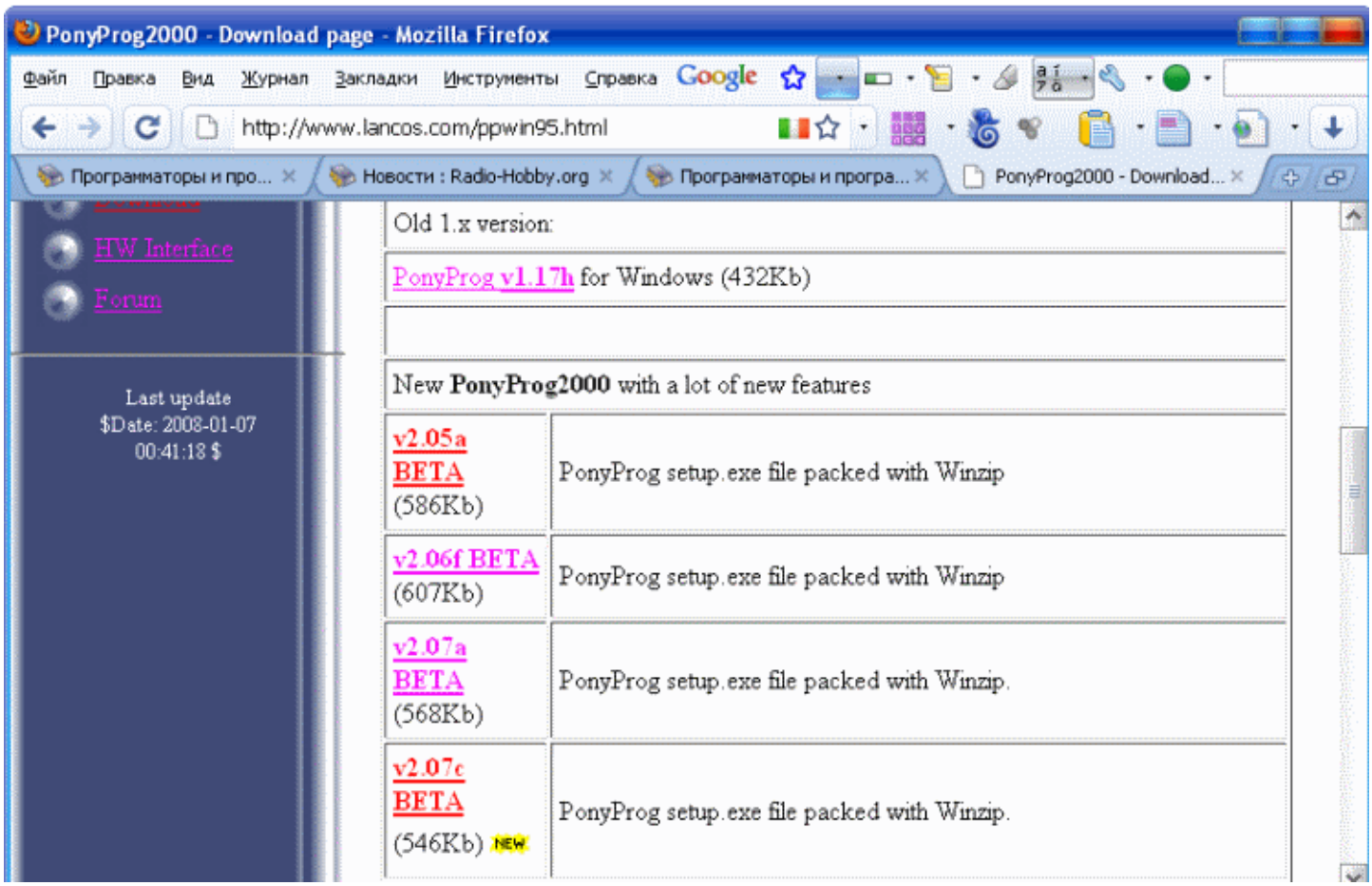
# Программирование микроконтроллеров (PonyProg)


Набрав в Интернет - браузере адрес http://www.lancos.com/ppwin95.html увидим на экране картину со списком версий программы PonyProg. В самой нижней его части – последняя англоязычная версия программы (2.07c).

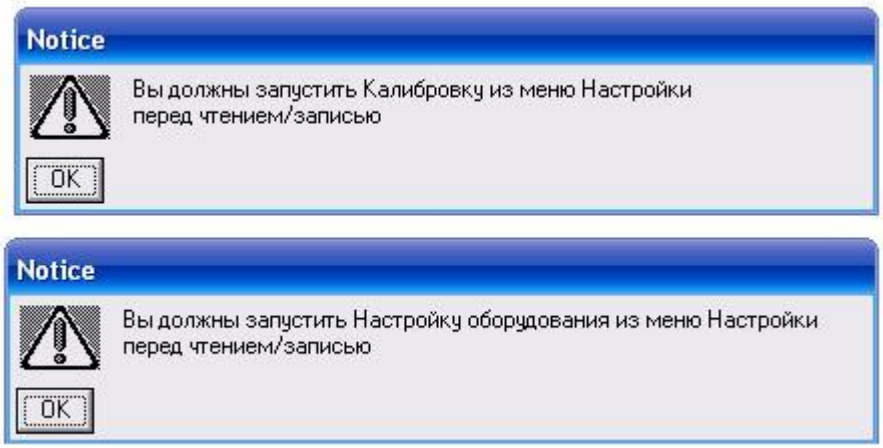



Что же "скачать"? Прежде всего, "щелкнув" по надписи "v2.07c BETA", оригинальный англоязычный вариант. (Так же скачать можно с нашего сервера). В полученном файле-архиве ponyprogV207c.zip находится программа-установщик setup.exe. После запуска она автоматически установит PonyProg на компьютере. От пользователя потребуется лишь отвечать согласием на все выводимые на экран запросы.
Те, кто предпочитает работать с русскоязычной программой, должны установить руссификотор PONYPROG2000ru.exe в папку C:\Program Files\PonyProg2000 (она была создана программой-установщиком на предыдущем этапе.

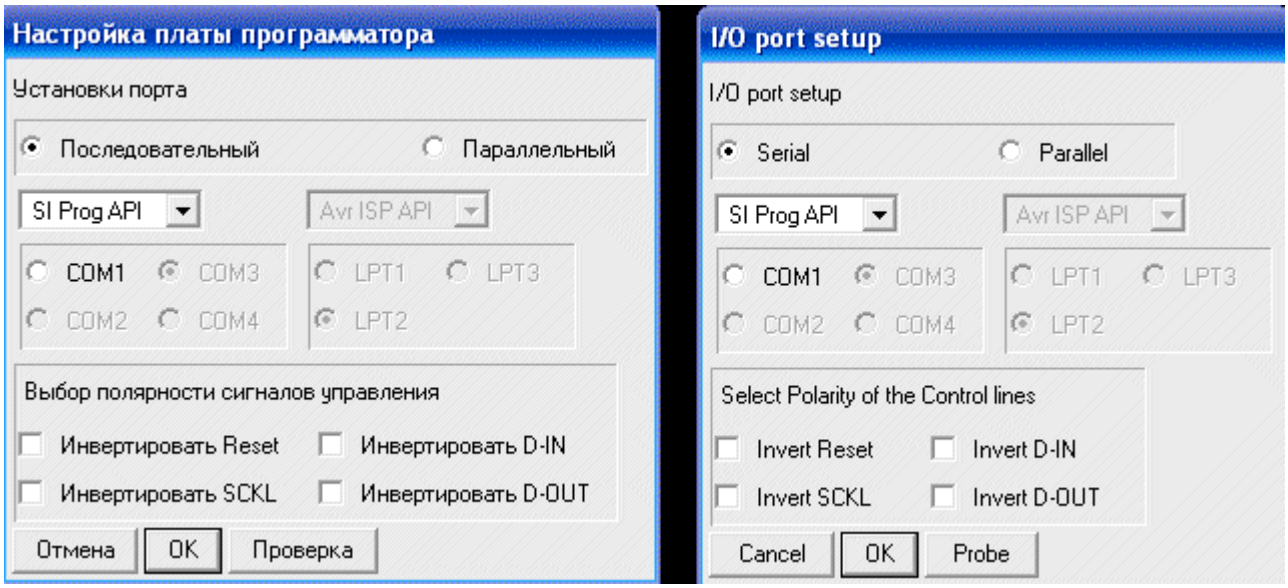
Это даст возможность по выбору запускать английский или русский варианты программы. Однако без установки первой вторая не работает. Учтите также, что русифицированный вариант относится к версии 2.05 и не содержит новшеств, появившихся в более поздних версиях, в том числе расширенного списка программируемых микросхем.
Все сказанное в этом разделе относится к вариантам программы PonyProg — приложения Windows.

# *Запуск и настройка PonyProg*

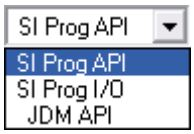
В результате установки PonyProg одноименный раздел появится в меню "Пуск/Программы" компьютера. В нем есть строка PonyProg2000 со значком в виде лошадиной головы. Можно запускать программу прямо отсюда или предварительно (для удобства) создать его ярлык на "Рабочей столе". Ярлыки английского и русского вариантов программы по умолчанию получат одно и то же имя PonyProg2000. Чтобы не путаться, лучше переименовать ярлык русского варианта, например, в PonyProgRus.
Первое, что будет выведено на экран после запуска PonyProg — главное окно с перечнем пунктов главного меню и кнопками управления в верхней части, а в нем — окно меньшего размера с краткой информацией о программе и ее авторе.



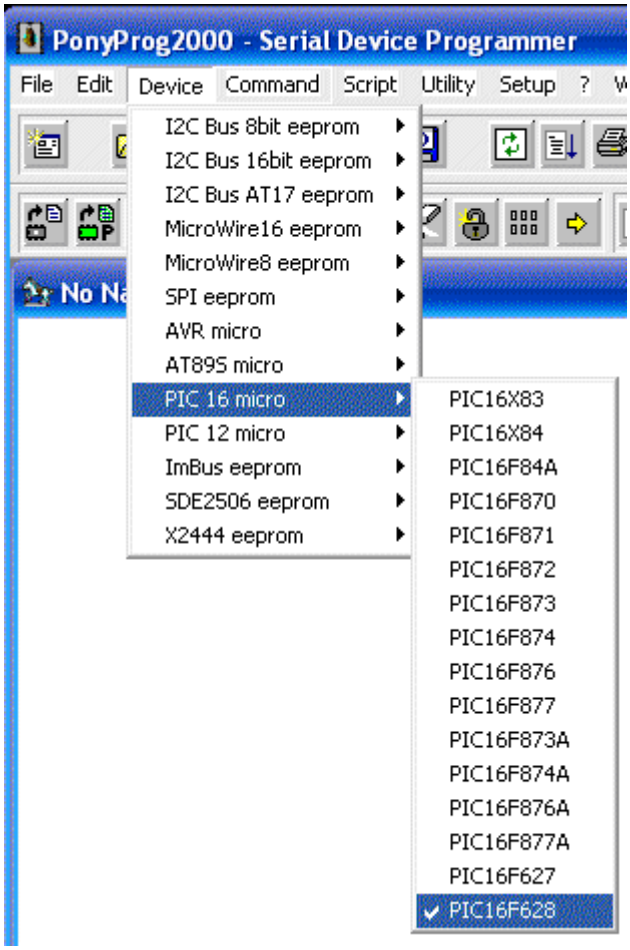

Закройте его, нажав кнопку "ОК". Тут же одно за другим появятся два предупреждения, которым мы последуем чуть позже, а пока просто закроем их, нажимая "ОК". Учтите, что в русском тексте предупреждений меню "Установки" ошибочно названо "Настройки". Аналогичная ошибка имеется и в английском варианте. Меню "Setup" названо "Options".




Если адаптер программирования до сих пор не подключен к компьютеру, сейчас самое время это сделать. Выберем в главном меню пункт "Установки", а в нем — "Настройка оборудования". На экране появится окно "Настройка платы программатора", укажем в нем тип порта (последовательный или параллельный), к которому подключен адаптер, и его имя (например, COM1). При нажатии кнопки со стрелкой вниз в соответствующем окошке "выпадет" список адаптеров, с которыми способна работать программа. Для последовательного порта список состоит всего из трех строк:




Как уже было сказано, SI Prog — оригинальный набор адаптеров, разработанных автором программы PonyProg специально для нее. Пометки API и I/O определяют способ общения программы с портами компьютера. В первом случае она использует стандартные функции Windows API (интерфейса прикладных программ Windows), во втором — обращается к портам "напрямую". Второй метод более эффективен, но, к сожалению, не все версии Windows его допускают. Поэтому во избежание неприятностей выбирайте API. На адаптеры, подключаемые к параллельному порту компьютера, программу настраивают аналогичным образом.
Сделав выбор, нажмите кнопку "Проверка". Компьютер сообщит, удалось ли ему обнаружить адаптер, подключенный к указанному порту (К сожалению, эта проверка работает только с оригинальными адаптерами набора Sl-Prog.). Если нет, первым делом проверьте, включен ли внешний источник питания адаптера (если он предусмотрен), убедитесь, что все разъемы сочленены надежно и правильно, а перемычки, специально предусмотренные в адаптерах для их распознавания компьютером (о них было рассказано в предыдущих разделах), находятся на своих местах. Причиной неработоспособности адаптера может быть и неправильный выбор метода общения с ним компьютера, о чем было сказано выше.
Панель "Выбор полярности сигналов управления" дает возможность настроить программу на работу с адаптерами, отсутствующими в списке, или с некоторыми "нестандартными" микросхемами. Ставя "галочки" в соответствующих клетках этой панели, задают программную инверсию всех или некоторых сигналов, учитывая таким образом особенности адаптера или микросхемы.
Закончив настройку, нажмите кнопку "ОК".
Далее выполним пункт "Калибровка" меню "Установки". Программа "измерит" скорость работы компьютера и вычислит значения переменных, определяющих в дальнейшем длительность импульсов и пауз между ними, формируемых в процессе программирования. Учтите, что как при калибровке, так и при собственно программировании все другие (кроме PonyProg) программы должны быть закрыты. Иначе неизбежны ошибки.
Описанные операции необходимы лишь при первом запуске PonyProg. Программа запомнит сделанные установки и при последующих запусках выполнит их автоматически. Повторить настройку придется лишь при смене адаптера или при подключении его к другому порту. Одно важное замечание. PonyProg не знает, рассчитан ли выбранный адаптер на работу с нужной микросхемой. Вся ответственность за правильный выбор лежит на пользователе.




Последний этап настройки — выбор типа программируемой микросхемы. Чтобы выполнить его, нужно выбрать в главном меню пункт "Устройство" и на экране появится список семейств микросхем, которые можно запрограммировать с помощью PonyProg. Выбрав одно из них, получим список входящих в него микросхем (рис. 4, показано окно версии 2.07c). Если микросхема была выбрана ранее, она помечена "галочкой".Чтобы сделать или изменить выбор, достаточно щелкнуть по названию нужной микросхемы. Списки исчезнут с экрана, а выбранное название появится в специальном окне в верхнем правом углу окна PonyProg. Еще в одном окне слева от упомянутого указано название семейства микросхем. Эти окна дают возможность выбирать семейство и микросхему в нем, не открывая меню "Устройство". Достаточно нажать в соответствующем окне кнопку со стрелкой вниз, чтобы "выпал" список, из которого можно сделать выбор.
Название выбранной микросхемы выведено и в нижней строке главного окна (строке состояния). Рядом указаны информационная емкость программируемой памяти этой микросхемы (суммарная емкость FLASH и EEPROM) в байтах и контрольная сумма (CRC) ее содержимого, точнее говоря, его копии в программном буфере PonyProg.

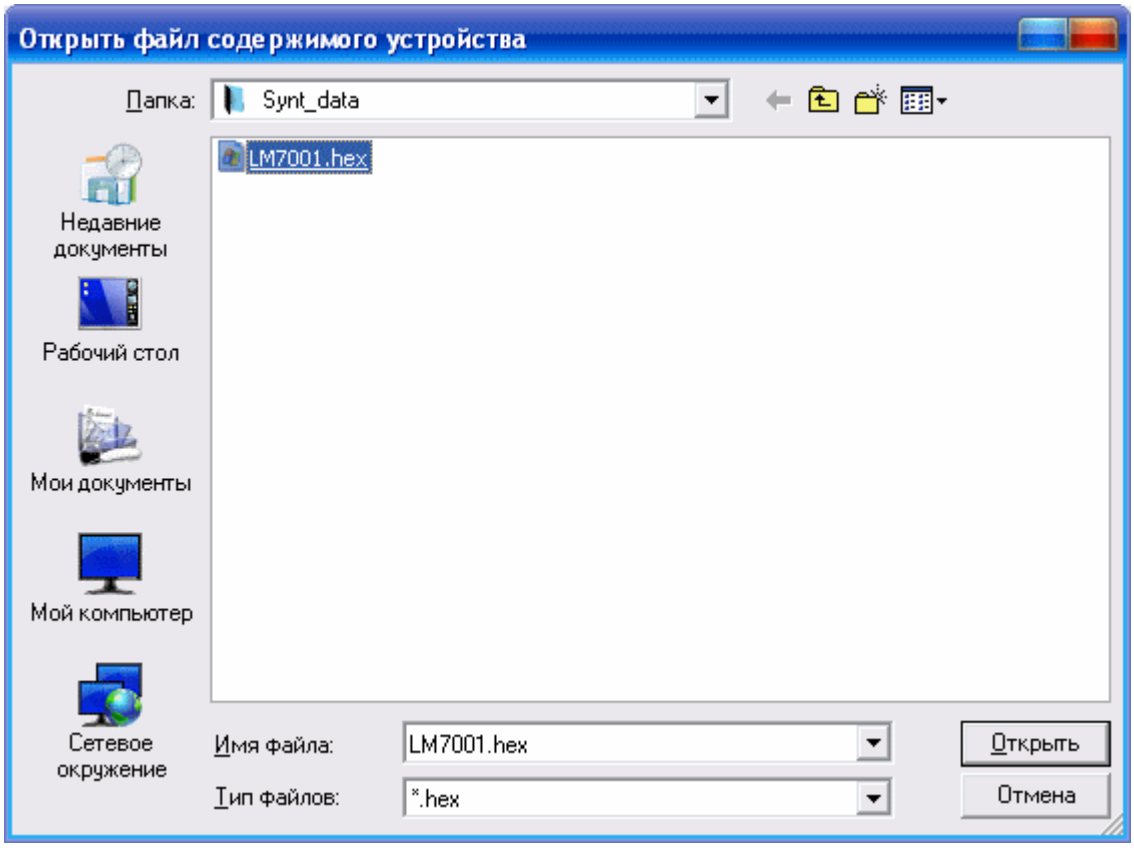
Интересно, что в списке микроконтроллеров семейства AVR имеется строка AVR Auto. Выбрав ее, мы даем программе возможность автоматически распознать вставленную в панель адаптера микросхему этого семейства. Дело в том, что все они снабжены специальным внутренним ПЗУ, в котором хранится "сигнатура" — три байта, однозначно определяющих тип микросхемы. Значения сигнатур указаны в справочных данных (datasheet-ах) микросхем.

Однако программатор сможет прочитать сигнатуру лишь в том случае, если в микросхеме не включена защита кода, делающая все содержимое ее памяти недоступным для внешнего мира. Подобной защитой охотно пользуются производители микроконтроллерных устройств, стремясь защитить свою продукцию от пиратского копирования. Вдобавок ко всему они еще и стирают название микросхемы с ее корпуса.
Напомним еще раз, PonyProg не следит за правильностью выбора адаптера и программируемой микросхемы. При их несовместимости в лучшем случае программирование не будет выполнено вообще или выполнено с ошибками. В худшем — микросхема может быть повреждена.

# *Загрузка исходных данных*

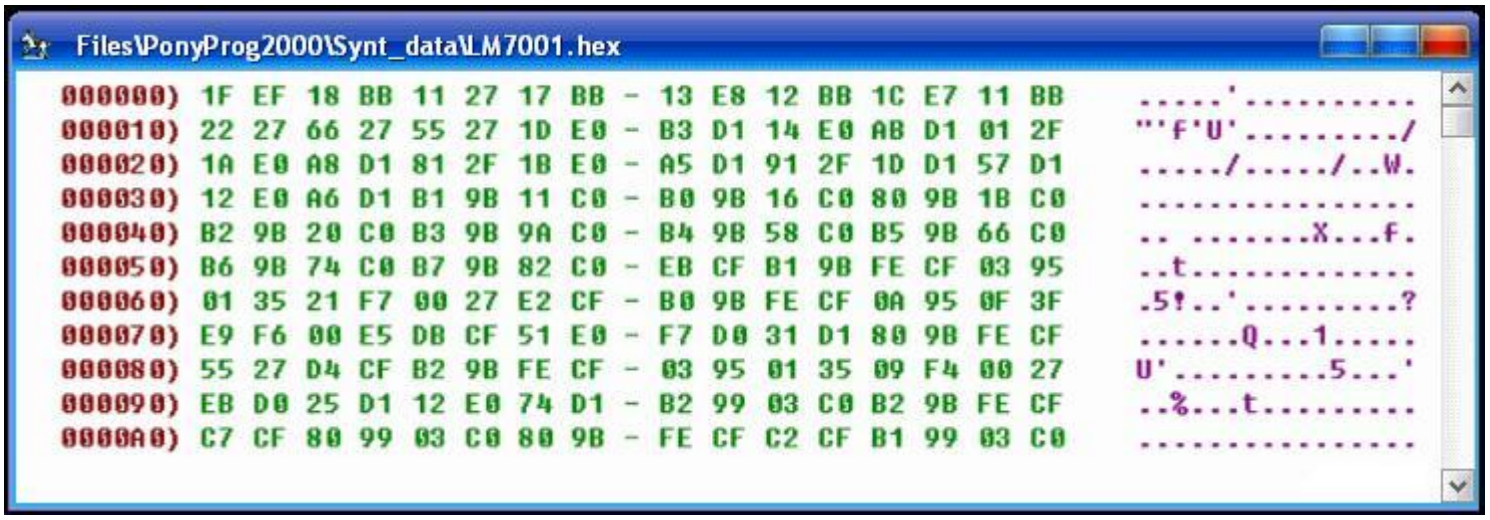
Коды, которые предстоит загрузить в микроконтроллер, обычно представлены одним или несколькими текстовыми файлами разработанного фирмой Intel и ставшего фактически стандартным HEX-формата. Лучше всего "скачать" нужные файлы из Интернета  или получить их в электронном виде каким-либо другим образом. Это гарантирует отсутствие многих ошибок, допускаемых при "ручном" вводе данных. Иногда исходные данные представлены файлами формата BIN ("сырой" двоичный - raw binary). Это точная копия содержимого памяти микроконтроллера без каких-либо служебных и вспомогательных данных. На других понятных PonyProg форматах останавливаться не будем, так как встречаются они довольно редко. Упомянем лишь формат E2P, разработанный специально для PonyProg, но непонятный, к сожалению, другим программам. В файлах этого формата кроме данных для FLASH и EEPROM записан тип микроконтроллера и введенный пользователем текстовый комментарий, в котором могут содержаться любые полезные ему сведения.

Итак, выберем пункт "Файл" главного меню, а в нем — один из пунктов "Открыть файл с данными...", "Открыть файл программы (FLASH)..." или "Открыть файл данных (EEPROM)...". Первым пунктом пользуются, если загружаемый файл содержит информацию для всех областей памяти программируемой микросхемы. Таковы файлы формата E2P, а также HEX-файлы для микроконтроллеров семейства PICmicro. Второй и третий пункты загружают в соответствующие области памяти микроконтроллера данные из разных файлов. Учтите, что имена HEX-файлов для загрузки EEPROM микроконтроллеров семейства AVR обычно имеют расширение .eep.
При выборе одного из упомянутых пунктов на экране откроется окно.

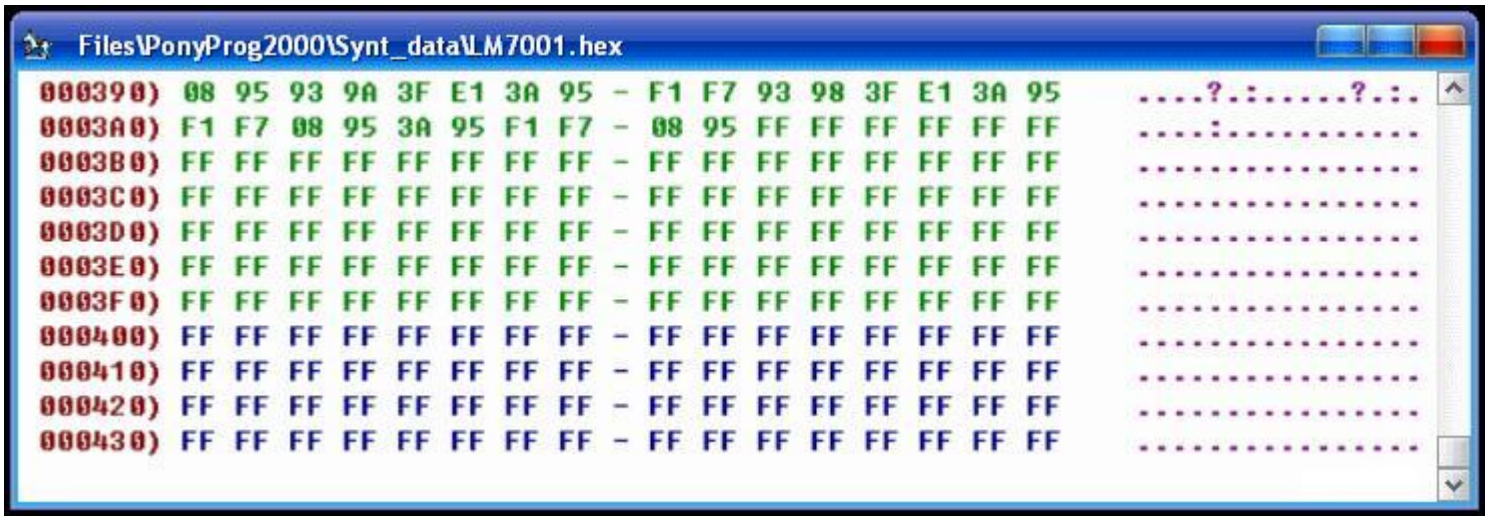



В списке будут содержаться лишь те файлы, имена которых имеют расширение, указанное в окне "Тип файлов". Нажав в нем кнопку со стрелкой вниз, можно перейти к файлам с другими расширениями или (выбрав "*") получить список всех файлов, имеющихся в папке. Файл будет загружен после двойного щелчка по его имени в списке либо после одинарного щелчка (его имя появится в окне "Имя файла" и нажатия на кнопку "Открыть". Можно также ввести нужное имя непосредственно в окно "Имя файла" с клавиатуры.

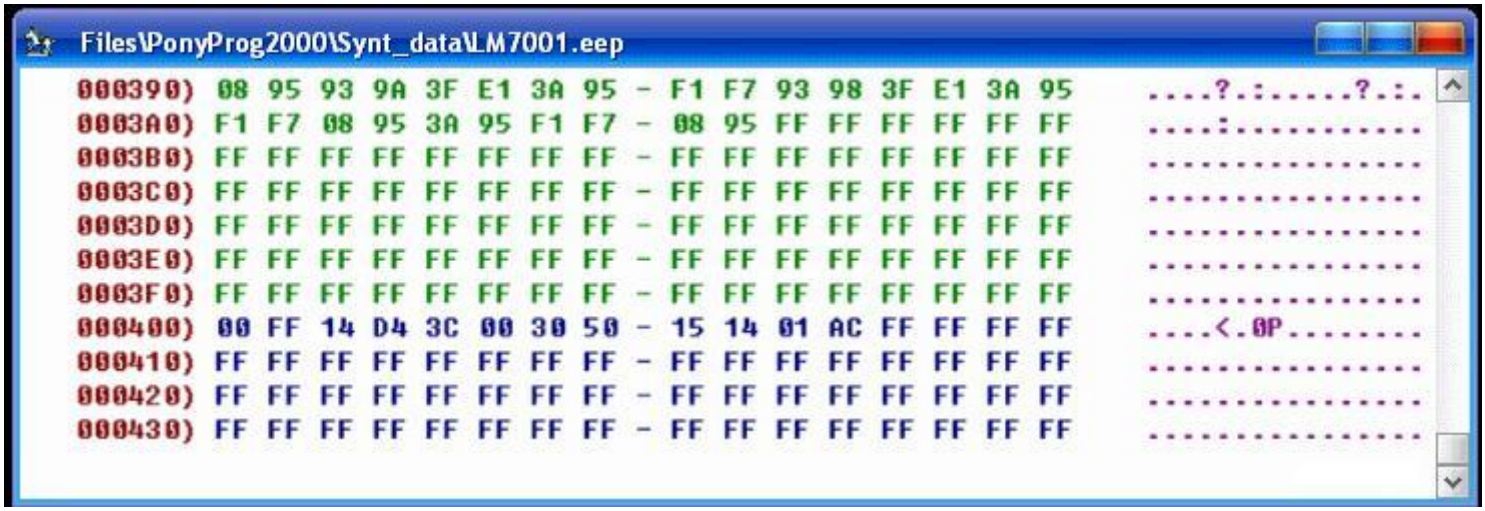
В результате на экране появится окно, озаглавленное именем загруженного файла. Строго говоря, оно находилось там и раньше, но называлось (в зависимости от версии PonyProg) "No Name" или "default" и было пустым. Теперь здесь кодовая таблица, отображающая загруженные данные.




Она состоит из строк, начинающихся с отделенного скобкой адреса первого байта строки. Затем следуют шестнадцатиричные значения 16-ти байтов с последовательно возрастающими адресами (для удобства они разделены тире на две группы по восемь) и далее — символьное представление тех же байтов. Обратите внимание на нижнюю часть таблицы.



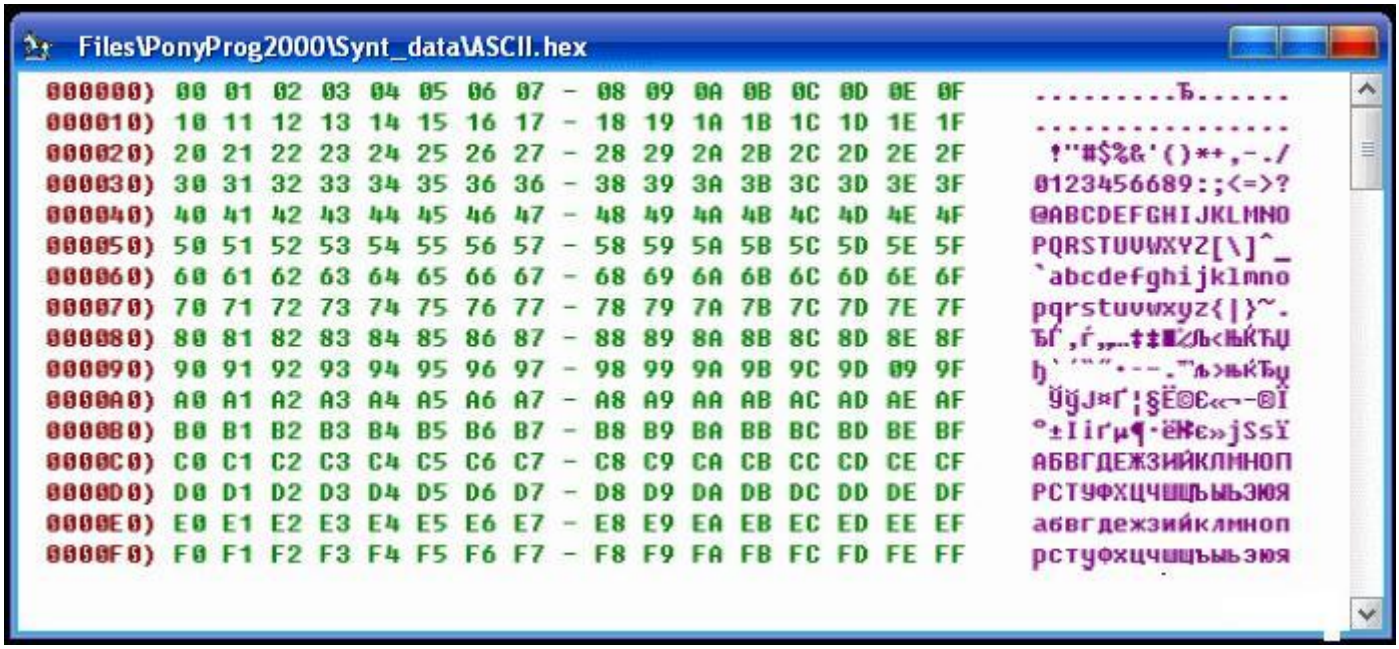

Последние строки, выделенные цветом, отображают содержимое буфера памяти данных (EEPROM). Команда "Открыть файл программы (FLASH)..." оставляет его незаполненным. Информация здесь появится только после выполнения команды "Открыть файл данных (EEPROM)...". Обратите внимание, что изменилось и имя в заголовке окна. Оно соответствует последнему загруженному в буфер файлу.




Необходимо учитывать, что адреса ячеек буфера EEPROM не соответствуют действительным адресам ячеек этой области памяти микроконтроллера. В PonyProg они просто продолжают адресацию буфера памяти программ (FLASH). В рассматриваемом случае байты по адресам 400H— 43FH отображают содержимое ячеек EEPROM микроконтроллера AT90S1200 с адресами 0—3FH.

Правая часть окна буфера программатора, как уже сказано, отображает его содержимое в символьном виде. Это полезно, если в программных кодах имеются какие-либо текстовые сообщения. Чаще всего — предназначенные для вывода на индикатор микроконтроллерного прибора. Но иногда автор программы "маскирует" внутри нее какие-либо дополнительные сведения, например название программы, номер ее версии, а то и собственные фамилию и имя, и даже номер телефона и адрес. Просматривая шестнадцатиричный код, все это трудно заметить, зато в символьном виде такая информация сразу бросается в глаза.
Байты со значениями 0—7FH всегда отображаются символами одинаково — в соответствии с кодовой таблицей ASCII (American Standard Code for Information Interchange). К сожалению этого не скажешь о байтах со значениями 80H—0FFH. Здесь имеется множество вариантов, зависящих как от особенностей настройки операционной системы компьютера, так и от режима отображения таких байтов, выбранного автором программы при ее разработке.
Даже разные версии PonyProg ведут себя неодинаково. "Русифицированная" v. 2.05a при выводе на экран заменяет все символы второй половины кодовой таблицы (в том числе русские буквы) точками.

Нерусифицированная v. 2.07c выводит их правильно, в полном соответствии с "кодовой страницей 1251".



Изображено окно буфера программатора с загруженным в него специально подготовленным файлом, содержащим последовательность байтов 0—0FFH

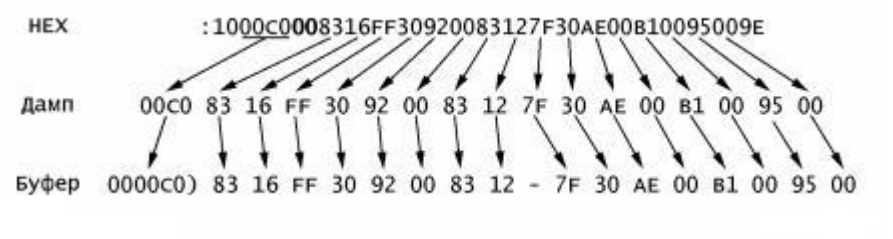
А у меня нет файла с программой микроконтроллера...
Такая ситуация возникает у тех, кто собирает устройство на микроконтроллере, если программу для него не удалось найти в Интернете или получить в электронном виде из какого-либо другого источника. Есть только "твердая копия" кодов программы, напечатанная в журнале. И ее вполне достаточно.
Есть много способов "набрать" нужный для программирования HEX-файл по опубликованной таблице кодов. Это можно сделать, например, с помощью программы CheckHex.

В "шестнадцатиричной" части окна у ячейки с нулевым адресом появится мигающий курсор. Если теперь нажать левую нопку мыши или клавишу Enter, будет открыто окно редактирования содержимого ячейки.
В нем отображено текущее значение кода, находящегося в выбранной ячейке памяти, в шестнадцатиричном, десятичном и символьном виде. В одном из этих форматов введите новое значение. В каком именно — безразлично. Учтите, при изменении содержимого одного из "окошек" значения в двух других останутся прежними. Тем не менее по завершении редактирования в ячейку будет записано именно вновь введенное значение. Если случайно или преднамеренно в разных форматах введены коды, двоичное представление которых не совпадает, приоритет будет отдан шестнадцатиричному, а если его не изменяли — десятичному значению.

Ввод кодов в разных форматах имеет некоторые особенности. Например, если шестнадцатиричное окно содержит три и более цифры, учтены будут лишь две старших (левых). Аналогично ведет себя и символьное окно, но в нем имеет значение только один, самый левый символ. А при вводе в десятичное окно числа, находящегося вне интервала 0—255, будут учтены только три старших разряда, причем в ячейку буфера будет записан остаток от деления представленного ими значения на 256.
После нажатия на кнопку OK новое значение кода будет занесено в буфер, а курсор установлен на ячейку, с адресом на единицу больше отредактированной. Повторяя описанные выше действия, можно записать все нужные коды. Делать это последовательно в порядке возрастания адресов ячеек вовсе не обязательно. При необходимости можно перевести курсор на любую нужную ячейку с помощью мыши или нажимая клавиши управления курсором.

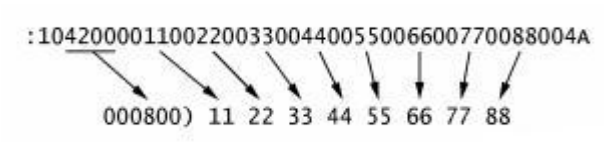

Для ввода длинной строки символов, можно установить курсор на ее начало в правой, символьной части окна буфера. После щелчка мышью появится окно, в которое и вводят нужный текст. Учтите, "старое" содержимое буфера при таком вводе автоматически не уничтожается, а лишь сдвигается в сторону больших адресов. Поэтому прежде, чем нажимать клавишу OK, не забудьте стереть лишнее.

Несколько слов о том, как в публикуемых таблицах "прошивки" микроконтроллера найти нужные для ввода коды. Несколько лет назад подобные таблицы чаще всего представляли собой шестнадцатиричный "дамп" памяти. Они очень похожи на то, что находится в окне буфера РопуРгод, и по этой причине удобны для ручного ввода. Позже в связи с распространением программаторов, читающих исходные данные из HEX-файлов, перешли на публикацию текста этих файлов.
Строки в формате HEX содержат те же (с небольшими дополнениями, облегчающими компьютерный анализ) данные, что и строки дампа.

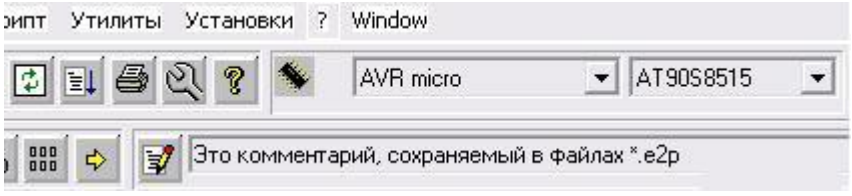



Выделенные полужирным шрифтом нули в восьмой и девятой позиции строки HEX-файла — признак, что она содержит данные. Строки с другими символами в этих позициях — служебные, на них при ручном вводе, как правило, можно не обращать внимания. В строке не обязательно 16 байтов данных, может быть и больше, и меньше. Но адрес всегда относится к первому из них. Два последних символа HEX-строки — контрольную сумму — в буфер не заносят.
Одна из особенностей программы PonyProg заключена в том, что адреса, указанные в HEX-файле, совпадают с адресами ячеек буфера только для программной (FLASH) памяти микроконтроллера. Буфер EEPROM продолжает буфер FLASH-памяти, поэтому адреса его ячеек больше действительных на значение информационной емкости последней.
Например, для микроконтроллера AT90S2313 и других с объемом памяти программ 2 Кбайт буфер EEPROM начинается ячейкой с адресом 800H, которая содержит, однако, код, предназначенный для ячейки EEPROM с нулевым адресом.

Ручной ввод данных, предназначенных для записи в EEPROM микроконтроллеров серии PICmicro, усложняет то, что в отличие от микроконтроллеров многих других серий ассемблер, транслируя программу, помещает эти данные в тот же файл, что и коды программы. Он присваивает им условные адреса, начиная С4200H, причем байты данных чередуются с байтами (как правило нулевыми), не несущими никакой информации. Поэтому вводить данные в буфер EEPROM программатора следует так, как показано на рис.




Набор кодов вручную занимает довольно много времени. Торопиться здесь не стоит, лучше лишний раз убедиться в правильности выполняемых действий. Если не удалось завершить работу за один сеанс, сохраните промежуточный результат, выбрав в меню "Файл" один из пунктов "Сохранить...". В зависимости от выбранного пункта будет сохранен весь буфер либо только FLASH, либо только EEPROM. Предварительно будет задан вопрос, какое имя присвоить файлу. Учтите, при некоторых неправильных действиях, может появиться сообщение об ошибке записи. В подобном случае попробуйте сохранить данные FLASH и EEPROM в разных файлах или в другом формате.
Если не предполагается пользоваться другими (кроме PonyProg) программами управления программированием, можно сохранить информацию в формате *.e2p. Кроме содержимого всех областей памяти в такой файл будет записан тип микроконтроллера и текстовый комментарий, который вводят, выбрав в меню "Правка" пункт "Правка комментария". Набранный текст выводится в верхней правой части окна PonyProg.

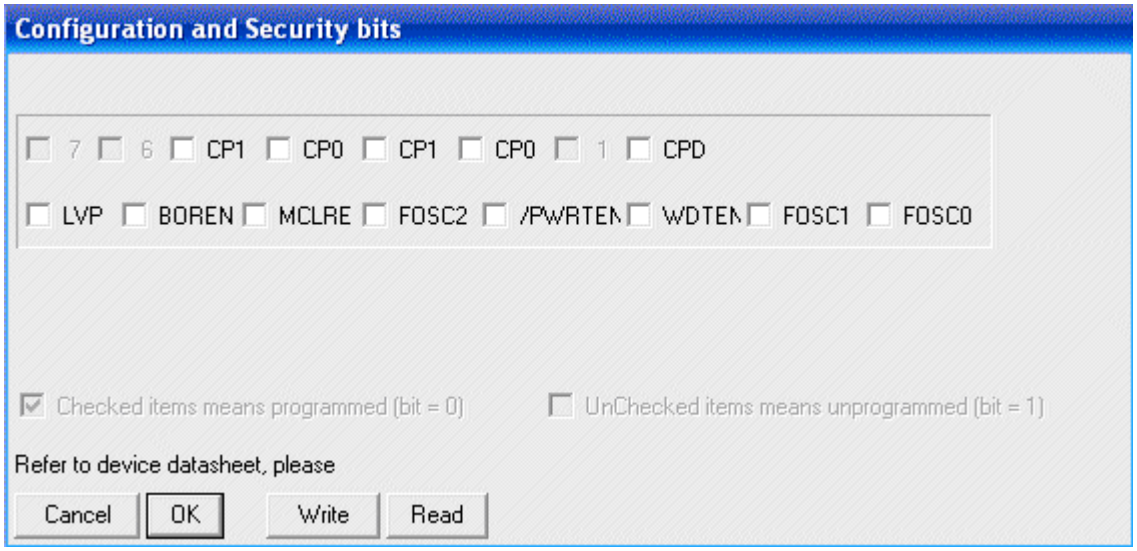



Записав промежуточный результат, PonyProg можно закрыть. Чтобы продолжить работу в удобное время, достаточно, запустив PonyProg, загрузить в буфер сохраненный файл (файлы).

# *О конфигурации.*

Среди областей внутренней памяти микроконтроллера есть одна, о содержимом которой, описывая конструкцию на микроконтроллере, часто забывают рассказать. Это так называемые биты конфигурации, известные также под названиями Locks ("замки") и Fuses ("плавкие вставки"). Записывая в эту область нули и единицы, задают режимы работы узлов микроконтроллера, в том числе тактового генератора и сторожевого таймера, изменяют функциональное назначение некоторых выводов микросхемы.
Важное значение имеет и возможность, задав соответствующую конфигурацию, запретить доступ с помощью программатора к внутренней памяти микроконтроллера. Однако пользоваться этой возможностью следует с большой осторожностью и только при полной уверенности, что защищаемая программа загружена без ошибок и работоспособна. После включения защиты найти ошибки в содержимом памяти программ и внести какие-либо изменения уже не удастся.
Число и назначение битов конфигурации у разных микроконтроллеров неодинаково. Точную информацию о них лучше всего получать из описаний (datasheets) соответствующих приборов. Например, у микроконтроллеров серий AT90, AT89S с помощью программатора последовательного типа можно лишь включить защиту памяти. У приборов серий ATtiny, ATmega, PICmicro возможности изменения конфигурации значительно шире.

Окно управления конфигурацией открывается в PonyProg при выборе пункта меню "Команды" — "Security and Configuration Bits...". Почему-то его название не переведено на русский язык даже в "русифицированной" программе. Вид окна зависит от типа микроконтроллера.




Если загруженный в буфер программатора файл содержал данные о конфигурации, в окошках рядом с названиями битов будут расставлены "галочки".

При необходимости их можно убрать или поставить самостоятельно, щелкая мышью по окошкам.
Назначение битов в рассматриваемом случае следующее:
CP1, CP0, CPD — защита кода, если значения всех этих битов равны 1 ("галочек" нет), она отключена. Чтобы предотвратить случайное включение, биты CP1 и CP0 дублированы. Лишь занеся в оба "дубля" одинаковые значения, можно задать один из возможных режимов защиты.
LVP — низковольтное программирование разрешено (1) или не разрешено (0). В первом случае для перевода микросхемы в режим программирования напряжение +12 В не требуется. Изменять без надобности состояние этого бита не следует. Если программатор читает содержимое памяти микроконтроллера — бит установлен правильно. В противном случае поможет только замена программатора.
BODEN — внутренний детектор понижения напряжения питания включен (1) или выключен (0). Включать детектор следует только при уверенности, что в загружаемой в микроконтроллер программе предусмотрено его использование.

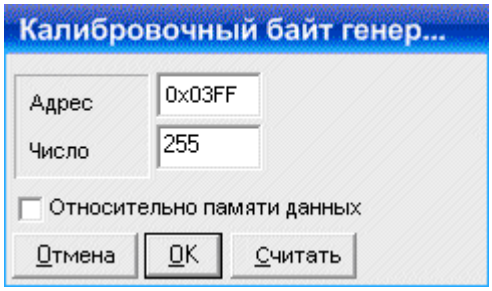
MCLRE — вывод 4 микросхемы служит входом сигнала установки микроконтроллера в исходное состояние MCLR (1) или обычным цифровым входом RA5 (0).
/PWRTEN — таймер задержки пуска микроконтроллера после подачи напряжения питания выключен (1) или включен (0). Обычно его включают, чтобы дать время на "раскачку" тактовому генератору с кварцевым резонатором.
WDTEN — сторожевой таймер (WDT) включен (1) или выключен (0). Ошибочное включение этого таймера нередко бывает причиной того, что запрограммированный микроконтроллер, начав работать правильно, каждые несколько секунд возвращается в исходное состояние. В подобной ситуации попробуйте выключить WDT.

F0SC2—FOSCO — тип тактового генератора и режим работы выводов 15 и 16 микроконтроллера:
111 — частота внутреннего тактового генератора задана резистором (у PIC16F628) или RC цепью (у PIC16F628A), подключенными к выводу 16, генерируемый сигнал выведен для контроля или другого использования на вывод 15.
110 — аналогично 111, но генератор внешнего выхода не имеет, вывод 15 служит входом/выходом RA6.
101 — внутренний генератор работает без внешних элементов, генерируемый им сигнал выведен на вывод 15, вывод 16 служит входом/выходом RA7.
100 — аналогично 101, но генератор внешнего выхода не имеет, вывод 15 служит входом/выходом RA6.
011 — внутренний генератор не действует. Внешний тактовый сигнал подают на вывод 16, вывод 15 служит входом/выходом RA6.
010 — между выводами 15, 16 подключен высокочастотный (HS) кварцевый резонатор.
001 — между выводами 15, 16 подключен обычный (XT) кварцевый резонатор.
000 — между выводами 15, 16 подключен маломощный (LP) кварцевый резонатор.
Если в описании конструкции на микроконтроллере не указан тип тактового генератора, его удается определить и выбрать нужные значения битов FOSC2— FOSC0, проверив по схеме, какие элементы и цепи подключены к выводам 15 и 16.

К сожалению в документации на микроконтроллеры серии PICmicro нет четких критериев, по которым следует относить резонаторы к группам HS, XT или LP. Чаще всего подходит вариант XT. Но если генератор не возбуждается или работает неустойчиво, а подборка подключенных между его выводами и общим проводом конденсаторов не помогает, попробуйте и другие варианты конфигурации. Возможно параметры использованного при повторении конструкции кварца значительно отличаются от примененного ее автором.
Значения некоторых разрядов слова конфигурации иногда бывают индивидуальными для каждого экземпляра микроконтроллера определенного типа. Например, в PIC12F629 и PIC12F675 двумя старшими разрядами этого слова при заводской регулировке изготовленной микросхемы устанавливают номинальное значение образцового напряжения для ее аналоговых узлов. В подобных случаях необходимо, нажав на соответствующую кнопку в окне "Configuration and Security Bits", прочитать слово конфигурации новой, еще не подвергавшейся стиранию и программированию микросхемы и позаботиться о том, чтобы значения этих разрядов после программирования остались прежними.

Многие современные микроконтроллеры оснащены внутренним тактовым генератором, не требующим для своей работы никаких внешних элементов.

Частоту этого генератора подстраивают программно, изменяя код в специальном регистре микроконтроллера. Значение кода, соответствующее номинальной частоте генератора (обычно из ряда 1, 2, 4 или 8 МГц), найденное для данного экземпляра микроконтроллера при заводской настройке, записывают в его память.
У микроконтроллеров фирмы Atmel это специальная область памяти, не стираемая при очистке FLASH-памяти и EEPROM. В некоторых случаях записанное здесь значение переносится в регистр калибровки генератора автоматически при включении питания. В других необходимо "вручную" (предусмотренной для этого командой программатора) прочитать значение калибровочного кода и занести его в определенную ячейку FLASH-памяти или EEPROM.




В PonyProg для работы с калибровочным кодом предусмотрено два пункта меню "Команды". Первый из них ("Считать калибровочный байт ген.") позволяет получить значение этого кода из специальной памяти микроконтроллера и сохранить его в буфере по адресу, указанному с помощью второго пункта ("Настройка калибровки генератора"). Если поставить "галочку" у надписи "Относительно памяти данных", код будет занесен не во FLASH, а в EEPROM.

Хочу предупредить, работа этих пунктов меню производит впечатление не вполне отлаженной. Микроконтроллеры PICmicro, оснащенные внутренним подстраиваемым генератором, обычно хранят "заводское" значение калибровочного кода уже записанным во FLASH-память, как правило, в ее самую старшую ячейку. При неосторожном стирании памяти это значение будет потеряно навсегда. По-этому прежде, чем подавать команду стирания, код необходимо перенести в соответствующую ячейку буфера программатора, в котором уже находятся подготовленные к программированию данные. Но и после этого следует соблюдать осторожность. Код может быть уничтожен, например, при повторной загрузке буфера из файла. На всякий случай запишите его на бумаге или прямо на корпусе микросхемы.

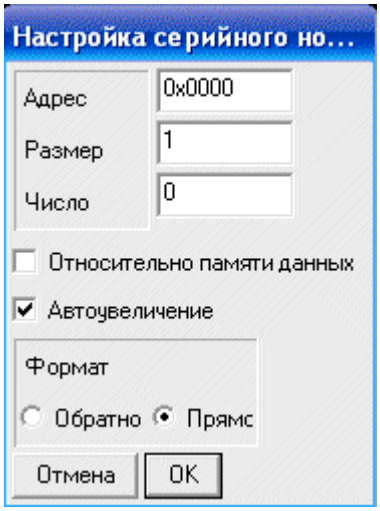
В упомянутом в подзаголовке меню среди пунктов, несомненно, полезных при подготовке к программированию ("Очистить буфер", "Заполнить буфер"), есть и такие, неосторожное исполнение которых может исказить подготовленные к программированию данные, причем заметить внесенные изменения "визуально" очень непросто.
Пункт "Удвоить" увеличивает вдвое объем буфера программирования. Плохо то, что при этом каждый записанный в нем ранее байт повторяется дважды в соседних ячейках. К счастью, в последних версиях PonyProg исполнение этого пункта для микроконтроллеров заблокировано.
Пункт "Переставить байты" меняет местами четные и нечетные байты буфера. Чтобы вернуть содержимое буфера в исходное состояние, достаточно выбрать этот пункт повторно.

а)
```
000000) 4C 28 FF FF FF FF FF FF - F2 00 03 08 F3 00 04 08
000010) F4 00 36 30 86 00 86 1F - 0F 28 75 08 F7 04 75 09
000020) 86 1B 13 28 F7 05 F1 03 - 71 08 03 1D 19 28 03 30
```

б)
```
000000) 28 4C FF FF FF FF FF FF - 00 F2 08 03 00 F3 08 04
000010) 00 F4 30 36 00 86 1F 86 - 28 0F 08 75 04 F7 09 75
000020) 1B 86 28 13 05 F7 03 F1 - 08 71 1D 03 28 19 30 03
```

в)
```
000000) 65 00 FF FF FF FF FF FF - F2 00 03 08 F3 00 04 08
000010) F4 00 36 30 86 00 86 1F - 0F 28 75 08 F7 04 75 09
000020) 86 1B 13 28 F7 05 F1 03 - 71 08 03 1D 19 28 03 30
```

Пункт "Установить серийный номер" используют для того, чтобы при записи одной и той же программы в несколько микросхем пронумеровать экземпляры. Результат его выполнения показан на рис, в. Как видим, номер занесен в первые две ячейки буфера FLASH-памяти, в результате чего потеряны находившиеся там программные коды. Работать такая программа, естественно, не будет.

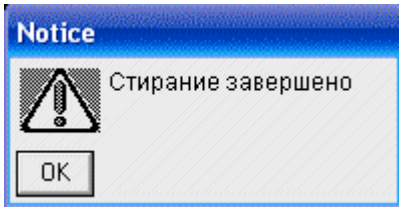



Если нумерация действительно необходима, следует позаботиться о том, чтобы номер оказался записанным в заведомо не используемые программой ячейки. Установить адрес, по которому будет записан номер, позволяет пункт "Установка серийного номера", открывающий окно, показанной на рис. Кроме адреса, здесь можно задать длину ("Размер") номера в байтах, порядок их следования ("Формат") и начальное значение ("Число"). Формат "Обратно" соответствует общепринятому порядку записи младшего байта по меньшему адресу (равен указанному в окне "Адрес"), а старшего — по большему. В формате "Прямо" порядок следования байтов от старшего к младшему. Если выбран режим "Автоувеличение", значение номера автоматически возрастает на единицу после каждого выполнения пункта "Установить серийный номер", в противном случае оно остается неизменным.
Чтобы надежно обнаружить непреднамеренное искажение подготовленных к программированию данных случайным выполнением "опасных" операций, рекомендуется запомнить CRC правильно заполненного буфера и сверить его с фактическим значением непосредственно перед программированием. CRC (Cyclic Redundance Code — циклический избыточный код) вычисляют по специальному алгоритму. В отличие от обычной контрольной суммы, он чувствителен к изменениям не только значений в ячейках буфера, но и порядка их следования.

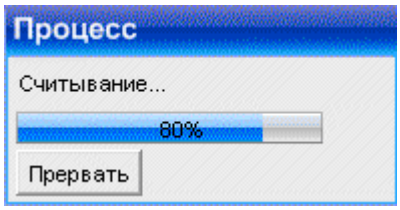

Как показано на рис, текущее значение CRC выведено в нижней части (строке состояния) главного окна PonyProg вместе с выбранным типом программируемой микросхемы и объемом ее памяти (суммой объемов FLASH и EEPROM). Те же сведения можно получить, выбрав в меню "Команды" пункт "Информация". К сожалению, CRC не указывает, значение какой именно ячейки изменено. Обнаружив искажение, придется либо просмотреть и сравнить с требуемыми значения по всех ячейках буфера, либо повторить операции по подготовке данных заново.

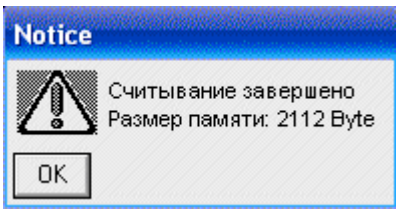
## *Программирование*

Многие оболочки программирования позволяют убедиться в готовности установленной в адаптер микросхемы к программированию командами вроде "Проверить на чистоту". В PonyProg такой возможности нет. Чтобы очистить память микроконтроллера от возможно содержавшейся в ней ранее информации, необходимо выбрать в меню "Команды" пункт "Стереть" и получить сообщение, показанное на рис. Если этого не сделать, в некоторые ячейки, возможно, не удастся записать нужные коды, так как программатор не в силах заменить ноль в разряде ячейки единицей. Конечно, перед стиранием следует быть уверенным, что память не содержит ценной информации.

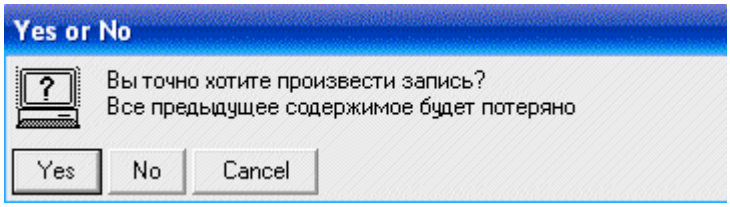


Если старое содержимое памяти может пригодиться, его, прежде чем стирать, следует прочитать и сохранить в файле. Если в буфере программирования уже находится подготовленная к записи информация, нужно создать еще один буфер с помощью пункта "Новое окно" меню "Файл". Затем перейти к меню "Команды" и выполнить один из пунктов "Считать все", "Считать программу (FLASH)" или "Считать данные (EEPROM)" в зависимости оттого, какая область памяти представляет интерес. На экране появится информация о ходе считывания, а затем — о его завершении

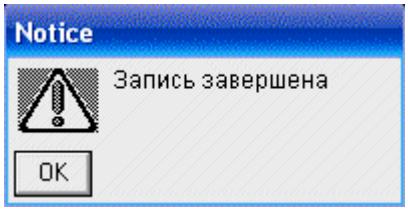


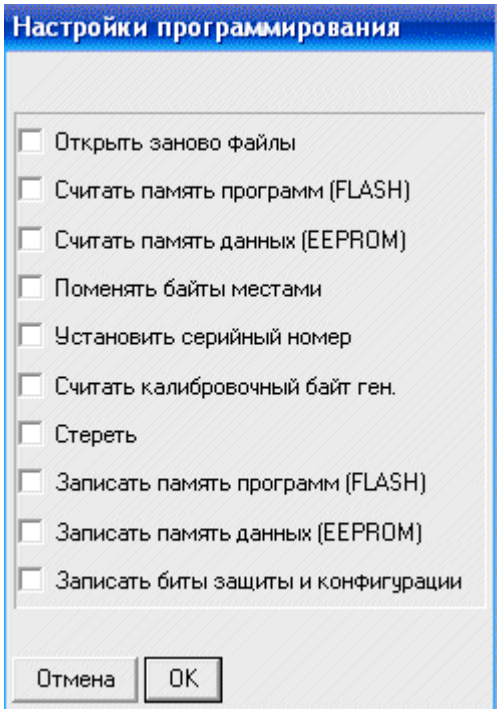

Теперь можно, вернувшись в меню "Файл", сохранить информацию, как это делалось при ручной подготовке данных . После этого память микросхемы очищают командой "Стереть", а ненужное более окно закрывают.




После возвращения в окно с подготовленными данными не спешите выбирать пункт "Программирование". О его использовании мы поговорим позже. Чтобы загрузить данные из буфера в микросхему, необходимо, в зависимости от того, какие из областей ее памяти вы собираетесь запрограммировать, выбрать один из пунктов "Записать все", "Записать программу (FLASH)", "Записать данные (EEPROM)". На экране появится предупреждение (рис.) — немного запоздалое, так как все "предыдущее содержимое" уже стерто.




Запись начнется после нажатия на кнопку "Yes". О ее ходе сообщит окно,  подобное показанному на рис, но с названием процесса — "Запись...". По ее завершении будет автоматически выполнена сверка фактического содержимого памяти микросхемы с содержимым буфера, о ходе которой сообщит окно "Процесс - Проверка...". Если ошибок нет, об этом на экран будет выведено сообщение (рис.). Теперь запрограммированный микроконтроллер можно извлекать из панели адаптера и устанавливать туда, где он должен работать.




Сверку содержимого памяти и буфера можно произвести и с помощью команд "Проверить все", "Проверить программу (FLASH)" или "Проверить данные (EEPROM)". Но следует предостеречь — эти команды иногда сообщают о несуществующих ошибках. Дело в том, что FLASH-память микроконтроллеров серии PIC16 четырнадцатиразрядная. Максимальное значение кода в ячейке этой памяти — 0x3FFF. В буфере PonyProg под этот код отведено 16 разрядов (два байта), значение кода в которых после очистки буфера — OxFFFF. Некорректное сравнение этих значений и воспринимается как ошибка программирования. Так как подобные "ошибки" фиксируются не всегда, их анализ, по-видимому, ведется по-разному в разных ветвях алгоритма сравнения.

И наконец, о команде "Программирование". Прежде чем ею воспользоваться, необходимо выбрать пункт меню "Настройки программирования..." и расставить "галочки" в окне, показанном на рис. Теперь при каждом выборе пункта "Программирование" отмеченные команды будут выполнены автоматически в той последовательности, в которой они перечислены в окне.