();
    WndFrameDraw();
    DrawText();
    element_c[0]->ReDrawAct(x_c,y_c);
    for (int i = 1; i <= count_c; i++) element_c[i]->ReDraw(x_c,y_c);
    }

void WndFrameDraw()
    {
    Draw.Frame (x_c, y_c, x_end_c, y_end_c, FALSE, TRUE, 0 , 7, 0);
    Scr.BackColor (7);
    for (int i = x_c+1; i < x_end_c; i++)
        {
        for (int j = y_c+1; j < y_end_c; j++)
            {
            Scr.Put (i,j);
            }
        }
    }

void OnKeyDo (int c)
    {
    next_c = prev_c = 0;
    TDrawAct();
    ReDraw();
    while (1)
        {
        c = Kbd.WaitKey();
        switch (c)
            {
            case KBD_RIGHT: next_c = 1;TDraw();return;
            case KBD_LEFT : prev_c = 1;TDraw();return;
            case KBD_UP   : CButton* temp = element_c [count_c];
                    void (*temp_f) (void) = functions_c[count_c];
                    for (int i = count_c; i != 0; i--)
                        {
                        element_c [i] = element_c [i - 1];
                        functions_c[i] = functions_c[i - 1];
                        }
                    functions_c[0] = temp_f;
                    element_c [0] = temp;
                    break;
            case KBD_DOWN : temp = element_c [0];
                    temp_f = functions_c[0];
                    for (i = 0; i <= count_c-1; i++)
                        {
                        element_c [i] = element_c [i + 1];
                        functions_c[i] = functions_c[i + 1];
                        }
                    functions_c[count_c] = temp_f;
                    element_c [count_c] = temp;
                    break;
            case KBD_ENTER: TDraw();(*functions_c[0])();return;
            case KBD_ESC  : TDraw();return;
            }
        for (int i = 0; i <= count_c; i++)
        if (element_c [i]->HotKey(c))
            {
            (*functions_c[i])();
```

```
            TDraw();
            return;
            }
        element_c[0]->ReDrawAct(x_c,y_c);
        for (i = 1; i <= count_c; i++) element_c[i]->ReDraw(x_c,y_c);
        }
    }

void TDraw ()
    {
    Draw.Button (title_x_c, 0, &title_c[0], title_lgh_c, 0, 4);  //!!!
    }

void TDrawAct ()
    {
    Scr.BackColor (2);
    Draw.Button (title_x_c, 0, &title_c[0], title_lgh_c, 0, 4);
    Scr.BackColor (7);
    }

int NextTest ()
    {
    if (next_c == 1) return 1;
    if (prev_c == 1) return 2;
    return 0;
    }

~CMenu()
    {
    for (int i = 0; i <= count_c; i++) {DEL (element_c [i]);}
    }
};
//=============================================================================
//=============================================================================
class CManager
{
private:
CMenu*   menu_c [MAX_WND];
CWindow* wnd_c  [MAX_WND];
int count_c, menu_count_c;
short draw_c;

public:
short page_c;

CManager () { count_c = menu_count_c = -1;wnd_c [0] = NULL; page_c = 0; }
~CManager ()
    {
    for (int i = 0; i <= count_c; i++)
        {
        DEL (wnd_c[i]);
        }
    }

void AddMenu (CMenu* menu = NULL) {menu_count_c++; menu_c [menu_count_c] = menu; }

int Add (CWindow* wnd)
    {
    if ( count_c < MAX_WND )
        {
        count_c++;
        wnd_c [count_c] = wnd;
        WndReDraw();
        return TRUE;
        }
    else
        {
        return FALSE;
        DEL (wnd);
        }
    }

void Del (CWindow* wnd)
    {
    for (int i = 0; i <= count_c; i++)
        {
        if (wnd_c [i] == wnd)
            {
            DEL (wnd_c[i]);
            for (int j = i; j < count_c; j++)
                {
                int k = j + 1;
                wnd_c [j] = wnd_c [k];
                }
            }
        }
    count_c--;
```

```
    WndReDraw();
    }

void ReDraw()
    {
    WndReDraw();
    Scr.BackColor(7);    //!
    for (int i = 0; i <= SCR_ROW - 1; i++) Scr.Put (i,0);
    for (i = menu_count_c; i >= 0; i--) menu_c [i]->TDraw();
    }

void WndReDraw()
    {
    draw_c = 1;

    if (page_c == 0)
        {
        Scr.SetBaseAdress((char far*)0xb9000000L);
        Scr.BackColor(7);    //!
        for (int i = 0; i <= SCR_ROW - 1; i++) Scr.Put (i,0);
        for (i = menu_count_c; i >= 0; i--) menu_c [i]->TDraw();
        Draw.DeskDraw();
        for (i = 0; i <= count_c; i++) wnd_c [i]->ReDraw();
        if (draw_c == 1)
            {
            SetPage1();
            page_c = 1;
            draw_c = 0;
            return;
            }
        return;
        }

    if (page_c == 1)
        {
        Scr.SetBaseAdress((char far*)0xb8000000L);
        Scr.BackColor(7);    //!
        for (int i = 0; i <= SCR_ROW - 1; i++) Scr.Put (i,0);
        for (i = menu_count_c; i >= 0; i--) menu_c [i]->TDraw();
        Draw.DeskDraw();
        for (i = 0; i <= count_c; i++) wnd_c [i]->ReDraw();
        if (draw_c == 1)
            {
            SetPage0();
            page_c = 0;
            draw_c = 0;
            return;
            }
        return;
        }
    }
void EndDraw ()
    {
    if (page_c == 1 && draw_c == 1)
        {
        SetPage0();
        page_c = 0;
        draw_c = 0;
        return;
        }
    if (page_c == 0 && draw_c == 1)
        {
        SetPage1();
        page_c = 1;
        draw_c = 0;
        return;
        }
    }

void Run ()
    {
    for (int i = menu_count_c; i >= 0; i--) menu_c [i]->TDraw();
    WndReDraw();
    int c = 0;
    while (c != KBD_ALTX)
        {
        c = Kbd.WaitKey();
        switch (c)
            {
            case KBD_ALTF9:   verify_func();break;
            case KBD_CTRLF9:  write_func();break;
            case KBD_ALTF8:   erase_func();break;
            case KBD_F7:      step_into_func();break;
            case KBD_F8:      step_over_func();break;
            case KBD_F4:      goto_cur_func();break;
            case KBD_F3:      open_func();  break;
            case KBD_F9:      asm_func();   break;
```

```
                case KBD_F2:      save_func();  break;
                case KBD_ALTF10:  disasm_func();break;
                case KBD_F1: case KBD_ALTF1: case KBD_CTRLF1: help_func();break;
                case KBD_F10: case 333:
file_men:           menu_c[0]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[0]->NextTest() == 1) {goto comp_men;}
                    if (menu_c[0]->NextTest() == 2) {goto help_men;}
                    break;
                case 346:
comp_men:           menu_c[1]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[1]->NextTest() == 1) {goto debug_men;}
                    if (menu_c[1]->NextTest() == 2) {goto file_men;}
                    break;
                case 332:
debug_men:          menu_c[2]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[2]->NextTest() == 1) {goto prog_men;}
                    if (menu_c[2]->NextTest() == 2) {goto comp_men;}
                    break;
                case 325:
prog_men:           menu_c[3]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[3]->NextTest() == 1) {goto win_men;}
                    if (menu_c[3]->NextTest() == 2) {goto debug_men;}
                    break;
                case 317:
win_men:            menu_c[4]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[4]->NextTest() == 1) {goto help_men;}
                    if (menu_c[4]->NextTest() == 2) {goto prog_men;}
                    break;
                case 335:
help_men:           menu_c[5]->OnKeyDo(0);
                    WndReDraw();
                    if (menu_c[5]->NextTest() == 1) {goto file_men;}
                    if (menu_c[5]->NextTest() == 2) {goto win_men;}
                    break;
                case KBD_F5    : wnd_c[count_c]->ManMes(FULL_SCR);WndReDraw();break;
                case KBD_ALTF3 : if (count_c >= 0) Del (wnd_c [count_c]);WndReDraw();break;
                case KBD_F6    : CWindow* temp = wnd_c [0];
                          for (int i = 0; i <= count_c-1; i++)
                              {
                              wnd_c [i] = wnd_c [i + 1];
                              }
                          wnd_c [count_c] = temp;
                          WndReDraw();
                          break;
                case KBD_CTRLF5:
                         while (c != KBD_ESC)
                              {
                              if (count_c >= 0)
                                  Scr.Put(wnd_c[count_c]->x_c,wnd_c[count_c]->y_c,'M');
                              c = Kbd.WaitKey();
                              switch(c)
                                  {
                                  case KBD_UP   : if (count_c >= 0)
                                                       wnd_c[count_c]->ManMes(MOVE_UP);
                                                  WndReDraw();
                                                  break;
                                  case KBD_DOWN : if (count_c >= 0)
                                                       wnd_c[count_c]->ManMes(MOVE_DOWN);
                                                  WndReDraw();
                                                  break;
                                  case KBD_LEFT : if (count_c >= 0)
                                                       wnd_c[count_c]->ManMes(MOVE_LEFT);
                                                  WndReDraw();
                                                  break;
                                  case KBD_RIGHT: if (count_c >= 0)
                                                       wnd_c[count_c]->ManMes(MOVE_RIGHT);
                                                  WndReDraw();
                                                  break;
                                  }
                              }
                          WndReDraw();
                          break;

                case KBD_CTRLF6: while (c != KBD_ESC)
                                     {
                                     if (count_c >= 0)
                                         Scr.Put(wnd_c[count_c]->x_c,wnd_c[count_c]->y_c,'S');
                                     c = Kbd.WaitKey();
                                     switch(c)
                                         {
                                         case KBD_UP   : if (count_c >= 0)
                                                              wnd_c[count_c]->ManMes(BIGGER_UP);
```

```
                                                    WndReDraw();
                                                    break;
                                        case KBD_DOWN : if (count_c >= 0)
                                                        wnd_c[count_c]->ManMes(BIGGER_DOWN);
                                                    WndReDraw();
                                                    break;
                                        case KBD_LEFT : if (count_c >= 0)
                                                        wnd_c[count_c]->ManMes(BIGGER_LEFT);
                                                    WndReDraw();
                                                    break;
                                        case KBD_RIGHT: if (count_c >= 0)
                                                        wnd_c[count_c]->ManMes(BIGGER_RIGHT);
                                                    WndReDraw();
                                                    break;
                                        }
                                    }
                                WndReDraw();
                                break;

                default:      if (count_c > -1) wnd_c [count_c]->OnKeyDo (c);break;
                }
            }
        }

    void Move ()
        {
        WndReDraw();
        int c = 0;
        while (c != KBD_ESC)
            {
            if (count_c >= 0) Scr.Put(wnd_c[count_c]->x_c,wnd_c[count_c]->y_c,'M');
            c = Kbd.WaitKey();
            switch(c)
                {
                case KBD_UP   : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(MOVE_UP);WndReDraw();break;
                case KBD_DOWN : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(MOVE_DOWN);WndReDraw();break;
                case KBD_LEFT : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(MOVE_LEFT);WndReDraw();break;
                case KBD_RIGHT: if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(MOVE_RIGHT);WndReDraw();break;
                }
            }
        }

    void Size ()
        {
        WndReDraw();
        int c = 0;
        while (c != KBD_ESC)
            {
            if (count_c >= 0) Scr.Put(wnd_c[count_c]->x_c,wnd_c[count_c]->y_c,'S');
            c = Kbd.WaitKey();
            switch(c)
                {
                case KBD_UP   : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(BIGGER_UP);WndReDraw();break;
                case KBD_DOWN : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(BIGGER_DOWN);WndReDraw();break;
                case KBD_LEFT : if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(BIGGER_LEFT);WndReDraw();break;
                case KBD_RIGHT: if (count_c >= 0)
                                    wnd_c[count_c]->ManMes(BIGGER_RIGHT);WndReDraw();break;
                }
            }
        }

    void Zoom ()
        {
        if(count_c >= 0)
            {
            wnd_c[count_c]->ManMes(FULL_SCR);
            WndReDraw();
            }
        }

    void Next ()
        {
        if(count_c >= 0)
            {
            CWindow* temp = wnd_c [0];
            for (int i = 0; i <= count_c-1; i++)
                wnd_c [i] = wnd_c [i + 1];
            wnd_c [count_c] = temp;
            WndReDraw();
            }
```

```
    }

void Close()
    {
        if(count_c >= 0)
        {
        Del (wnd_c [count_c]);
        WndReDraw();
        }
    }

CWindow* GetAct()  {if (count_c == -1) return NULL; return wnd_c[count_c];}
CWindow* GetAct1() {if (count_c <= 0)   return wnd_c[count_c]; return wnd_c[count_c-1];}
};
//============================================================================
//============================================================================
class CProg
{

CManager* manager_c;

public:

CProg ()
    {
    manager_c = NEW (CManager);
    }

int  GetPage() { return manager_c->page_c; }
void EndDraw() { manager_c->EndDraw(); }
void Size()     { manager_c->Size();  }
void Move()    { manager_c->Move(); }
void Zoom()    { manager_c->Zoom(); }
void Next()    { manager_c->Next(); }
void Close()   { manager_c->Close(); }
CWindow* GetAct() { return manager_c->GetAct(); }
CWindow* GetAct1() { return manager_c->GetAct1(); }
void Del (CWindow* wnd) { manager_c->Del (wnd); }
void AddWnd (CWindow* wnd) { manager_c->Add (wnd); }
void AddMenu (CMenu* menu) { manager_c->AddMenu (menu); }
void ReDraw () { manager_c->ReDraw(); }

void Run()
    {
    Draw.DeskDraw();
    Scr.BackColor(7);    //!
    for (int i = 0; i <= SCR_ROW - 1; i++) Scr.Put (i,0);
    manager_c->Run();
    }

~CProg()
    {
    SetPage0();
    Scr.SetBaseAdress((char far*)0xb8000000l);
    DEL (manager_c);
    Scr.TextColor (7);   //!
    Scr.BackColor (0);   //!
    Scr.Clr();
    Scr.SetCur(0,0,1);
    }
};
//============================================================================
```

File Dis\Dis.asm

```
.model small, c
.code
;-----------------------------------------------------------------------------
;Pic disassemble eniege (14-bit) v 1.05b                              (c) Pit
;input   - [si] : command in hex
;output  - [di] : command in ascii
;destroy - ax bx cx si di
;-----------------------------------------------------------------------------
public DIS

DIS proc cod:word
        push si di

        push es
        mov ax,ds
        mov es,ax
        xor al,al
        mov cx,100d
```

```
        lea di,buf
        rep stosb
        mov ax,cod
        lea di,buf
        lea bx,opcode
        mov cx,39d      ;êîë-âî êîìàíä
        push ax

opcode_chse: cmp ds:[bx],ax  ;öèêë âûáîðêè
        je opcode_find  ;êîìàíä
        inc bx
        inc bx

        cmp cl,31d      ;ðåãèñòðîâûå êîìàíäû ;!!!
        jne reg_comm_start
        xor al,al

reg_comm_start: cmp cl,08h       ;áèòîâûå êîìàíäû
        jne n_bit
        and ah,11111100b

n_bit:      cmp cl,03h       ;jump & call
        jne n_call_goto
        and ah,11111000b

n_call_goto: loop opcode_chse
        pop ax
        jmp end_ret

opcode_find: pop ax
        push ax
        sub bx,offset opcode
        shr bx,1
        push bx
        shl bx,1     ;
        mov cx,bx        ;bx=
        shl bx,1     ;bx*6
        add bx,cx        ;
        mov si,bx
        add si,offset opc_ascii ;[bx]=command
        mov cl,06h
        rep movsb                ;Âûâîäèì
        mov ds:byte ptr [di],' ';êîìàíäó
        inc di                   ;â áóôåð
        pop bx
        push bx
        cmp bx,36d
        jae label_ch     ;Âûáîðêà ìåòêè
        cmp bx,32d
        jae  bit_ch              ;Âûáîðêà áèòà
        cmp bx,25d
        jae  con_ch              ;Âûáîðêà çíà÷åíèÿ
        cmp bx,20d
        jne no_specal_1
        and al,01111111b
no_specal_1: cmp bx,09d
        jne no_specal_2
;       and al,01111111b
no_specal_2: cmp bx,9d        ;!!!
        jae  reg_ch              ;Âûáîðêà ðåãèñòðà
        pop bx
        pop ax
        jmp end_ret

reg_ch:  push ax
        and al,01111111b
        mov bl,06h
        mul bl
        xchg si,ax
        add si,offset reg_name
        mov cl,06h
        rep movsb
        pop ax
        test al,80h
        jz n_d
        mov cl,06h
        dec di
d_@:    cmp byte ptr [di],20h
        jne d_write
        dec di
        loop d_@
d_write: inc di
        mov word ptr [di],'d,'
        inc di
        inc di
n_d:            pop bx
        pop ax
```

```
        jmp end_ret

label_ch:    push ax
        and ah,00000111b
        xchg al,ah
        mov cl,04h
        shl al,cl
        call hex2ascii
        dec di
        pop ax
        call hex2ascii
        mov es:byte ptr [di],'h'
        inc di
        pop bx
        pop ax
        jmp end_ret

con_ch: mov byte ptr [di],'0'
        inc di
        call hex2ascii
        mov byte ptr [di],'h'
        inc di
        pop bx
        pop ax
        jmp end_ret

bit_ch:      pop bx
        push ax
        and al,01111111b
        push bx
        mov bl,06h
        mul bl
        xchg si,ax
        add si,offset reg_name
        mov cl,06h
        rep movsb
        pop bx ax

        mov cl,06h
        dec di
b_@:    cmp byte ptr [di],20h
        jne b_write
        dec di
        loop b_@
b_write: inc di

        mov al,','
        stosb
        pop ax
        push ax
        and ax,0000001110000000b
        shl ax,1
        add ah,30h
        mov byte ptr [di],ah
        pop ax
        jmp end_ret

end_ret: pop es di si
        mov dx,ds
        lea ax,buf
        ret
        endp

hex2ascii:   lea bx,asc
        xor ah,ah
        mov cl,04h
        shl ax,cl
        shr al,cl
        xlat
        xchg al,ah
        xlat
        stosw
        ret

.data
asc   db '0123456789ABCDEF'

reg_name  db 'indr  ','rtcc  ','pcl   ','status','frs   ','porta ','portb '
      db 'noreg ','eedata','eeadr ','pclath','intcon','ram01 ','ram02 ','ram03 '
      db 'ram04 ','ram05 ','ram06 ','ram07 ','ram08 ','ram09 ','ram10 '
      db 'ram11 ','ram12 ','ram13 ','ram14 ','ram15 ','ram16 ','ram17 '
      db 'ram18 ','ram19 ','ram20 ','ram21 ','ram22 ','ram23 ','ram24 '
      db 'ram25 ','ram26 ','ram27 ','ram28 ','ram29 ','ram30 ','ram31 ','ram32 '
      db 'ram33 ','ram34 ','ram35 ','ram36 ','ram37 '
      db 'ram38 ','ram39 ','ram40 ','ram41 ','ram42 '
      db 'ram43 ','ram44 ','ram45 ','ram46 ','ram47 '
      db 'ram48 ','ram49 ','ram50 ','ram51 ','ram52 '
```

```
      db 'ram53 ','ram54 ','ram55 ','ram56 ','ram57 '
      db 'ram58 ','ram59 ','ram60 ','ram61 ','ram62 '
      db 'ram63 ','ram64 ','ram65 ','ram66 ','ram67 '

opcode    dw 0063h,0064h,0008h,0009h,0000h,0062h,0103h
      dw 0065h,0066h
      dw 0000h,0b00h,0e00h,0c00h,0400h
      dw 0200h,0300h,0900h,0d00h,0800h,0f00h
      dw 0100h,0600h,0500h,0700h,0a00h
      dw 3c00h,3000h,3E00h,3900h,3a00h,3400h,3800h
      dw 1000h,1400h,1800h,1c00h
      dw 2000h,2800h

opc_ascii db 'sleep ','clrwdt','return','retfie','nop   ','option','clrw  '
      db 'trisa ','trisb '
      db 'movwf ','decfsz','swapf ','rrf   ','iorwf '
      db 'subwf ','decf  ','comf  ','rlf   ','movf  ','incfsz'
      db 'clrf  ','xorwf ','andwf ','addwf ','incf  '
      db 'sublw ','movlw ','addlw ','andlw ','xorlw ','retlw ','iorlw '
      db 'bcf   ','bsf   ','btfsc ','btfss '
      db 'call  ','goto  '

buf       db 100d dup (00h)

end
```

File Dis\Disasem.cpp

```
//---------------------------------------------------------------------------
//16f84 disasem v0.99alpha                                          (c)Pit
//---------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern "C" char* DIS (int code);

const int MAX_DATA = 0x2200;
const int MAX_OUT  = 20000;
const int MAX_NOP  = 30;
FILE *In, *Out;
int Check = 0;

void Error (char* str)
    {
    fprintf(Out,"Error: %s.\n",str);
    fclose (In);
    fclose (Out);
    exit(0);
    }

int GetDigit ()
    {
    int digit = 0;
    if ((digit = getc(In)) == EOF) Error("EOF");
    digit -= digit>'9' ? 'A'-10 : '0';
    if (digit < 0 || digit > 0x0F) Error("Hex digit expected");
    return digit;
    }

unsigned int GetByte ()
    {
    int byte = 0;
    byte = GetDigit();
    byte = (byte << 4) + GetDigit();
    Check += byte;
    return byte;
    }

unsigned int GetWord ()
    {
    unsigned int word = 0;
    word = GetByte();
    word = (word << 8) + GetByte();
    return word;
    }

unsigned int GetWord1 ()
    {
    unsigned int word = 0;
    word = GetByte();
    word += GetByte() << 8;
    return word;
```

```
    }

int GetHex(unsigned int buf[],int bufsize)
    {
    int type = 0, count = 0, counter = 0;
    unsigned int adress = 0;

    while (type != 1)
        {
        if (getc(In) != ':') Error("Expected ':'");
        Check = 0;
        count = GetByte();
        adress = GetWord();
        type =  GetByte();
        for (int i = 0; i < (count/2); i++)
            {
            if (adress >= bufsize) Error("Address to big");
            buf[adress] = GetWord1();
            adress  += 2;
            counter += 2;
            }
        if (count%2 == 1)
            {
            counter += 2;
            buf[adress] = GetByte();
            }
        GetByte();
        if (Check&0x00ff)
            {
            Error("Checksum error");
            }
        getc(In);
        }
    return counter;
    }

void main (int argc, char* argv[])
    {
    printf("PIC16x84x Disassembler v 0.99alpha\n");
    if (argv[1] == NULL)
        {
        printf("\nUsing disasm.exe [filename]\n\n");
        return;
        }
    char* filename = new char [12*10];
    char* outfilename = new char [12*10];
    for (int i = 0; i < 12*10; i++)
        {
        *(outfilename + i) = *(filename + i) = *(argv[1] + i);
        if (*(filename + i) == '.' || *(filename + i) == 0) break;
        }
    if (*(outfilename + i) == 0) *(outfilename + i++) = '.';
    else i++;
    *(outfilename + i++) = 'a';
    *(outfilename + i++) = 's';
    *(outfilename + i++) = 'm';
    *(outfilename + i++) = 0;
    if ((Out = fopen(outfilename,"w+")) == NULL)
        {
        printf ("Can not create file.\n");
        exit(0);
        }
    if ((In = fopen(argv[1],"r")) == NULL)
        {
        Error ("Can not open file");
        }
    int unsigned* ina = new unsigned int [MAX_DATA];
    for (i = 0; i < MAX_DATA; i++) *(ina + i) = 0;
    int counter = GetHex (ina,MAX_DATA);
    char* outa = new char [MAX_OUT];
    char* pointer = outa;
    int nop_count = 0, code_count = 0;
    for (int count = 0; count < 0x3ff*2; count += 2)
        {
        char* command = DIS(ina [count]);
        if (*command == '\0') Error ("Error while disassembling");
        if (strcmp("nop    ",command) == 0)
            {
            nop_count++;
            continue;
            }
        if (nop_count > 0 && nop_count <= MAX_NOP)
            {
            for (int i = 0; i < nop_count; i++)
                {
                *(pointer + 0) = *(pointer + 1) = *(pointer + 2) =
                *(pointer + 3) = *(pointer + 4) = *(pointer + 5) = ' ';
```

```
                    *(pointer + 6) = 'n';
                    *(pointer + 7) = 'o';
                    *(pointer + 8) = 'p';
                    *(pointer + 9) = '\n';
                    pointer += 10;
                    }
                nop_count = 0;
                }
            if (nop_count > MAX_NOP)
                {
                *(pointer + 0) = *(pointer + 1) = *(pointer + 2) =
                *(pointer + 3) = *(pointer + 4) = *(pointer + 5) = ' ';
                *(pointer + 6) = 'o';
                *(pointer + 7) = 'r';
                *(pointer + 8) = 'g';
                *(pointer + 9) = ' ';
                pointer += 10;
                itoa (code_count, pointer, 16);
                while (*pointer != 0) pointer++;
                *pointer = 'h';
                *(pointer + 1) = '\n';
                pointer += 2;
                }
            *(pointer + 0) = *(pointer + 1) = *(pointer + 2) =
            *(pointer + 3) = *(pointer + 4) = *(pointer + 5) = ' ';
            pointer += 6;
            i = 0;
            while (*(command + i) != '\0') *(pointer++) = *(command++);
            *(pointer++) = '\n';
            code_count++;
            }
    //lables
        char* pnt = outa;
        int lab = 0, lab_count = 0;
        int lables [0x3ff];
        for (i = 0; i < 0x3ff; i++) lables[i] = 0;
        while (pnt < pointer)
            {
            pnt = strstr(pnt,"call");
            if (!pnt) break;
            pnt += 7;
            sscanf(pnt, "%3x", &lab);
            for (i = 0; i < lab_count; i++) if (lab == lables[i]) break;
            if (lab == lables[i]) continue;
            lables[lab_count] = lab;
            lab_count++;
            }
        pnt = outa;
        while (pnt < pointer)
            {
            pnt = strstr(pnt,"goto");
            if (!pnt) break;
            pnt += 7;
            sscanf(pnt, "%3x", &lab);
            for (i = 0; i < lab_count; i++) if (lab == lables[i]) break;
            if (lab == lables[i]) continue;
            lables[lab_count] = lab;
            lab_count++;
            }
        pnt = outa;
        int line_num = 0;
        char num [4];
        while (pnt < pointer)
            {
            for (i = 0; i < lab_count; i++) if (line_num == lables[i])
                {
                itoa (lables[i], &num[0], 16);
                for (i = 0; i < 4; i++) if (num[i] == 0) break;
                i--;

                for (int j = 0; j <= i; j++) *(pnt + 3 - j) = num[i-j];
                for (;j < 3; j++) *(pnt + 3 - j) = '0';
                *(pnt + 4) = ':';
                for (char* s = pointer; s >= pnt; s--) *(s+1) = *s;
                pointer++;
                pnt[0] = '\n';
                pnt[1] = '_';
                pnt += 4;
                break;
                }
            pnt = strstr(pnt,"\n");
            if (!pnt) break;
            pnt++;
            line_num++;
            }
    //eol
        *pointer = 0;
```

```
    strcpy(pointer, "\n\n      org 2000h\n\n");
    char *str_t = new char[6];
    char *str   = new char[6];
    for (i = 0; i < 7*2; i += 2)
        {
        strcat(pointer, "      dw ");
        while (*pointer != 0) pointer++;
        int temp = ina[0x2000 + i];
        temp <<= 8;
        ina[0x2000 + i] >>= 8;
        ina[0x2000 + i] +=  temp;

        itoa (ina[0x2000 + i], str, 16);

        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[3-j] = str[k-j];
        for (;j < 4; j++) str_t[3 - j] = '0';
        str_t[4] = 0;

        strcat(pointer, str_t);
        strcat(pointer, "h\n");
        }
    strcat(pointer, "\n\n      org 2100h\n\n");
    for (i = 0; i < 32*2; i += 2)
        {
        strcat(pointer, "      dw ");
        while (*pointer != 0) pointer++;
//      itoa (ina[0x2100 + i], pointer, 16);
        int temp = ina[0x2100 + i];
        temp <<= 8;
        ina[0x2100 + i] >>= 8;
        ina[0x2100 + i] +=  temp;

        itoa (ina[0x2100 + i], str, 16);

        for (int k = 0; k < 4; k++) if (str[k] == 0) break;
        k--;
        for (int j = 0; j <= k; j++) str_t[3-j] = str[k-j];
        for (;j < 4; j++) str_t[3 - j] = '0';
        str_t[4] = 0;

        strcat(pointer, str_t);

        strcat(pointer, "h\n");
        }

    strcat (pointer, "\n      end");
    while (*pointer != 0) pointer++;
    fprintf (Out, ";Generated by PIC Disassembler v0.99alpha\n\n      org 0000h\n\n");

    fprintf (Out,
    "\tindr\tequ\t00h\n"
    "\trtcc\tequ\t01h\n"
    "\tpcl\tequ\t02h\n"
    "\tstatus\tequ\t03h\n"
    "\tfrs\tequ\t04h\n"
    "\tporta\tequ\t05h\n"
    "\tportb\tequ\t06h\n"
    "\tnoreg\tequ\t07h\n"
    "\teedata\tequ\t08h\n"
    "\teeadr\tequ\t09h\n"
    "\tpclath\tequ\t0ah\n"
    "\tintcon\tequ\t0bh\n"

    "\tram01\tequ\t0ch\n"
    "\tram02\tequ\t0dh\n"
    "\tram03\tequ\t0eh\n"
    "\tram04\tequ\t0fh\n"
    "\tram05\tequ\t10h\n"
    "\tram06\tequ\t11h\n"
    "\tram07\tequ\t12h\n"
    "\tram08\tequ\t13h\n"
    "\tram09\tequ\t14h\n"
    "\tram10\tequ\t15h\n"
    "\tram11\tequ\t16h\n"
    "\tram12\tequ\t17h\n"
    "\tram13\tequ\t18h\n"
    "\tram14\tequ\t19h\n"
    "\tram15\tequ\t1ah\n"
    "\tram16\tequ\t1bh\n"
    "\tram17\tequ\t1ch\n"
    "\tram18\tequ\t1dh\n"
    "\tram19\tequ\t1eh\n"
    "\tram20\tequ\t1fh\n"
    "\tram21\tequ\t20h\n"
    "\tram22\tequ\t21h\n"
```

```
    "\tram23\tequ\t22h\n"
    "\tram24\tequ\t23h\n"
    "\tram25\tequ\t24h\n"
    "\tram26\tequ\t25h\n"
    "\tram27\tequ\t26h\n"
    "\tram28\tequ\t27h\n"
    "\tram29\tequ\t28h\n"
    "\tram30\tequ\t29h\n"
    "\tram31\tequ\t2ah\n"
    "\tram32\tequ\t2bh\n"
    "\tram33\tequ\t2ch\n"
    "\tram34\tequ\t2dh\n"
    "\tram35\tequ\t2eh\n"
    "\tram36\tequ\t2fh\n"
    "\tram37\tequ\t30h\n"
    "\tram38\tequ\t31h\n"
    "\tram39\tequ\t32h\n"
    "\tram40\tequ\t33h\n"
    "\tram41\tequ\t34h\n"
    "\tram42\tequ\t35h\n"
    "\tram43\tequ\t36h\n"
    "\tram44\tequ\t37h\n"
    "\tram45\tequ\t38h\n"
    "\tram46\tequ\t39h\n"
    "\tram47\tequ\t3ah\n"
    "\tram48\tequ\t3bh\n"
    "\tram49\tequ\t3ch\n"
    "\tram50\tequ\t3dh\n"
    "\tram51\tequ\t3eh\n"
    "\tram52\tequ\t3fh\n"
    "\tram53\tequ\t40h\n"
    "\tram54\tequ\t41h\n"
    "\tram55\tequ\t42h\n"
    "\tram56\tequ\t43h\n"
    "\tram57\tequ\t44h\n"
    "\tram58\tequ\t45h\n"
    "\tram59\tequ\t46h\n"
    "\tram60\tequ\t47h\n"
    "\tram61\tequ\t48h\n"
    "\tram62\tequ\t49h\n"
    "\tram63\tequ\t4ah\n"
    "\tram64\tequ\t4bh\n"
    "\tram65\tequ\t4ch\n"
    "\tram66\tequ\t4dh\n"
    "\tram67\tequ\t4eh\n\n\n"
    );

    fwrite (outa, 1, (pointer - outa), Out);
    fclose (In);
    fclose (Out);
    }
```

File Prog\CCom.h

```
#ifndef __CCOM_H__
#define __CCOM_H__

#include <conio.h>
#include <dos.h>
//===CONSTANTS========================================================================
#define _VER 0.1
#define COM1 0x3f8
#define COM2 0x2f8
#define COM3 0x3e8
#define COM4 0x2e8

//===DEFINITION=======================================================================
class CCom
{

//---INTERFACE---
public:
        CCom      (int port);

void PowerOn   ();
void PowerOff  ();
int     GetData   ();
void SendData  (int data);
void SendCmnd  (int command);
//---IN LIFE---
protected:
void    SetClk    (int vaule);
void    SetDta    (int vaule);
```

```
int     GetDta    ();
void SendBit   (int bit);
int     GetBit    ();
//---ATRIBUTES---
protected:
int port_c,
    w_reg_c;
};

//--REALISATION---
CCom::CCom(int port = 1)
    {
    switch(port)
        {
        case 1: port_c = COM1;break;
        case 2: port_c = COM2;break;
        case 3: port_c = COM3;break;
        case 4: port_c = COM4;break;
        default: port_c = COM1;break;
        }
//port init
    outp(port_c + 4, 0);
    outp(port_c + 3, 3);
    outp(port_c + 2, 6);
    outp(port_c + 1, 0);
    outp(port_c + 3, 0x80);
    outp(port_c,     1);
    outp(port_c + 1, 0);
    outp(port_c + 3, 3);
    w_reg_c = 0;
    }

void CCom::PowerOn()
    {
    SetClk(0);
    SetDta(0);
    delay(10);

    outp(port_c+3, 0x43);
    delay(200);

    outp(port_c+3, 0x03);
    asm {cli;}
    for(int i = 0; i < 3; i++)
        {
        outp(port_c,0x00);
        while(!(inp(port_c+5) | 0x20));
        }
    outp(port_c+3, 0x43);
    asm {sti;}
    }

void CCom::PowerOff()
    {
    outp(port_c+3, 0x03);
    }

void CCom::SetClk(int vaule)
    {
    outp(port_c+4, (w_reg_c & (!2))|(vaule << 1));
    w_reg_c = (w_reg_c & (!2))|(vaule << 1);
    }

void CCom::SetDta(int vaule)
    {
    outp(port_c+4, (w_reg_c & (!1))|vaule);
    w_reg_c = (w_reg_c & (!1))|vaule;
    }

int CCom::GetDta()
    {
    return (inp(port_c+6) >> 4 ) & 1;
    }

void CCom::SendCmnd(int command)
    {
    for (int i = 0; i < 6; i++)   SendBit((command >> i) & 1);
    }

void CCom::SendBit(int bit)
    {
    asm {cli;}
    SetClk(1);
    SetDta(bit);
    delay(1);
    SetClk(0);
    SetDta(0);
```

```
    asm {sti;}
    }

int CCom::GetBit()
    {
    int bit = 0;
    asm {cli;}
    SetClk(1);
    SetDta(1);
    delay(1);
    bit = GetDta();
    SetClk(0);
    asm {sti;}
    return bit;
    }

int CCom::GetData()
    {
    int data = 0;
    GetBit();
    for (int i = 0; i < 14; i++)
        {
        data |= GetBit() << i;
//      data <<= 1;
        }
    GetBit();
    return data;
    }

void CCom::SendData(int data)
    {
    SendBit(0);
    for (int i = 0; i < 14; i++) SendBit((data>>i) & 1);
    SendBit(0);
    }

//EOF
#endif __CCOM_H__
```

File Prog\CPic.h

```
#ifndef __CPIC_H__
#define __CPIC_H__

#include "ccom.h"
//===CONSTANTS========================================================================
#define _VER       0.1
#define PROG_LGH   0x3ff
#define DATA_START 0x2000
#define EE_LGH     7
#define FLUSES_LGH 64
//===DEFINITION=======================================================================
class CPic
{

//---INTERFACE---
public:
        CPic    (int com = 1);
        ~CPic   ();
int     Read    (int *buf);
int     Write   (int *buf);
int     Verify  (int *buf);
int     Erase   ();

//---ATRIBUTES---
protected:
CCom* com_c;

};

//--REALISATION---
CPic::CPic(int com)
    {
    com_c = new CCom(com);
    }

CPic::~CPic()
    {
    delete com_c;
    }

int CPic::Read(int *buf = 0)
    {
```

```
    com_c->PowerOn();
    for (int i = 0; i < PROG_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        buf[i] = com_c->GetData();
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();

    delay(10);
    com_c->PowerOn();
    com_c->SendCmnd(0x00);
    com_c->SendData(0x3fff);
    for (i = 0; i < FLUSES_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        buf[0x2000+i] = com_c->GetData();
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();

    delay(10);
    com_c->PowerOn();
    for (i = 0; i < EE_LGH; i++)
        {
        com_c->SendCmnd(0x05);
        buf[0x2100+i] = com_c->GetData();
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();
    }

int CPic::Write(int *buf = 0)
    {
    com_c->PowerOn();
    for (int i = 0; i < PROG_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        int temp = com_c->GetData();
        if (temp != buf[i])
            {
            com_c->SendCmnd(0x02);
            com_c->SendData(buf[i]);
            com_c->SendCmnd(0x08);
            delay(11);
            }
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();

    delay(10);
    com_c->PowerOn();
    for (i = 0; i < EE_LGH*2; i++)
        {
        com_c->SendCmnd(0x05);
        int byte = 0;
        if (i % 2 == 0) byte = buf[0x2100+i/2] & 0x00ff;
        if (i % 2 == 1) byte = (buf[0x2100+i/2] & 0xff00) >> 8;
        int temp = com_c->GetData();
        if (temp & 0x00ff != byte)
            {
            com_c->SendCmnd(0x03);
            com_c->SendData(byte);
            com_c->SendCmnd(0x08);
            delay(11);
            }
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();

    delay(10);
    com_c->PowerOn();
    com_c->SendCmnd(0x00);
    com_c->SendData(0x3fff);
    for (i = 0; i < FLUSES_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        int temp = com_c->GetData();
        if (temp != buf[0x2000+i])
            {
            com_c->SendCmnd(0x02);
            com_c->SendData(buf[0x2000+i]);
            com_c->SendCmnd(0x08);
            delay(11);
            }
        com_c->SendCmnd(0x06);
        }
```

```
    com_c->PowerOff();
    }

int CPic::Verify(int *buf = 0)
    {
    com_c->PowerOn();
    for (int i = 0; i < PROG_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        if (buf[i] != com_c->GetData()) return 0;
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();
    delay(11);
    com_c->PowerOn();
    for (i = 0; i < EE_LGH*2; i++)
        {
        com_c->SendCmnd(0x05);
        if (i%2 == 0) if (buf[(i/2)+0x2100]&0x00ff != com_c->GetData()&0xff) return 0;
        if (i%2 == 1) if ((buf[(i/2)+0x2100]&0xff00)>>8 != com_c->GetData()&0xff) return 0;
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();
    delay(11);
    com_c->PowerOn();
    com_c->SendCmnd(0x00);
    com_c->SendData(0x3fff);
    for (i = 0; i < FLUSES_LGH; i++)
        {
        com_c->SendCmnd(0x04);
        if (buf[i+0x2000] != com_c->GetData()) return 0;
        com_c->SendCmnd(0x06);
        }
    com_c->PowerOff();
    return 1;
    }

int CPic::Erase()
    {
    com_c->PowerOn();
    com_c->SendCmnd(0x00);
    com_c->SendData(0x3fff);
    for(int i = 0; i < FLUSES_LGH; i++)   com_c->SendCmnd(0x06);
    com_c->SendCmnd(0x02);
    com_c->SendData(0x3fff);
    com_c->SendCmnd(0x01);
    com_c->SendCmnd(0x07);
    com_c->SendCmnd(0x08);
    delay(11);
    com_c->SendCmnd(0x01);
    com_c->SendCmnd(0x07);
    com_c->PowerOff();
    }

//EOF
#endif __CPIC_H__
```

File Prog\Prog.cpp

```
#include "cpic.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

FILE* In = NULL;
int Check = 0;

void InFile (int* buf = NULL)
    {
    FILE* out = fopen ("out.hex","w+");

    int check = 0;
    for (int i = 0; i < 0x2100; i += 8)
        {
        check = 0;
        int num = 0x10;
        if ((i + 8) >= 0x2100) num = 2*(0x2100 - i);
        fprintf (out, ":%02X%04X00", num, (i*2));
        check += num + ((i*2) & 0xff) + (((i*2) >> 8) & 0xff);
        for (int j = 0; j < 8 && (i+j) < 0x2100; j++)
            {
            fprintf (out, "%02X%02X", buf[i + j]&0xff, (buf[i + j]>>8)&0xff);
            check += (*(buf + i + j)&0xff) + ((*(buf + i + j)>>8)&0xff);
```

```
            }
        fprintf (out, "%02X\n", (0x100 - (check & 0xff)) & 0xff);
        }

    int k = 0x2100;
    for (i = 0; i < 0x2100+64; i += 8)
        {
        check = 0;
        int num = 0x10;
        if ((i + 8) >= 0x2100+64) num = 2*(0x2100+64 - i);
        fprintf (out, ":%02X%04X00", num, (i*2));
        check += num + ((i*2) & 0xff) + (((i*2) >> 8) & 0xff);
        for (int j = 0; j < 8 && (i+j) < 0x2100+64; j++)
            {
            fprintf (out, "%02X%02X", buf[k]&0xff, buf[k+1]&0xff);
            check += (*(buf+ k)&0xff) + (*(buf + k + 1)&0xff);
            k += 2;
            }
        fprintf (out, "%02X\n", (0x100 - (check & 0xff)) & 0xff);
        }
    fprintf (out, ":00000001FF\n");
    fclose (out);
    }

void Error (char* str)
    {
    fclose(In);
    FILE* out = fopen("out.hex","w+");
    fprintf(out,"Error: %s.\n",str);
    fclose (out);
    exit(0);
    }

int GetDigit ()
    {
    int digit = 0;
    if ((digit = getc(In)) == EOF) Error("EOF");
    digit -= digit>'9' ? 'A'-10 : '0';
    if (digit < 0 || digit > 0x0F) Error("Hex digit expected");
    return digit;
    }

unsigned int GetByte ()
    {
    int byte = 0;
    byte = GetDigit();
    byte = (byte << 4) + GetDigit();
    Check += byte;
    return byte;
    }

unsigned int GetWord1 ()
    {
    unsigned int word = 0;
    word = GetByte();
    word += GetByte() << 8;
    return word;
    }

unsigned int GetWord ()
    {
    unsigned int word = 0;
    word = GetByte();
    word = (word << 8) + GetByte();
    return word;
    }

void OutFile(int* buf, int bufsize)
    {
    int type = 0, count = 0;
    unsigned int adress = 0;

    while (type != 1)
        {
        if (getc(In) != ':') Error("Expected ':'");
        Check = 0;
        count = GetByte();
        adress = GetWord();
        type =  GetByte();
        for (int i = 0; i < (count/2); i++)
            {
            if (adress >= bufsize) Error("Address to big");
            buf[adress] = GetWord1();
            adress  += 1;
            }
        if (count%2 == 1)
            {
```

```
            buf[adress] = GetByte();
            }
        GetByte();
        if (Check&0x00ff)
            {
            Error("Checksum error");
            }
        getc(In);
        }
    }

void main(int argc, char* argv[])
    {
    int buf [0x2200];
    for (int i = 0; i < 0x2200; i++) buf[i] = 0;

    int port = argv[2][0] - '0';
    CPic pic(port);
    if (strcmpi(argv[1],"r") == 0)
        {
        pic.Read(buf);
        InFile(buf);
        return;
        }
    if (strcmpi(argv[1],"w") == 0)
        {
        In = fopen("in.hex","r");
        OutFile(buf, 0x2200);
        pic.Write(buf);
        }
    if (strcmpi(argv[1],"e") == 0)
        {
        pic.Erase();
        return;
        }
    if (strcmpi(argv[1],"v") == 0)
        {
        In = fopen("in.hex","r");
        OutFile(buf, 0x2200);
        if (pic.Verify(buf) == 1) Error("Verify OK");
        else Error("Verify WRONG");
        return;
        }
    }
```

File Sim\Dis.asm

```
comm_num proc
        lea bx,opcode
        mov cx,39d
                push ax
opcode_chse: cmp [bx],ax
        je opcode_find
        inc bx
        inc bx
        cmp cl,31d
        jne reg_comm_start
        xor al,al
reg_comm_start:  cmp cl,07h
        jne n_bit
        and ah,11111100b
n_bit:      cmp cl,03h
        jne n_call_goto
        and ah,11111000b
n_call_goto: loop opcode_chse
        pop ax
        mov bx,0ffffh
        ret
opcode_find: pop ax
        sub bx,offset opcode
        shr bx,1
        ret

opcode  dw  0063h,0064h,0008h,0009h,0000h,0062h,0103h
        dw  0065h,0066h
        dw  0000h,0b00h,0e00h,0c00h,0400h
        dw  0200h,0300h,0900h,0d00h,0800h,0f00h
        dw  0100h,0600h,0500h,0700h,0a00h
    dw  3c00h,3000h,3E00h,3900h,3a00h,3400h,3800h
    dw  1000h,1400h,1800h,1c00h
    dw  2000h,2800h
endp
```

File Sim\Emul.asm

```
;.model tiny
;.386
;.code
;org 100h
;start:

emul_pic proc
;=============================================================================
;PiC 16f84-04 emulator eniege v0.9a                        (c)Pit
;
;!! Working with disasm eneige only !!
;in ax - opcode (out by disasm)
;   bx - num of command (out by disasm)
;out - see PiC data area
;=============================================================================

        xor di,di
;**********************************
;Îáðàáîòêà Ñáðîñà
;**********************************

        cmp mlsr_status,00h      ;Áûë ëè ñðîñ?
        jnz no_reset

pic_reset:   mov pic_ip,00h
        cmp eeprom_write_flag,0ffh
        jnz reset_eeprom_ok
              or eecon1_reg,08h
reset_eeprom_ok: or intcon_reg,80h        ;Ðàçðåøèòü ïåðåðûâàíèÿ
        mov sleep_flag,00h              ;Âûéòè èç sleep
        and status_reg,0e7h
        and status_reg_2,0e7h
        mov wdt_num,18d                 ;Ñáðîñ wdt
        mov mlsr_status,0ffh
;DOTO REG
        ret

;**********************************
;Îáðàáîòêà WDT
;**********************************

no_reset:    test config_word,04h
        jz no_wdt_reset

        test option_reg,08h
        jz wdt_no_div
        call div_test_proc
        cmp div_var,00h
        jz no_wdt_reset
        jmp no_wdt_reset

wdt_no_div:  dec wdt_num
        cmp wdt_num,00h
        jnz no_wdt_reset
        inc pic_ip
        jmp pic_reset

;**********************************
;Îáðàáîòêà RTCC
;**********************************

no_wdt_reset:    test option_reg,20h  ;Èíêðåìåíò ïî âíåøíåìó ñèãíàëó
        jz rtcc_in       ;èëè âíóòðåííåìó

        test portb_reg,10h
        jnz rtcc_up_front

        test option_reg,10h
        jz rtcc_div_test
        jmp no_rtcc

rtcc_up_front:   test portb_reg,10h
        jnz rtcc_div_test
        jmp no_rtcc

rtcc_div_test:   test option_reg,08h
        jnz rtcc_no_div
        call div_test_proc
        cmp div_var,00h
        jz no_rtcc
```

```
rtcc_no_div: inc rtcc_reg
         jmp no_rtcc

rtcc_in: test option_reg,08h
         jnz no_rtcc_div1
         call div_test_proc
         cmp div_var,00h
         jnz no_rtcc

no_rtcc_div1:    inc rtcc_reg
         jmp no_rtcc

;-----------------------------------
;Ïðîâåðêà äåëèòåëÿ
;-----------------------------------
div_test_proc    proc

         mov cl,option_reg
         and cl,0f7h
         cmp cl,old_div
         jnz div_new

         cmp div_var,00h
         jz div_new
         dec div_var
         jmp div_test_end

div_new: mov old_div,cl
         mov ch,01h
         test option_reg,08h
         jnz div_no_rtcc
         inc cl

div_no_rtcc: shl ch,cl
         mov div_var,ch
         jnc div_test_end
         mov div_var,0ffh

div_test_end:    ret
         endp

;**********************************
;Second Cyrcle
;**********************************
no_rtcc: cmp sec_cyr_flag,0ffh
         jnz no_sec_cyr
         mov sec_cyr_flag,00h
         ret

;**********************************
;Îáðàáîòêà EEPROM
;**********************************

no_sec_cyr: cmp eeprom_write_flag,0ffh
         jnz eeprom_no_write_cyrcle
         dec eeprom_counter
         cmp eeprom_counter,00h
         jz eeprom_end_write
         ret

eeprom_end_write: mov eeprom_write_flag,00h
         or eecon1_reg,10h
         and eecon1_reg,0fdh
         jmp no_eeprom

eeprom_no_write_cyrcle:  test eecon1_reg,01h
         jnz eeprom_read

         test eecon1_reg,02h
         jz no_eeprom
         test eecon1_reg,04h
         jz no_eeprom
         cmp byte ptr eeprom_write_flag,05h
         jnz no_eeprom
         cmp eecon2_reg,0aah
         jnz no_eeprom
         cmp eeadr_reg,40h
         ja no_eeprom

         mov cl,eedata_reg
         mov cx,offset eeprom_data_area
         add cl,eeadr_reg
         mov si,cx
         mov byte ptr [si],cl

         mov eeprom_write_flag,0ffh
         mov eeprom_counter,10d
```

```
        ret

eeprom_read: cmp eeadr_reg,40h
        ja no_eeprom

        mov cx,offset eeprom_data_area
        add cl,eeadr_reg
        mov si,cx
        mov cl,[si]
        mov eedata_reg,cl
        and eecon1_reg,0feh
        jmp no_eeprom

;**********************************
;Îáðàáîòêà ïðåðûâàíèé
;**********************************

no_eeprom:   test intcon_reg,80h
        jz no_interrupt

                test intcon_reg,08h
        jz no_rb_int
                mov cl,portb_reg
        cmp cl,old_portb
        jz no_rb_int
        mov pic_ip,03h
        or intcon_reg,01h
        mov sleep_flag,05h

no_rb_int:      test intcon_reg,10h
        jz no_int_int
                test option_reg,40h
        jz up_int
                test portb_reg,01h
        jz no_int_int
        jmp int_int_ok
up_int:         test portb_reg,01h
        jnz no_int_int

int_int_ok:  or intcon_reg,02h
        mov pic_ip,03h
                mov sleep_flag,05h

no_int_int:  test intcon_reg,20h
        jz no_interrupt
        cmp rtcc_reg,0ffh
        jne no_interrupt
        mov rtcc_reg,00h
        mov pic_ip,03h
        or intcon_reg,04h

        test eecon1_reg,10h
        jz no_eeint
        test intcon_reg,40h
        jnz eeint_ok
        and eecon1_reg,0efh
        jmp no_eeint
eeint_ok:       mov pic_ip,03h
        mov sleep_flag,05h

no_eeint:    mov cl,intcon_reg
        mov intcon_reg_2,cl
        ret

;**********************************
;Îáðàáîòêà sleep'a
;**********************************
no_interrupt:    cmp sleep_flag,0ffh
        jnz no_sleep
        cmp sleep_flag,05h
        jnz no_sleep
        mov sleep_flag,00h
        and status_reg,0e7h

        ret

;**********************************
;Îáðàáîòêà êîìàíä
;**********************************
no_sleep:    shl bx,1
        jmp command_table[bx]

command_table   dw sleep_c,clrwdt_c,return_c,retfie_c,nop_c,option_c
        dw clrw_c,trisa_c,trisb_c,movwf_c,decfsz_c,swapf_c
        dw rrf_c,iorwf_c,subwf_c,decf_c,comf_c,rlf_c,movf_c
        dw incfsz_c,clrf_c,xorwf_c,andwf_c,addwf_c,incf_c
        dw sublw_c,movlw_c,addlw_c,andlw_c,xorlw_c,retlw_c
```

```
        dw iorlw_c,bcf_c,bsf_c,btfsc_c,btfss_c,call_c,goto_c

sleep_c: mov wdt_num,18d
        test option_reg,8
        jz no_zero_wdt_div
                and option_reg,0f8h

no_zero_wdt_div: or status_reg,10h
        and status_reg,0f7h
        mov sleep_flag,0ffh
        jmp end_emul

clrwdt_c:    mov wdt_num,18d
                test option_reg,8
        or status_reg,18h

        jz end_emul
                and option_reg,0f8h
        jmp end_emul

retfie_c:    or intcon_reg,80h

return_c:    dec stack_status
        mov cx,stack_1
        mov pic_ip,cx

;       dec pic_ip
;       dec pic_ip
        mov di,offset stack_start
        mov si,di
        inc si
        inc si
        mov cx,07h
        rep movsw
        mov stack_8,0h
        mov sec_cyr_flag,0ffh
        jmp end_emul

nop_c:       jmp end_emul

option_c:       mov cl,w_reg
        mov option_reg,cl
                jmp end_emul

clrw_c:      mov w_reg,00h
        or status_reg,4h
        jmp end_emul

trisa_c:        mov cl,w_reg
        mov trisa_reg,cl
        jmp end_emul

trisb_c:        mov cl,w_reg
        mov trisb_reg,cl
        jmp end_emul

movwf_c:        call get_reg_adr
        mov cl,w_reg
        mov byte ptr [si],cl
        jmp end_emul

decfsz_c:    call get_reg_adr
        dec cl
        or cl,cl
        jnz no_decfsz_skip
command_skip:    inc pic_ip
        mov sec_cyr_flag,0ffh
no_decfsz_skip:  mov byte ptr [di],cl
        jmp end_emul

swapf_c: call get_reg_adr
        rol cl,4
        mov byte ptr [di],cl
                jmp end_emul

rrf_c:          call get_reg_adr
        test cl,1
        setz ch
        push cx
        mov ch,status_reg
        mov cl,8
        shl ch,cl
        mov cl,byte ptr [si]
        shr cl,1
        add cl,ch
        pop cx
        or status_reg,ch
```

```
        mov byte ptr [di],cl
        jmp end_emul

rlf_c:      call get_reg_adr
        test byte ptr [si],1
        setz ch
        push cx
        mov ch,status_reg
        and ch,1
        mov cl,byte ptr [si]
        shl cl,1
        add cl,ch
        pop cx
        or status_reg,ch
        mov byte ptr [di],cl
        jmp end_emul

iorwf_c: call get_reg_adr
        or cl,w_reg
               call z_flag_test
        mov byte ptr [di],cl
        jmp end_emul

subwf_c: call get_reg_adr
        mov ch,w_reg
        push cx
        and cx,0f0fh
        sub cl,ch
        jc subwf_no_dc_flag
        or status_reg,2h
subwf_no_dc_flag: and status_reg,0fdh
        pop cx
        sub cl,ch
        call flags_test
        mov byte ptr [di],cl
        jmp end_emul

decf_c:     call get_reg_adr
        dec cl
        call z_flag_test
        mov byte ptr [di],cl
        jmp end_emul

comf_c:     call get_reg_adr
        neg cl
        call z_flag_test
        mov byte ptr [di],cl
        jmp end_emul

movf_c:     call get_reg_adr
        mov byte ptr [di],cl
        or cl,cl
        call z_flag_test
        jmp end_emul

incfsz_c:       call get_reg_adr
        inc cl
        or cl,cl
        jnz no_incfsz_skip
command_skip_1:  inc pic_ip
        mov sec_cyr_flag,0ffh
no_incfsz_skip:  mov byte ptr [di],cl
        jmp end_emul


clrf_c:     call get_reg_adr
        mov byte ptr [si],0
               or status_reg,04h
        jmp end_emul

xorwf_c: call get_reg_adr
        mov ch,w_reg
        xor ch,cl
        call z_flag_test
        mov byte ptr [di],ch
        jmp end_emul

andwf_c: call get_reg_adr
        mov ch,w_reg
        and ch,cl
        call z_flag_test
        mov byte ptr [di],ch
        jmp end_emul

incf_c:         call get_reg_adr
        inc cl
        call z_flag_test
```

```
        mov byte ptr [di],cl
        jmp end_emul

addwf_c: call get_reg_adr
        mov ch,w_reg
        push cx
        and cx,0f0fh
        add cl,ch
        jc addwf_no_dc_flag
        or status_reg,2h
addwf_no_dc_flag: and status_reg,0fdh
        pop cx
        add cl,ch
        call flags_test
        mov byte ptr [di],cl
        jmp end_emul

sublw_c: mov cl,w_reg
        mov ch,al
        push cx
        and cx,0f0fh
        sub cl,ch
        jc sublw_no_dc_flag
        or status_reg,2h
sublw_no_dc_flag: and status_reg,0fdh
        pop cx
        sub cl,ch
        call flags_test
        mov w_reg,cl
        jmp end_emul

movlw_c:        mov w_reg,al
        jmp end_emul

addlw_c:        mov cl,w_reg
        mov ch,al
        push cx
        and cx,0f0fh
        add cl,ch
        jc addlw_no_dc_flag
        or status_reg,2h
addlw_no_dc_flag: and status_reg,0fdh
        pop cx
        add cl,ch
        call flags_test
        mov w_reg,cl
        jmp end_emul

andlw_c:        and w_reg,al
        call z_flag_test
        jmp end_emul

xorlw_c:        xor w_reg,al
        call z_flag_test
        jmp end_emul

retlw_c:        mov w_reg,al
        jmp return_c

iorlw_c:        or w_reg,al
        call z_flag_test
        jmp end_emul

bcf_c:      call get_bit
        call get_reg_adr
;       sub ch,0ffh
;       not ch
;       and byte ptr [si],ch
        jmp end_emul

bsf_c:          call get_bit
        call get_reg_adr
                or byte ptr [si],ch
        jmp end_emul

btfsc_c:        call get_reg_adr
        call get_bit
        test byte ptr [si],ch
        jnz no_bcommand_skip
bcom_skip:   inc pic_ip
        mov sec_cyr_flag,0ffh
no_bcommand_skip: jmp end_emul

btfss_c:        call get_bit
        call get_reg_adr
                test byte ptr [si],ch
        jnz bcom_skip
```

```
         jmp end_emul

call_c:         inc stack_status
         mov si,offset stack_8
         mov di,si
         dec si
         dec si
         mov cx,07h
         std
         rep movsw
         cld
         mov cx,pic_ip
         mov stack_1,cx
         call get_lable
         mov pic_ip,cx
         dec pic_ip
         jmp end_emul

goto_c:         call get_lable
         mov pic_ip,cx
         dec pic_ip

end_emul:    inc word ptr pic_ip
         push si ax
         lea si,pic_ip
         lodsw
         mov pclath_reg,ah
         mov pcl_reg,al
         pop ax si
         mov cl,portb_reg
                mov old_portb,cl
         push ax
         xor ax,ax
         mov indr_reg,al
         mov indr_reg_2,al
         pop ax
         cmp eecon2_reg,55h
         jnz  end_no_eewrt
         mov eeprom_write_flag,05h

end_no_eewrt:    cmp di,offset status_reg_2
         jnz st_reg2_change
         mov cl,status_reg_2
         mov status_reg,cl
                jmp next_reg_test
st_reg2_change:  mov cl,status_reg
         mov status_reg_2,cl

next_reg_test:  cmp di,offset intcon_reg_2
         jnz intc_reg2_change
         mov cl,intcon_reg_2
         mov intcon_reg,cl
                jmp next_reg_test1
intc_reg2_change:mov cl,intcon_reg
         mov intcon_reg_2,cl

next_reg_test1:  cmp di,offset pcl_reg_2
         jnz plc_reg2_change
         mov cl,pcl_reg_2
                mov pcl_reg,cl
         jmp next_reg_test2
plc_reg2_change: mov cl,pcl_reg
                mov pcl_reg_2,cl

next_reg_test2:  sub di,offset reg_pic_1
         cmp di,12d
         ja reg_pic_page2
;                mov cl,indr_reg_2
;        mov indr_reg,cl
         mov cl,fsr_reg_2
         mov fsr_reg,cl
         mov cl,pclath_reg_2
         mov pclath_reg,cl
                ret

reg_pic_page2:
;        mov cl,indr_reg
;        mov indr_reg_2,cl
         mov cl,fsr_reg
         mov fsr_reg_2,cl
         mov cl,pclath_reg
         mov pclath_reg_2,cl
         ret

;----------------------------------
;Ôëàãè
;----------------------------------
```

```
flags_test:  jc c_flag_ok
         and status_reg,0feh
         jmp z_flag_test
c_flag_ok:   or status_reg,1

z_flag_test: jz z_flags_ok
                 and status_reg,0fbh
         ret
z_flags_ok:  or status_reg,04h
         ret

;---------------------------------
;Ïîëó÷åíèå áèòà
;---------------------------------
get_bit proc
         mov cx,ax
         and cx,0000001110000000b
                 shl cx,1
         ret
endp

;---------------------------------
;Ïîëó÷åíèå ìåòêè
;---------------------------------
get_lable proc
         mov cx,ax
         and ch,00000111b
         ret
endp

;---------------------------------
;Ïîëó÷åíèå àäðåñà ðåãiñòðà
;---------------------------------
get_reg_adr proc

;input  ax - opcode
;output si - adress of register
;   di - output register adress

         mov cl,al
         and cl,7fh
         xor ch,ch
         cmp cl,0h
         jz kosv_adr
         test status_reg,20h

         jz reg_1_page
reg_2_page:  add cx,[offset reg_pic_2]
         jmp reg_2_page_done
reg_1_page:  add cx,[offset reg_pic_1]
reg_2_page_done: mov si,cx
         mov cl,byte ptr [si]
         jmp d_test

kosv_adr:    mov cl,fsr_reg
         xor ch,ch
                 test cl,80h
         jz reg_1_page_
         and cl,7fh
         jmp reg_2_page
reg_1_page_: and cl,7fh
         jmp reg_1_page

;        test status_reg,80h
;        jz reg_1_page
;        jmp reg_2_page

d_test:      test al,80h
         jz no_d_flag
         mov di,offset w_reg
         ret
no_d_flag:   mov di,si
         ret
endp

data_table:
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;PiC Data Area
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

old_portb db ?
pic_ip dw ?
config_word dw ?
mlsr_status db ?
w_reg db ?

stack_start equ $
```

```
stack_1 dw ?
stack_2 dw ?
stack_3 dw ?
stack_4 dw ?
stack_5 dw ?
stack_6 dw ?
stack_7 dw ?
stack_8 dw ?
stack_status db ?

reg_pic_1 equ $

indr_reg db ?
rtcc_reg db ?
pcl_reg  db ?
status_reg db ?
fsr_reg db ?
porta_reg db ?
portb_reg db ?
no_reg db ?
eedata_reg db ?
eeadr_reg db ?
pclath_reg db ?
intcon_reg db ?
ram00 db ?
ram01 db ?
ram02 db ?
ram03 db ?
ram04 db ?
ram05 db ?
ram06 db ?
ram07 db ?
ram08 db ?
ram09 db ?
ram10 db ?
ram11 db ?
ram12 db ?
ram13 db ?
ram14 db ?
ram15 db ?
ram16 db ?
ram17 db ?
ram18 db ?
ram19 db ?
ram20 db ?
ram21 db ?
ram22 db ?
ram23 db ?
ram24 db ?
ram25 db ?
ram26 db ?
ram27 db ?
ram28 db ?
ram29 db ?
ram30 db ?
ram31 db ?
ram32 db ?
ram33 db ?
ram34 db ?

reg_pic_2 equ $

indr_reg_2 db ?
option_reg db ?
pcl_reg_2 db ?
status_reg_2 db ?
fsr_reg_2 db ?
trisa_reg db ?
trisb_reg db ?
no_reg_2 db ?
eecon1_reg db ?
eecon2_reg db ?
pclath_reg_2 db ?
intcon_reg_2 db ?
ram35 db ?
ram36 db ?
ram37 db ?
ram38 db ?
ram39 db ?
ram40 db ?
ram41 db ?
ram42 db ?
ram43 db ?
ram44 db ?
ram45 db ?
ram46 db ?
ram47 db ?
```

```
ram48 db ?
ram49 db ?
ram50 db ?
ram51 db ?
ram52 db ?
ram53 db ?
ram54 db ?
ram55 db ?
ram56 db ?
ram57 db ?
ram58 db ?
ram59 db ?
ram60 db ?
ram61 db ?
ram62 db ?
ram63 db ?
ram64 db ?
ram65 db ?
ram66 db ?
ram67 db ?

wdt_num db ?
sec_cyr_flag db ?
div_var db ?
old_div db ?
eeprom_counter db ?
eeprom_write_flag db ?
eeprom_data_area db 40h dup (?)
sleep_flag db    ?

endp
```

File Sim\Sim.asm

```
.model tiny
.code
.386
org 100h

start:       mov ah,3dh
        xor al,al
        lea dx,in_file
        int 21h
        jnc file_ok
        mov mlsr_status,0ffh
        lea di,data_table
        lea si,init_data
        mov cx,(offset sleep_flag - offset data_table)
        rep movsb
        jmp emul_st
file_ok:        xchg ax,bx
        mov ah,3fh
        mov cx,(offset sleep_flag - offset data_table)
        lea dx,data_table
        int 21h
        mov ah,3eh
        int 21h
emul_st:
;       mov si,80h
;       mov ax,word ptr [si]
        mov ah,3dh
        xor al,al
        lea dx,com_file
        int 21h
        xchg bx,ax
        mov cx,02h
        mov ah,3fh
        lea dx,cmd
        int 21h
        mov ax,cmd
        mov comman,ax
        call comm_num
        cmp bx,0ffffh
        jz exit
        call emul_pic
exit:        lea dx,out_file
        mov ah,3ch
        xor cx,cx
        int 21h
        xchg ax,bx
        mov ah,40h
        mov cx,(offset sleep_flag - offset data_table)
        inc cx
```

```
        lea dx,data_table
        int 21h
        mov ah,3eh
        int 21h
        ret

comman dw  ?
cmd dw ?
com_file db 'command',0
in_file db 'in.sim',0
out_file db 'out.sim',0

include emul.asm
include dis.asm

init_data db (offset sleep_flag - offset data_table) dup (00h) ;!!
end start
```

File Sim\Sim_Dis\Dis.asm

```
.model small, c
.code
;-------------------------------------------------------------------------------
;Pic disassembly eniege (14-bit) v 1.05b                                  (c) Pit
;input - [si] : command in hex
;output - [di] : command in ascii
;destroy - ax bx cx si di
;-------------------------------------------------------------------------------
public DIS

DIS proc cod:word
        push si di
        push es
        mov ax,ds
        mov es,ax
        xor al,al
        mov cx,100d
        lea di,buf
        rep stosb
        mov ax,cod
        lea di,buf
        lea bx,opcode
        mov cx,39d
        push ax

opcode_chse: cmp ds:[bx],ax
        je opcode_find
        inc bx
        inc bx
        cmp cl,31d
        jne reg_comm_start
        xor al,al

reg_comm_start: cmp cl,08h
        jne n_bit
        and ah,11111100b

n_bit:      cmp cl,03h
        jne n_call_goto
        and ah,11111000b

n_call_goto: loop opcode_chse
        pop ax
        jmp end_ret

opcode_find: pop ax
        push ax
        sub bx,offset opcode
        shr bx,1
        push bx
        shl bx,1
        mov cx,bx
        shl bx,1
        add bx,cx
        mov si,bx
        add si,offset opc_ascii
        mov cl,06h
        rep movsb
        mov ds:byte ptr [di],' '
        inc di
        pop bx
        push bx
        cmp bx,36d
```

```
         jae label_ch
         cmp bx,32d
         jae  bit_ch
         cmp bx,25d
         jae  con_ch
         cmp bx,20d
         jne no_specal_1
         and al,01111111b
no_specal_1: cmp bx,09d
         jne no_specal_2
;        and al,01111111b
no_specal_2: cmp bx,9d
         jae reg_ch
         pop bx
         pop ax
         jmp end_ret

reg_ch:      push ax
         and al,01111111b
         call hex2ascii
         mov byte ptr [di],'h'
         inc di
         pop ax
         test al,80h
         jz n_d
         mov cl,06h
         dec di
d_@:     cmp byte ptr [di],20h
         jne d_write
         dec di
         loop d_@
d_write: inc di
         mov word ptr [di],'d,'
         inc di
         inc di
n_d:             pop bx
         pop ax
         jmp end_ret

label_ch:    push ax
         and ah,00000111b
         xchg al,ah
         mov cl,04h
         shl al,cl
         call hex2ascii
         dec di
         pop ax
         call hex2ascii
         mov es:byte ptr [di],'h'
         inc di
         pop bx
         pop ax
         jmp end_ret

con_ch:      call hex2ascii
         mov byte ptr [di],'h'
         inc di
         pop bx
         pop ax
         jmp end_ret

bit_ch:      pop bx
         push ax
         and al,01111111b
         push bx
         call hex2ascii
         mov byte ptr [di],'h'
         inc di
         pop bx ax
         mov al,','
         stosb
         pop ax
         push ax
         and ax,0000001110000000b
         shl ax,1
         add ah,30h
         mov byte ptr [di],ah
         pop ax
         jmp end_ret

end_ret: pop es di si
         mov dx,ds
         lea ax,buf
         ret
         endp

hex2ascii:   lea bx,asc
```

```
        xor ah,ah
        mov cl,04h
        shl ax,cl
        shr al,cl
        xlat
        xchg al,ah
        xlat
        stosw
        ret

.data
asc db  '0123456789ABCDEF'

opcode  dw  0063h,0064h,0008h,0009h,0000h,0062h,0103h
    dw  0065h,0066h
    dw  0000h,0b00h,0e00h,0c00h,0400h
    dw  0200h,0300h,0900h,0d00h,0800h,0f00h
    dw  0100h,0600h,0500h,0700h,0a00h
    dw  3c00h,3000h,3e00h,3900h,3a00h,3400h,3800h
    dw  1000h,1400h,1800h,1c00h
    dw  2000h,2800h

opc_ascii db 'sleep ','clrwdt','return','retfie','nop   ','option','clrw  '
      db 'trisa ','trisb '
      db 'movwf ','decfsz','swapf ','rrf   ','iorwf '
      db 'subwf ','decf  ','comf  ','rlf   ','movf  ','incfsz'
      db 'clrf  ','xorwf ','andwf ','addwf ','incf  '
      db 'sublw ','movlw ','addlw ','andlw ','xorlw ','retlw ','iorlw '
      db 'bcf   ','bsf   ','btfsc ','btfss '
      db 'call  ','goto  '

buf       db 100d dup (00h)
end
```

File Sim\Sim_Dis\Sim_Dis.cpp

```
//---------------------------------------------------------------------------
//16f84 disasem v0.99alpha                                          (c)Pit
//---------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include <fcntl.h>
#include <sys\stat.h>
extern "C" char* DIS (int code);

const int MAX_DATA = 0x2200;
const int MAX_OUT  = 20000;
FILE *In, *Out;
int Check = 0;

void Error (char* str)
    {
    fprintf(Out,"Error: %s.\n",str);
    fclose (In);
    fclose (Out);
    exit(0);
    }

int GetDigit ()
    {
    int digit = 0;
    if ((digit = getc(In)) == EOF) Error("EOF");
    digit -= digit>'9' ? 'A'-10 : '0';
    if (digit < 0 || digit > 0x0F) Error("Hex digit expected");
    return digit;
    }

unsigned int GetByte ()
    {
    int byte = 0;
    byte = GetDigit();
    byte = (byte << 4) + GetDigit();
    Check += byte;
    return byte;
    }

unsigned int GetWord ()
    {
    unsigned int word = 0;
    word = GetByte();
    word = (word << 8) + GetByte();
```

```
    return word;
    }

unsigned int GetWord1 ()
    {
    unsigned int word = 0;
    word = GetByte();
    word += GetByte() << 8;
    return word;
    }

int GetHex(unsigned int buf[],int bufsize)
    {
    int type = 0, count = 0, counter = 0;
    unsigned int adress = 0;

    while (type != 1)
        {
        if (getc(In) != ':') Error("Expected ':'");
        Check = 0;
        count = GetByte();
        adress = GetWord();
        type =  GetByte();
        for (int i = 0; i < (count/2); i++)
            {
            if (adress >= bufsize) Error("Address to big");
            buf[adress] = GetWord1();
            adress  += 2;
            counter += 2;
            }
        if (count%2 == 1)
            {
            counter += 2;
            buf[adress] = GetByte();
            }
        GetByte();
        if (Check&0x00ff)
            {
            Error("Checksum error");
            }
        getc(In);
        }
    return counter;
    }

void main (int argc, char* argv[])
    {
    if (argv[1] == NULL)
        {
        return;
        }
    char* filename = new char [12*10];
    char* outfilename = new char [12*10];
    for (int i = 0; i < 12*10; i++)
        {
        *(outfilename + i) = *(filename + i) = *(argv[1] + i);
        if (*(filename + i) == '.' || *(filename + i) == 0) break;
        }
    if (*(outfilename + i) == 0) *(outfilename + i++) = '.';
    else i++;
    *(outfilename + i++) = 'd';
    *(outfilename + i++) = 'i';
    *(outfilename + i++) = 's';
    *(outfilename + i++) = 0;
    if ((Out = fopen(outfilename,"w+")) == NULL)
        {
        exit(0);
        }

    if ((In = fopen(argv[1],"r")) == NULL)
        {
        Error ("Can not open file");
        }

    int unsigned *ina = new unsigned int [MAX_DATA];
    for (i = 0; i < MAX_DATA; i++) *(ina + i) = 0;
    int counter = GetHex (ina,MAX_DATA);

    outfilename = new char [12*10];
    for (i = 0; i < 12*10; i++)
        {
        *(outfilename + i) = *(filename + i) = *(argv[1] + i);
        if (*(filename + i) == '.' || *(filename + i) == 0) break;
        }
    if (*(outfilename + i) == 0) *(outfilename + i++) = '.';
    else i++;
    *(outfilename + i++) = 'c';
```

```
    *(outfilename + i++) = 'o';
    *(outfilename + i++) = 'd';
    *(outfilename + i++) = 0;
    int cod = 0;
    if ((cod = open(outfilename,O_CREAT|O_RDWR|O_BINARY, S_IREAD|S_IWRITE)) == -1)
        {
        exit(0);
        }
    lseek (cod, 0, SEEK_SET);
    for (i = 0; i < counter/2; i++)
    _write (cod, &ina[i*2], 2);
    _write (cod, 0, 0);

    char* outa = new char [MAX_OUT];
    char* pointer = outa;
    for (int count = 0; count < 0x3ff*2; count += 2)
        {
        char* command = DIS(ina [count]);
        if (*command == '\0') Error ("Error while disassembling");
        for (int i = 0; i <= 13; i++)
            {
            if (*command == 0)
                {
                *(pointer++) = ' ';
                continue;
                }
            *(pointer++) = *(command++);
            }
        *(pointer++) = '\n';
        }
    fwrite (outa, 1, (pointer - outa), Out);
//  int bla = 0;
//  fseek(Out, (pointer - outa), SEEK_SET);
//  fwrite (&bla, 2, 0, Out);
    fclose (In);
    fclose (Out);
    close (cod);
```