

ПРОГРАММАТОРЫ И ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ

А. ДОЛГИЙ, г. Москва

"Радио", 2004, № 1, с. 51—55

Ставшие сегодня обычными радиолюбительские конструкции на микроконтроллерах подкупают простотой схемы и широкими возможностями. Однако прежде, чем собранное устройство заработает, микроконтроллер, в отличие от традиционных интегральных микросхем, выпускаемых с завода "готовыми к употреблению", необходимо "обучить". Для этого в его память нужно занести программу — последовательность команд, исполняя которые микроконтроллер будет делать все, что требуется.

Программу обычно составляют параллельно с разработкой схемы и конструкции прибора и окончательно отлаживают, испытывая готовое изделие. Мы не будем вникать в многочисленные возникающие при этом проблемы. Наша задача — научиться заносить в микроконтроллер готовую программу.

Что будем записывать

Обычно в описании прибора программа, по которой должен работать микроконтроллер, представлена таблицей кодов — их следует занести в память микроконтроллера. Вот фрагмент такой таблицы в так называемом HEX-формате (табл. 1):

Таблица 1

```
:1000C000A60C2618652805126728051600003F2093  
:1000D000A60C26186D2805126F28051600003F2073  
:1000E000A60C2618752805127728051600003F2053  
:1000F000A60C26187D2805127F28051600003F2033  
:10010000A60C2618852805128728051600003F2012  
:10011000000000000000000000000005163F203F2006  
:1001200000000000000000000000000800FF3FFF3F4B
```

Подробнее о нем можно прочитать, например в [1, 2].

Напомним, микроконтроллеры работают в двоичной системе счисления, различая лишь по два состояния (0 и 1) каждого из своих выводов, многочисленных ячеек памяти и других внутренних узлов. Большинство двоичных ячеек-разрядов для ускорения и удобства выполнения различных операций над их содержимым объединены в группы — восьмиразрядные байты и более длинные слова. В публикуемых таблицах исключительно для удобства их восприятия человеком помещают не двоичные, а шестнадцатиричные изображения программных кодов. Каждые четыре двоичных цифры заменяют одной шестнадцатиричной:

0000 — 0	0100 — 4	1000 — 8	1100 — C
0001 — 1	0101 — 5	1001 — 9	1101 — D
0010 — 2	0110 — 6	1010 — A	1110 — E
0011 — 3	0111 — 7	1011 — B	1111 — F

Куда будем записывать

Внутреннюю память программ микроконтроллера в варианте, допускающем многократное изменение содержимого, часто называют FLASH-памятью, хотя этот термин характеризует ее устройство и принцип действия, а не функциональное назначение. Кроме программной имеется, как правило, и внутренняя электрически изменяемая память данных (EEPROM) для хранения различного рода констант, подбираемых в процессе настройки готового изделия или время от времени корректируемых в процессе эксплуатации.

Частью внутренней перепрограммируемой памяти микроконтроллера можно считать и конфигурационные ячейки — своего рода переключатели, управляющие некоторыми узлами микроконтроллера. Записав в эти ячейки нули или единицы, такие узлы можно включить, выключить, установить нужный режим их работы. Учтите, сделать это можно лишь в процессе программирования. Исполняемая микроконтроллером программа ни проверить его конфигурацию, ни изменить ее не может. Именно неправильно заданная конфигурация часто единственная причина неработоспособности отлаженной и неоднократно проверенной программы.

Среди конфигурационных есть ячейки, управляющие защитой памяти. Включив ее, делают содержимое внутренней памяти недоступным для любых

внешних воздействий кроме полного стирания. Только уничтожив всю хранившуюся информацию, удастся отключить защиту. Таким образом предотвращают несанкционированное копирование программы. Учтите, однажды включенную защиту памяти микроконтроллера, рассчитанного на однократное программирование (OTP), уже не отключить никаким способом.

Как будем записывать

В обычном рабочем режиме внутренняя память микроконтроллера для внешнего доступа закрыта. Чтобы получить возможность читать и записывать информацию, нужно подать на определенные выводы микросхемы специальную комбинацию уровней напряжения, не встречающуюся при обычной работе. У большинства современных микроконтроллеров для программирования предусмотрен последовательный интерфейс (не путать с интерфейсом, служащим для связи микроконтроллера с другими устройствами в рабочем режиме). Это значит, что двоичные значения кодов и команд, управляющих процессом программирования, подают на предназначенный для этого вывод микросхемы поочередно разряд за разрядом, сопровождая синхронизирующими импульсами на другом выводе. Таким образом в процессе программирования активно участвуют всего две цепи. Иногда требуется еще одна — для вывода (тоже последовательным кодом) содержимого памяти и ответов микроконтроллера на команды, управляющие программированием. Но во многих случаях двустороннюю связь организуют по одной цепи.

Переход с параллельного интерфейса программирования, требовавшего задействовать почти все выводы микроконтроллера, на последовательный привел к значительному упрощению программаторов — устройств для занесения информации во внутреннюю память. Еще больше упростил их перенос внутрь микроконтроллеров довольно сложной автоматики программирования, формирующей импульсы строго определенной длительности и повышенного напряжения. Внутри переместились и сами источники этого напряжения. Внешнее напряжение, отличающееся от обычного, требующегося для питания микроконтроллера если и подают, то лишь как сигнал переключения последнего в режим программирования.

В аппаратной части программатора остаются один-два электронных ключа, и два-три буферных элемента, согласующих входы и выходы микроконтроллера с

внешними цепями. Основная часть работы по формированию и анализу последовательностей импульсов во время программирования возложена на управляющий этим процессом компьютер.

Часто задают вопрос, нельзя ли записать программу в микроконтроллер, не имея компьютера? Ответ неожиданный: можно. В принципе достаточно переключателя, чтобы устанавливать на входе программирования уровни 0 или 1 в соответствии с кодовой таблицей, и кнопки для подачи синхроимпульсов. Естественно, кнопка и переключатель должны быть снабжены узлами подавления "дребезга" контактов. Оперирова этими органами вполне возможно занести в память микроконтроллера всю программу.

Беда в том, что самая простая программа состоит из нескольких сотен нулей и единиц, а в более сложных их тысячи. Многие ли обладают достаточным терпением и аккуратностью, чтобы безошибочно все это набрать вручную? Ведь после любого сбоя придется повторять работу с самого начала. Лучше уж поручить эти однообразные операции компьютеру.

Программатор = адаптер + управляющая программа

Прежде, чем продолжить рассказ, разберемся немного в терминологии. Слово "программатор" употребляют сегодня как минимум в двух значениях. Во-первых, это устройство, с помощью которого соединяют компьютер (источник данных) с микросхемой, в которую должны быть занесены эти данные. Этот прибор правильнее называть адаптером программирования. Во-вторых, программатор — программа, под управлением которой компьютер формирует все необходимые для записи данных в микроконтроллер сигналы на выводах одного из своих портов (того, к которому подключают адаптер).

Путаница в понятиях нередко приводит к взаимному непониманию в спорах о том, какой программатор лучше. Один утверждает, что программатор А, его можно собрать за день. Второй — что программатор Б, он удобнее в пользовании. Действительно, адаптер А очень прост, но сопровождающая его программа А неудобна. Сложный в изготовлении адаптер Б работает под управлением программы Б, оснащенной многими сервисными функциями. Так что оба правы. Но нередко спорящим невдомек, что адаптер А вполне может работать с программой Б и наоборот. Именно такая ситуация будет рассмотрена ниже.

Как подключить адаптер

Для связи компьютера с программируемым микроконтроллером через адаптер пригодны два вида стандартных портов, известных под аббревиатурами LPT и COM. То, что порт LPT параллельный, а COM последовательный, в данном случае значения не имеет. Важна возможность формировать нужные импульсы, программно изменяя уровни напряжения на отдельных линиях этих портов, и "читать" ответные сигналы микроконтроллера. Число практически равноправных выходных и входных линий вполне достаточно в портах обоих типов. Поэтому с точки зрения правильности и скорости программирования подключение адаптера, к порту того или иного типа, не дает никакого выигрыша.

Некоторое преимущество COM-порта состоит в том, что из его выходных сигналов легко получить не только напряжение +5 В мощностью, достаточной для питания самого адаптера и программируемой микросхемы, но и +12 В для переключения в режим программирования микроконтроллеров, например, серии PICmicro (более известных под названием PIC-контроллеры). Здесь удастся обойтись без дополнительного внешнего источника питания, который как правило необходим LPT-адаптерам. Хотя известны конструкции с питанием и от такого порта [3].

К тому же к розетке порта LPT1 в компьютере обычно подключен принтер, который на время работы с программатором приходится отключать. Очень редко компьютеры бывают снабжены вторым параллельным портом LPT2, к которому можно подключить адаптер, не отключая принтер от LPT1. Справедливости ради нужно сказать, что современные принтеры все чаще снабжают интерфейсом USB и порт LPT остается свободным.

Весьма заманчиво было бы использовать USB для связи компьютера с адаптером программирования. Прежде всего потому, что в этом интерфейсе специально предусмотрена подача на подключаемое устройство от компьютера питающего напряжения 5 В. К сожалению, организация аппаратного и программного взаимодействия по этому интерфейсу довольно сложна. Так что разработка USB-программатора — дело будущего.

Последовательных портов в компьютере обычно два (COM1 и COM2), адаптер можно подключить к любому. Но вилки этих портов зачастую разнотипны. Одна из них, как правило, COM1, девяти-, а COM2 — 25-контактная, что нужно учитывать при изготовлении адаптеров. При несовпадении типа разъемов можно



Рис. 1

приобрести переходник или изготовить его по схеме, показанной на **рис. 1**. Для подключения адаптера с 25-контактной розеткой к девятиконтактной вилке порта разъем X1 должен быть розеткой (с буквенным индексом F), а X2 — вилкой (с индексом M). В противоположном случае — наоборот.

Длина соединительных кабелей, шлейфов или проводов компьютер—адаптер и адаптер—программируемая микросхема во всех случаях должна быть минимальной — 0,2...0,5 м. Понятно, это доставит некоторые неудобства, так как разъемы портов размещены на обычно труднодоступной тыльной стороне корпуса системного блока. Но придется смириться с неудобствами. Именно длинные провода — одна из главных причин сбоев в процессе программирования.

SI-Prog — программируем все

Описание сравнительно несложного адаптера, предназначенного для работы под управлением известной программы PonyProg, было опубликовано в [1]. Рекомендации по его проверке можно найти в [4]. Адаптер состоит из основной платы, подключаемой к разъему COM-порта, и нескольких сменных плат-переходников с панелями для программируемых микросхем различных типов, в том числе микроконтроллеров самых распространенных серий PICmicro, AVR и AT89S. Среди радиолюбителей за этим устройством закрепилось название PonyProg, хотя К. Ланконелли (Lanconelli), автор исходного варианта адаптера и обслуживающей его программы, предпочитает называть свой адаптер SI-Prog. Будем следовать его примеру.

Адаптеры для PICmicro

Фирма Microchip (разработчик микроконтроллеров серии PICmicro) рекомендует для их программирования адаптер по схеме, показанной на **рис. 2**.

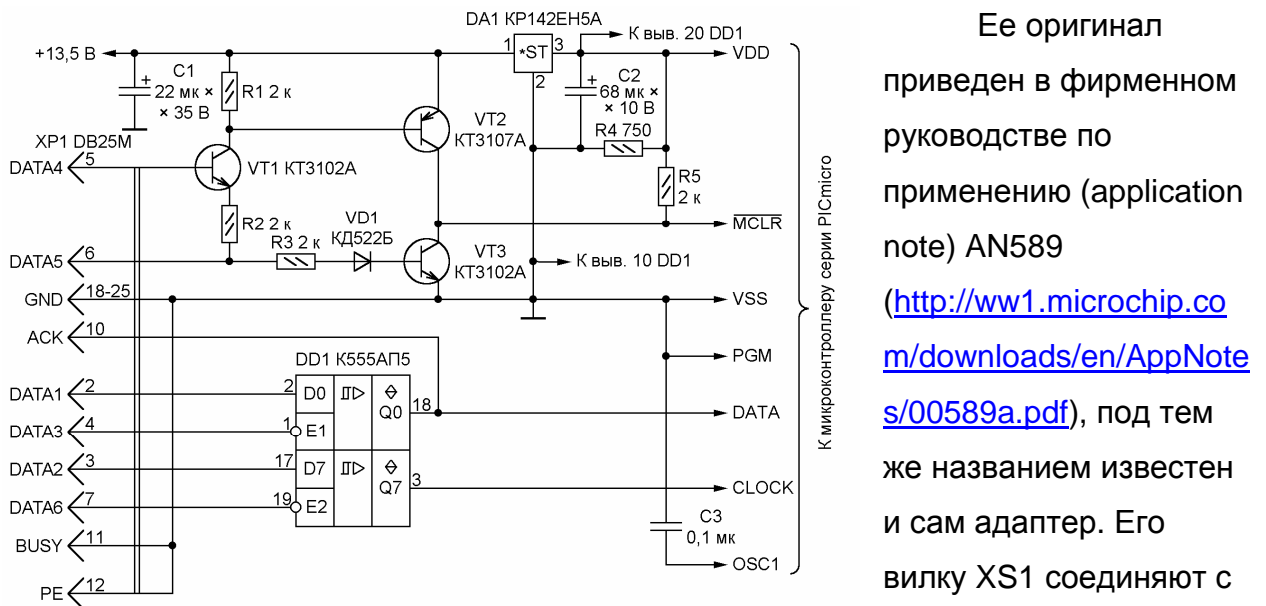


Рис. 2

порта LPT на корпусе системного блока компьютера. Соединение с общим проводом (GND) цепей BUSY и PE дает управляющей программе определить, что к LPT-порту подключен именно адаптер. Если разъем порта свободен или к нему подключен принтер, сочетание логических уровней в этих цепях иное.

Напряжение +13,5 В подают от любого источника, способного отдать ток не менее 50 мА. Микросхема DA1 — стабилизатор напряжения 5 В. На транзисторах VT1—VT3 собран узел управления напряжением в цепи, переводящей микроконтроллер в режим программирования. В зависимости от логических уровней на линиях DATA4 и DATA5 оно принимает три значения: 0, 5 и 12 В.

Выводы различных микроконтроллеров подключают к адаптеру в соответствии с **табл. 2**.

Ее оригинал
приведен в фирменном
руководстве по
применению (application
note) AN589
(<http://ww1.microchip.com/downloads/en/AppNotes/00589a.pdf>), под тем
же названием известен
и сам адаптер. Его
вилку XS1 соединяют с
25-контактной розеткой

Таблица 2

МК	Корпус	MCLR	CLOCK	DATA	PGM	OSC1	VSS (Общ.)	VDD (+Uпит)
PIC12C5xx PIC12C671 PIC12C672 PIC12CE673 PIC12CE674 PIC12F629 PIC12F675	PDIP-8	4	6	7	—	2	8	1
PIC16F630 PIC16F676	PDIP-14	4	12	13	—	2	14	1
PIC1400	PDIP-28	14	11	12	—		20	9
PIC16F83 PIC16F84	PDIP-18	4	12	13	—	16	5	14
PIC16F627 PIC16F628	PDIP-18	4	12	13	10	16	5	14
PIC16F870 PIC16F872 PIC16F873	PDIP-28	1	27	28	24	9	8, 19	20
PIC16F871 PIC16F874 PIC16F876 PIC16F877	PDIP-40	1	39	40	36	13	12, 31	11, 32

Примечание. Назначение выводов модификаций микроконтроллеров в корпусах других типов может не совпадать с указанным в таблице.

"Радио", 2004, № 2, с. 51, 52

Цепь PGM имеется только у микроконтроллеров с двумя вариантами режима программирования: обычным для PICmicro "высоковольтным", включаемым подачей в цепь MCLR напряжения +12 В, и "низковольтным" (low voltage programming, LVP), для которого достаточно +5 В. Режим LVP имеет некоторые отличия от обычного и учтены они далеко не во всех программах, управляющих программированием. Чтобы независимо от установленной ранее конфигурации микроконтроллера разрешить высоковольтное программирование, вывод PGM необходимо соединить с VSS (общим проводом). Учтите, это должно быть сделано до подачи на микроконтроллер напряжения питания.

Иногда рекомендуют включать в цепь PGM последовательно резистор номиналом 240 Ом. Логический уровень на соответствующем выводе микроконтроллера во время программирования останется низким, но вывод будет защищен от перегрузки в случае, если занесенная в память программа "вдруг" начнет исполняться и окажется, что в ней предусмотрена настройка соответствующего разряда порта на вывод и запись в него лог. 1.

К выводу OSC1 в рабочем режиме микроконтроллера подключают один из выводов кварцевого резонатора или другого частото задающего элемента. Хотя тактовый генератор во время программирования заблокирован, некоторые управляющие программы переводят микроконтроллер в этот режим слишком

медленно. Если в процессе переключения генератор все-таки успеет совершить несколько колебаний, исходное состояние программного счетчика микроконтроллера станет не нулевым. Это приведет к записи программных кодов не в те ячейки, для которых они предназначены. Чтобы исключить сбои по этой причине и надежно заблокировать работу тактового генератора, вывод OSC1 рекомендуют на время программирования соединить с общим проводом непосредственно или через конденсатор сравнительно большой емкости (C3 на рис. 2), хотя "официальные" протоколы программирования этого и не требуют.

Не указанные в табл. 2 выводы микроконтроллеров во время программирования можно оставить свободными. Они находятся, как правило, в высокоимпедансном состоянии. Если же к ним все-таки подключены какие-нибудь внешние элементы, на ход и результат программирования они не повлияют.

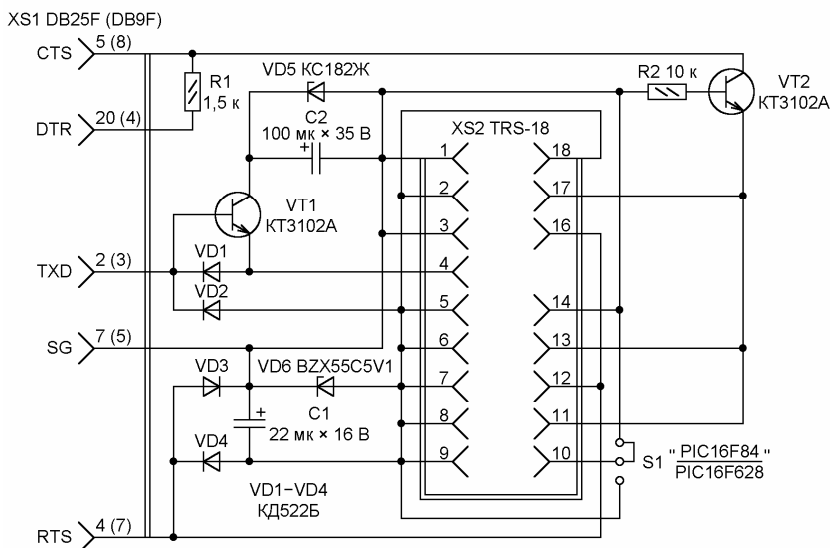


Рис. 3

Среди адаптеров для PICmicro, подключаемых к порту COM и не требующих дополнительного источника питания, популярностью пользуется так называемый JDM. Его схема, изображенная на рис. 3, содержит ряд решений "на грани

фола", тем не менее адаптер зарекомендовал себя с наилучшей стороны.

В исходном варианте он предназначен для микроконтроллеров PIC12C508, PIC12C509, PIC16C84 (устаревший вариант PIC16F84) и микросхем памяти с

интерфейсом I²C. Их программируют, вставляя в панель XS2 различным образом, как показано на рис. 4.

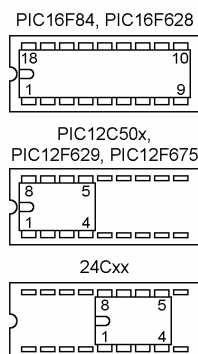


Рис. 4

Имеющаяся на схеме рис. 3 перемычка S1 (в прототипе она отсутствует) позволяет соединить вывод 10 панели XS2 с общим проводом, что необходимо для программирования микроконтроллеров PIC16F628, устанавливаемых в

панель аналогично PIC16F84. Адаптер пригоден и для других микроконтроллеров серии PICmicro (см. табл. 2), если предусмотреть для них соответствующие панели.

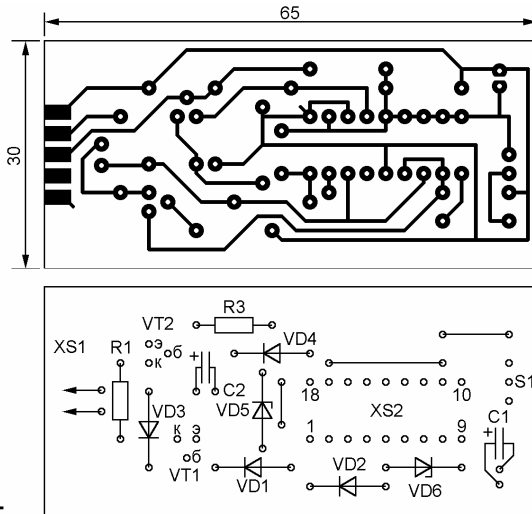


Рис. 5

Печатная плата адаптера JDM с розеткой X1 DB9F и расположение элементов на ней изображены на **рис. 5**.

Плата односторонняя, ее ребром вставляют между рядами выводов розетки XS1, но лишь выводы 1—5 припаивают

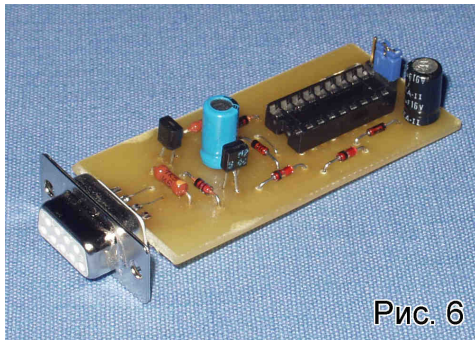


Рис. 6

непосредственно к контактным площадкам.

Выводы 7 и 8 соединяют с соответствующими площадками проволочными перемычками.

Внешний вид готового адаптера показан на

рис. 6.

В некоторых случаях с помощью JDM не удается запрограммировать восьмивыводные

микроконтроллеры PIC16F629 и PIC16F675. Причина этого — слишком большой интервал между включением напряжения питания и подачей команды перехода в

режим программирования. Японский радиолобитель,

адрес электронной почты которого

<mailto:hoheplyohamu@hotmail.com>, предложил

устранить недостаток, дополнив адаптер JDM узлом,

схема которого приведена на **рис. 7**. Цепь, шедшую

ранее к выводу 2 панели XS2 (см. рис. 3), следует

разомкнуть.

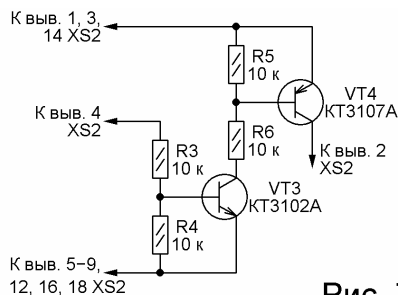


Рис. 7

Адаптеры для микроконтроллеров фирмы Atmel

Схема стандартного адаптера для микроконтроллеров фирмы Atmel серий AVR (в нее входят микросхемы с названиями, начинающимися с AT90, ATmega, ATtiny) и AT89S показана на **рис. 8**.

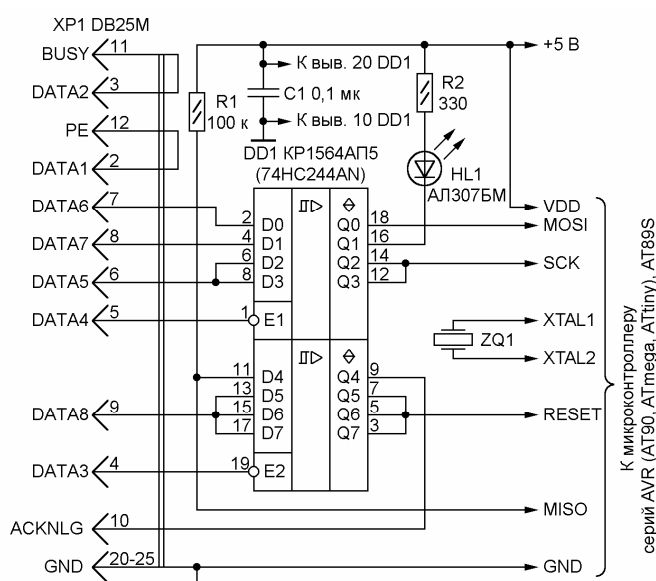


Рис. 8

сигнализации о режиме работы устройства, но делает это лишь в том случае, если управляющая программа вырабатывает соответствующий сигнал.

Напряжение + 5 В подают от внешнего стабилизированного источника, более высокое здесь не требуется.

С выводами программируемой микросхемы адаптер соединяют в соответствии с **табл. 3**.

Таблица 3

МК	Корпус	RESET	MOSI	MISO	SCK	XTAL1	XTAL2	VCC (+Uпит)	GND (Общ.)
AT90S2323 AT90S2343 ATtiny12 ATtiny13	PDIP-8	1	5	6	7	2	3	8	4
ATtiny15	PDIP-8	1	5	6	7	—	—	8	4
AT90S1200 AT90S2313	PDIP-20	1	17	18	19	5	4	20	10
ATtiny26	PDIP-20	10	1	2	3	7	8	5, 15	6, 16
AT90S4433 ATmega8	PDIP-28	1	17	18	19	9	10	7, 20	8, 22
AT90S8515 ATmega161 ATmega162	PDIP-40	9	6	7	8	19	18	40	20
AT90S8535 ATmega8535 ATmega16 ATmega163 ATmega32 ATmega323	PDIP-40	9	6	7	8	13	12	10, 30	11, 31
ATmega64 ATmega103 ATmega128 ATmega169	TQFP-64	20	12	13	11	24	23	21, 52, 64	22, 53, 63

Примечание. Назначение выводов модификаций микроконтроллеров в корпусах других типов может не совпадать с указанным в таблице.

Его вилку XP1, как и в предыдущем случае, подключают к розетке порта LPT компьютера. Перемычки между контактами 3 и 11, 2 и 12 вилки позволяют управляющей программе "опознать" адаптер. Светодиод HL1 служит для

В отличие от микроконтроллеров серии PICmicro, рассматриваемые приборы требуют обязательной работы тактового генератора во время программирования. Все необходимые в этом режиме операции выполняются за определенное число тактов. Поэтому в непосредственной близости от микроконтроллера должен быть установлен и подключен к выводам XTAL1 и XTAL2 кварцевый резонатор ZQ1. Никакие длинные провода здесь недопустимы.

Резонансная частота ZQ1 — не менее 4 МГц, выдерживать ее с большой точностью нет необходимости. Обычно пригоден резонатор, с которым микроконтроллеру предстоит работать в дальнейшем. Однако при слишком "резвом" управляющем компьютере, формирующем сигналы недостаточной длительности, на время программирования придется установить более

высокочастотный резонатор, не превышая, конечно, допустимой для данного микроконтроллера частоты. Другой вариант — подать тактовые импульсы нужной частоты на вывод XTAL1 от внешнего генератора, оставив XTAL2 свободным.

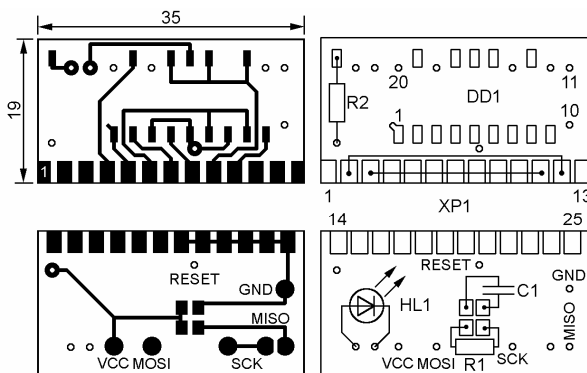


Рис. 9

Чертежи печатной платы рассматриваемого адаптера и схема

расположения на ней элементов показаны на рис. 9. Плата двусторонняя из фольгированного стеклотекстолита толщиной 1,5 мм.

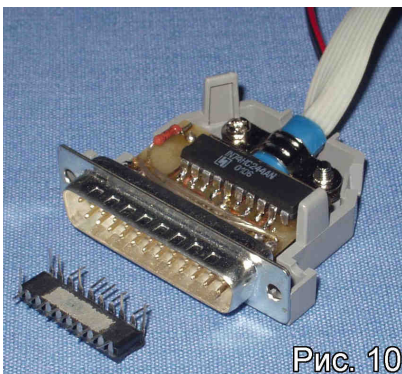


Рис. 10

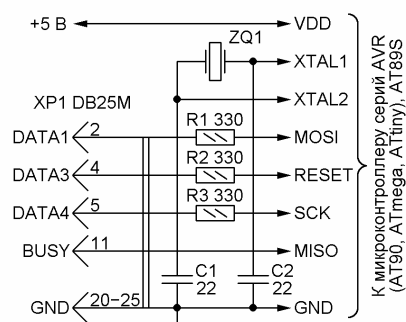
Диаметр контактных площадок с обратной стороны преднамеренно увеличен, что уменьшает риск "промазать", сверля отверстия с лицевой стороны. Размеры платы позволяют разместить ее внутри пластмассового корпуса вилки DB25M (XP1), как показано на фотографии рис. 10.

Плату вставляют ребром между рядами выводов вилки и припаивают каждый из них к соответствующей контактной площадке. Контакты 3 и 11, 2 и 12 соединяют перемычками из изолированного провода. Выводы микросхемы DD1 (кроме 10—12, 14, 18 и 20) перед монтажом

отгибают. Оставленные в исходном состоянии вставляют в отверстия и припаивают к контактным площадкам с обратной стороны платы, отогнутые — к прямоугольным площадкам со стороны установки микросхемы. Конденсатор С1 и резистор R1 — для поверхностного монтажа, хотя при необходимости можно применить и обычные малогабаритные. Светодиод, установленный со стороны, противоположной микросхеме, должен быть виден через специально просверленное в корпусе разъема отверстие.

Адаптер для программирования микроконтроллеров семейства AVR может быть упрощен путем исключения из него буферной микросхемы (DD1 на рис. 8) и соединения перемычками контактных площадок входов ее элементов с соответствующими выходными (2 с 18, 3 с 17, 4 с 16 и т. д.). Естественно, вероятность сбоев при программировании при этом возрастет.

Схема еще одного предельно упрощенного варианта адаптера показана на **рис. 11**. Она аналогична использованной в отладочной плате для



микроконтроллера AT90S2313, выпускаемой фирмой Dontronics под названием DT-006.

Среди микроконтроллеров фирмы Atmel имеются относящиеся по структуре и системе команд к все еще популярному семейству MCS-51 и сравнительно дешевые AT89C1051, AT89C2051 и AT89C4051. Их широкому распространению среди

Рис. 11

радиолюбителей препятствует одно — сравнительная сложность адаптера программирования. Дело в том, что интерфейс программирования этих микроконтроллеров — своего рода переходная ступень между параллельным и последовательным. Число задействованных в программировании выводов вынуждено ограничено, так как не может превышать общего числа выводов микросхемы, но все-таки остается значительно большим, чем при истинно последовательным интерфейсе.

Стандартный, рекомендуемый фирмой-изготовителем адаптер для этих микроконтроллеров построен на семи микросхемах средней степени интеграции. Его схему и описание (документ AN285) можно найти на Интернет-сайте фирмы по адресу http://www.atmel.com/dyn/resources/prod_documents/DOC0285.PDF или в переведенном на отечественную элементную базу виде в [5].

Более простой адаптер предложен немецким радиолюбителем DL2TM, его схема показана на **рис. 12**. Розетка XS1 адаптера аналогична устанавливаемой

на принтерах и соединяют ее с LPT-портом компьютера стандартным "принтерным" кабелем. Программируемую микросхему вставляют в панель XS2. Команды и данные для программирования поступают от компьютера двумя потоками на входы регистров-преобразователей последовательного кода в параллельный DD1 и DD2. Коммутатор DD3 поочередно подает на входные линии

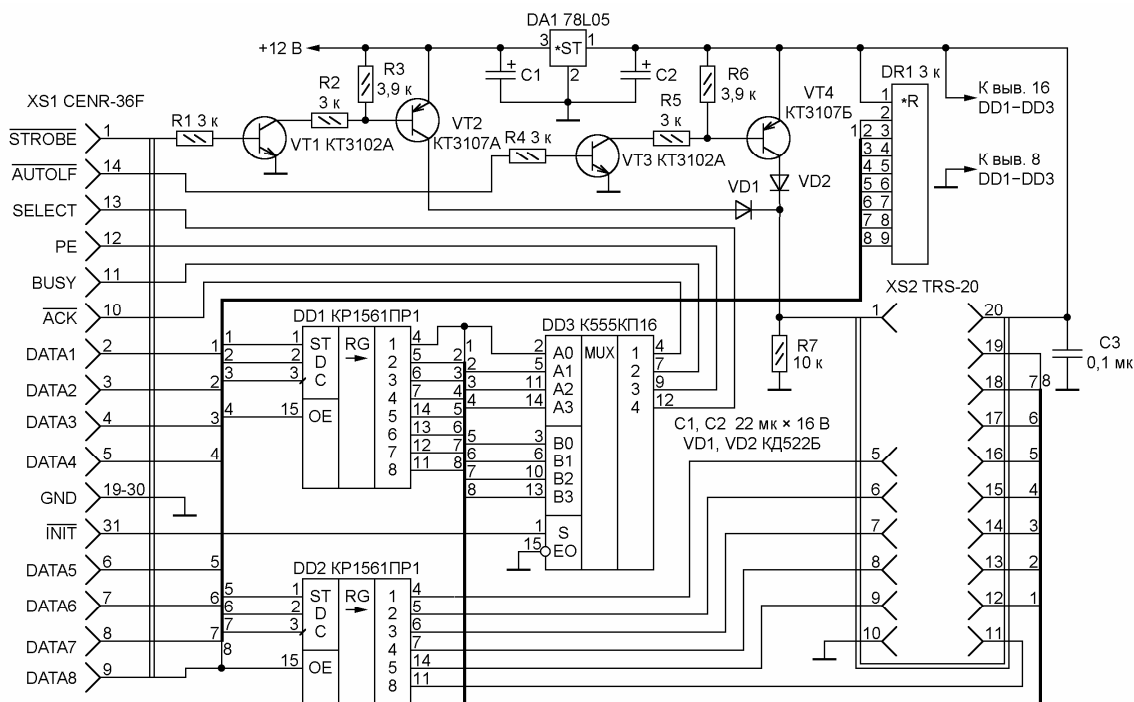


Рис. 12

порта старшую и младшую четверку разрядов ответных сообщений микроконтроллера.

Адаптер питают напряжением +12 В от внешнего источника. С помощью стабилизатора DA1 получают напряжение +5 В для питания микросхем, а ключи на транзисторах VT1—VT4 управляют уровнем напряжения на входе микроконтроллера, переводящем его в режим программирования.

Чертеж печатной платы этого адаптера приведен на **рис. 13**. Перемычки на стороне расположения компонентов навесные (из отрезков луженого провода). Если имеется возможность изготовить двустороннюю плату, их можно сделать печатными.

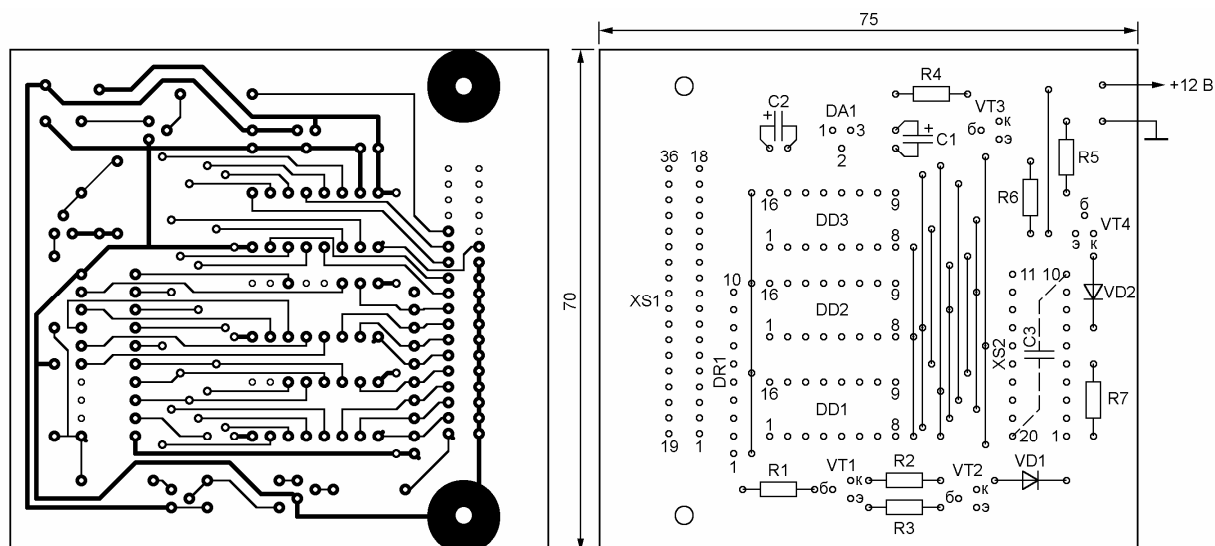


Рис. 13

"Радио", 2004, № 4, с. 51, 52

Адаптер готов, что дальше?

После того, как адаптер для подключения к компьютеру программируемой микросхемы готов, желательно его проверить. Если он рассчитан на подключение к COM-порту компьютера, воспользуйтесь программой TCOM, которая находится на FTP-сервере журнала "Радио" по адресу <ftp://ftp.radio.ru/pub/2003/05/tcom/tcom.exe>, о порядке ее использования для проверки одного из вариантов адаптера SI Prog можно прочитать в [4].

Другие адаптеры проверяют аналогичным образом. Нужно лишь, как сказал герой кинофильма "С легким паром", "мыслить логически" и, проанализировав схему адаптера, представить себе, какие значения должно принимать напряжение на том или ином гнезде панели программируемой микросхемы при том или ином состоянии линий порта.

Теперь остается найти подходящую "программу программирования", которая, будучи запущенной на компьютере, возьмет на себя управление адаптером и формирование всех нужных сигналов на выводах программируемой микросхемы.

Подобные программы бывают двух видов. Простейшие лишь реализуют алгоритм, нужный для загрузки кода в микроконтроллер одного или нескольких очень похожих типов. Обычно бывает достаточно, запуская такую программу или отвечая на ее запрос, указать имя файла с исходными данными. Все остальное будет сделано автоматически. Пользователю не предоставлено никаких

дополнительных возможностей, иногда невозможно даже прочитать записанное и сверить его с исходными данными.

Как правило, эти программы написаны их авторами лишь для того, чтобы запрограммировать недавно выпущенную (или ту, с которой пришлось столкнуться впервые) микросхему. Для опытного программиста это иногда проще, чем искать готовую программу. Однако посторонний пользователь подобных программ рискует столкнуться с проблемами и дефектами, которые автор программы в свое время не обнаружил и не исправил, а сделать это без его участия практически невозможно.

Нужно сказать, что фирмы—изготовители микроконтроллеров тоже распространяют (в том числе на справочных компакт-дисках и на своих Интернет-сайтах) простые программы для их программирования, либо включают их в состав систем автоматизированной разработки программного обеспечения. Позволяя потенциальным потребителям быстрее освоить новый товар, это способствует увеличению спроса. Достоинство "фирменных" программ — строгое соблюдение всех предписанных режимов программирования и быстрое обновление при выявлении каких-либо дефектов в старых версиях. Недостаток — они работают, как правило, только с фирменным же (или рекомендованным фирмой) адаптером, который предлагается купить и весьма недешево. Справедливости ради нужно отметить, что многие фирмы (Atmel и MicroChip в том числе) публикуют схемы своих адаптеров, не делая из них секрета. Именно к таким относятся некоторые из тех, о которых шла речь в предыдущих разделах.

Другая категория "программ программирования" — универсальные, способные работать с адаптерами различных типов и программировать большое число разнообразных микросхем памяти и микроконтроллеров. Универсальные программы снабжены, как правило, развитым сервисом и очень удобны для пользователя. Очень хорошо, если автор программы "поддерживает" ее, постоянно устраняя не замеченные ранее ошибки и расширяя список программируемых микросхем и пригодных для них адаптеров.

Из универсальных программ, распространяемых через Интернет бесплатно, наиболее популярны PonyProg и IC-PROG. Сравнить их довольно сложно — каждая имеет свои особенности, которые кто-то сочтет достоинствами, а кто-то — недостатками. Поэтому перейдем к рассмотрению конкретных программ и способов работы с ними.

Устанавливаем PonyProg на компьютер

Набрав в Интернет-браузере адрес <<http://www.lancos.com/ppwin95.html>>, увидим на экране картину, подобную показанной на **рис. 14**, с длинным списком версий программы PonyProg. Возможно, к моменту публикации данной статьи список станет еще длиннее. В самой нижней его части — последняя англоязычная версия программы (2.06с), а выше перечень исполняемых файлов предыдущей версии (2.05а), адаптированных на разные языки, в том числе на русский.

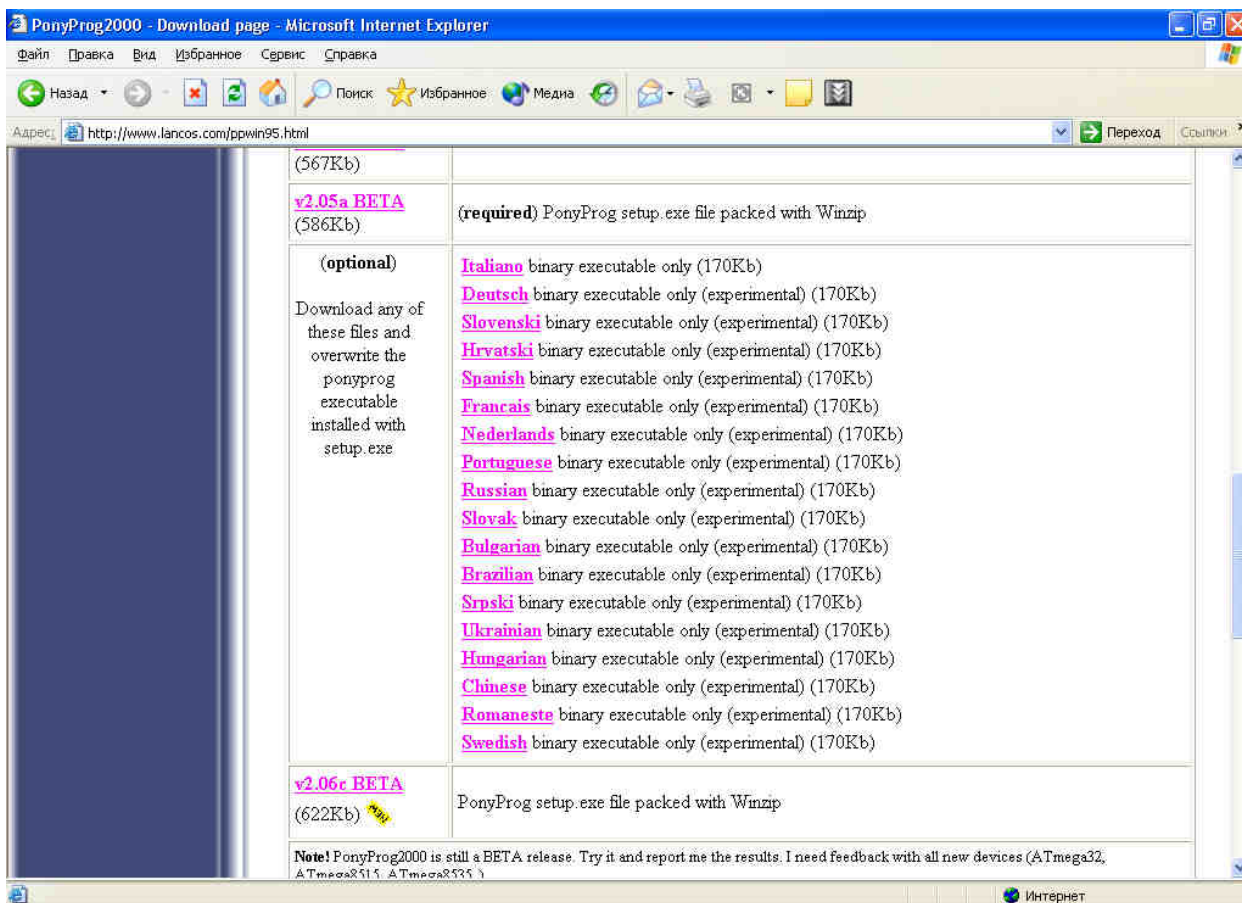


Рис. 14

Что же "скачать"? Прежде всего, "щелкнув" по надписи "v2.06c BETA", оригинальный англоязычный вариант. В полученном файле—архиве **ponyprogV206c.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 скачать (щелкнув по слову Russian, см. рис. 14) архив **PonyProg2000-Russian.zip** и находящимся в нем файлом **PONYPROG2000.exe** заменить одноименный в папке **C:\Program Files\PonyProg2000** (она была создана программой-установщиком на предыдущем этапе. Можно поместить русифицированный файл

в отдельную папку, что 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 2.05a 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. Однако те, кто предпочитает операционную систему LINUX, найдут на той же Интернет-странице и предназначенные для нее варианты.

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 получают 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.

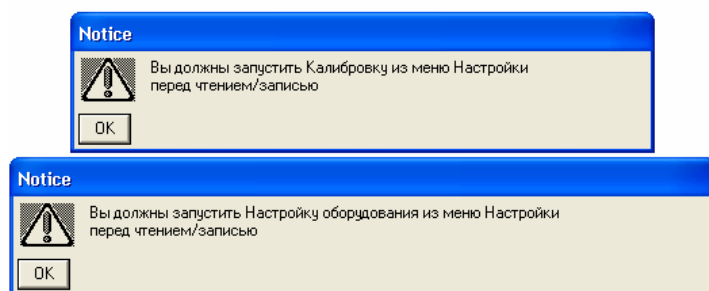


Рис. 15

Закройте его, нажав кнопку "OK".

Тут же одно за другим появятся два предупреждения (рис. 15), которым мы последуем чуть позже, а пока просто закроем их, нажимая "OK". Учтите, что в русском тексте предупреждений

меню "Установки" ошибочно названо "Настройки". Аналогичная ошибка имеется и в английском варианте. Меню "Setup" названо "Options".

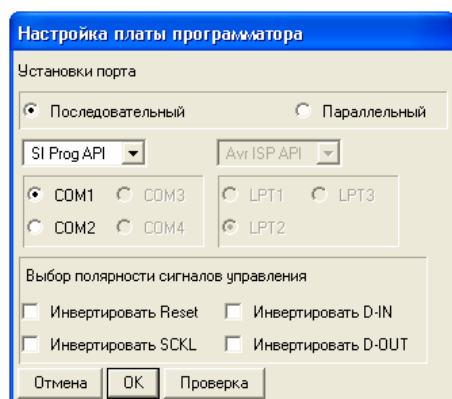


Рис. 16

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”, показанное на **рис. 16**, 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:

SI Prog API
SI Prog I/O
JDM API

Как уже было сказано, SI Prog — оригинальный набор адаптеров, разработанных автором программы PonyProg специально для нее. О них можно прочитать в [6]. Схема адаптера JDM приведена на рис. 3 настоящего цикла. 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 порту¹. 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” (в нижней части рис. 16) 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

¹ К сожалению, эта проверка работает только с оригинальными адаптерами набора SI-Prog.

некоторых сигналов, учитывая таким образом особенности адаптера или микросхемы.

Закончив настройку, нажмите кнопку "ОК".

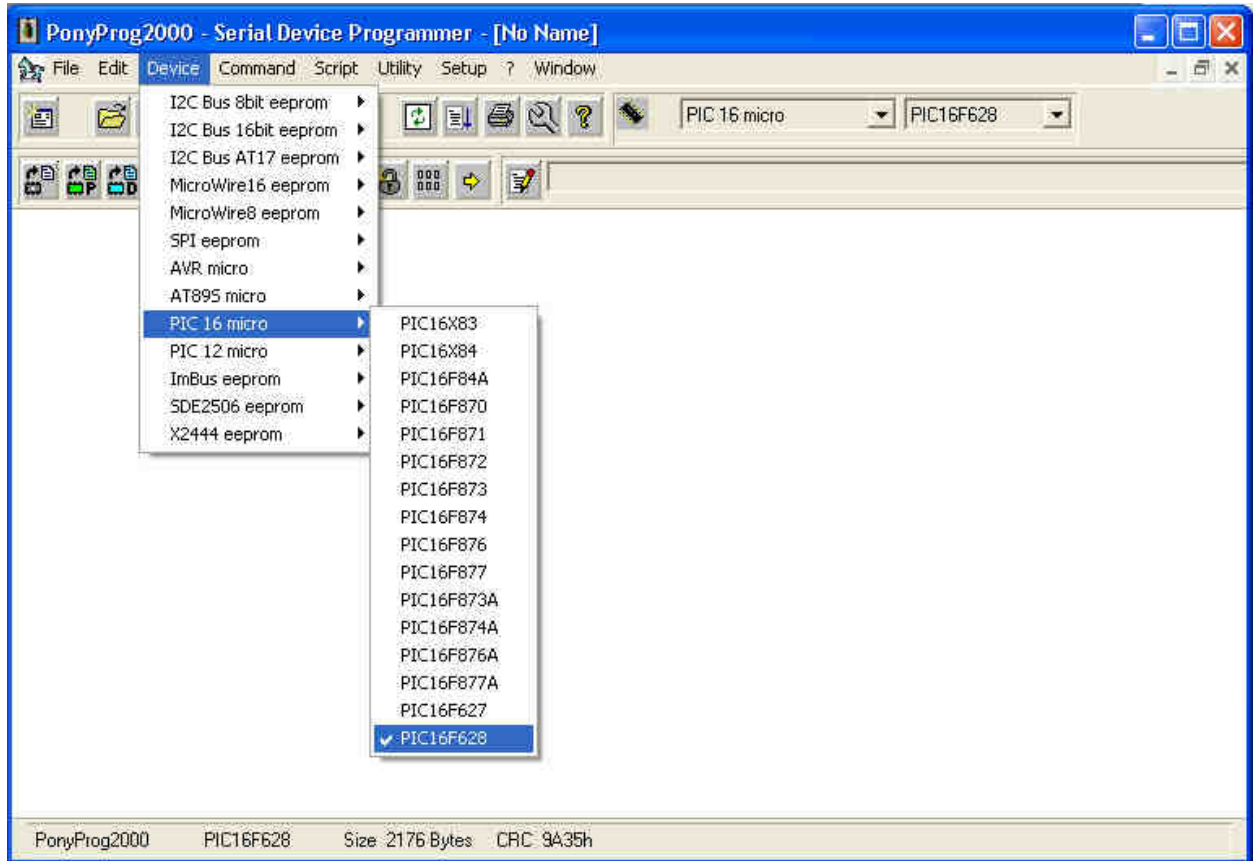
Далее выполним пункт "Калибровка" меню "Установки". Программа "измерит" скорость работы компьютера и вычислит значения переменных, определяющих в дальнейшем длительность импульсов и пауз между ними, формируемых в процессе программирования. Учтите, что как при калибровке, так и при собственно программировании 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.

"Радио", 2004, № 5, с. 51, 52

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. 17**, показано окно версии 2.06с). 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-ax) 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 Интернета² 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

² **Примечание редакции.** Такие файлы для всех публикуемых в журнале "Радио" конструкций на микроконтроллерах редакция размещает в Интернете на своем FTP-сервере по адресу <ftp://ftp.radio.ru/pub/ГГГГ/ММ/>, где ГГГГ — год издания (четырёхзначное число), ММ — номер журнала (двузначное число, например, 02). — Прим. ред.

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, подобное изображенному на **рис. 18**.

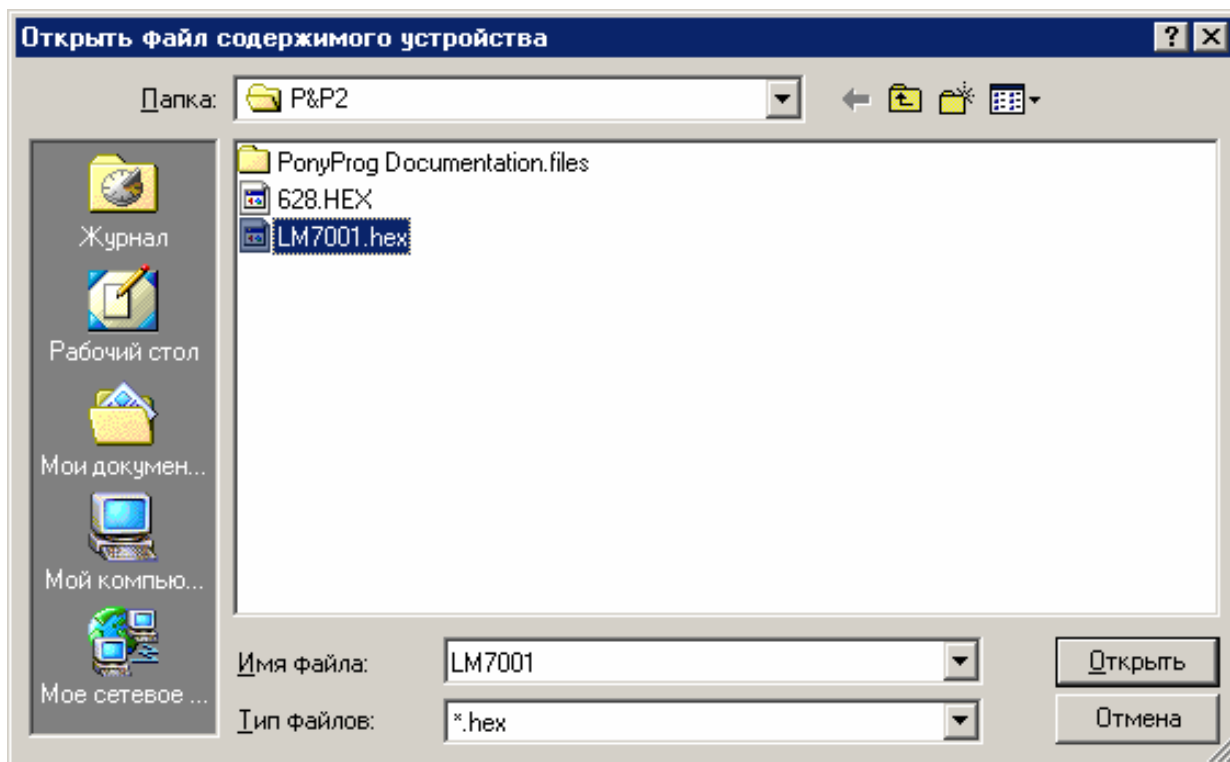
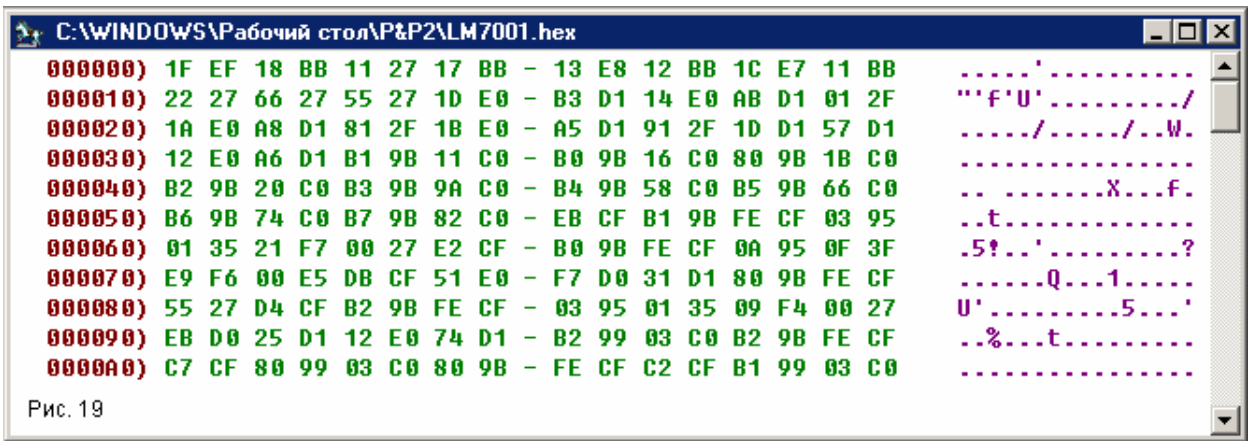


Рис. 18

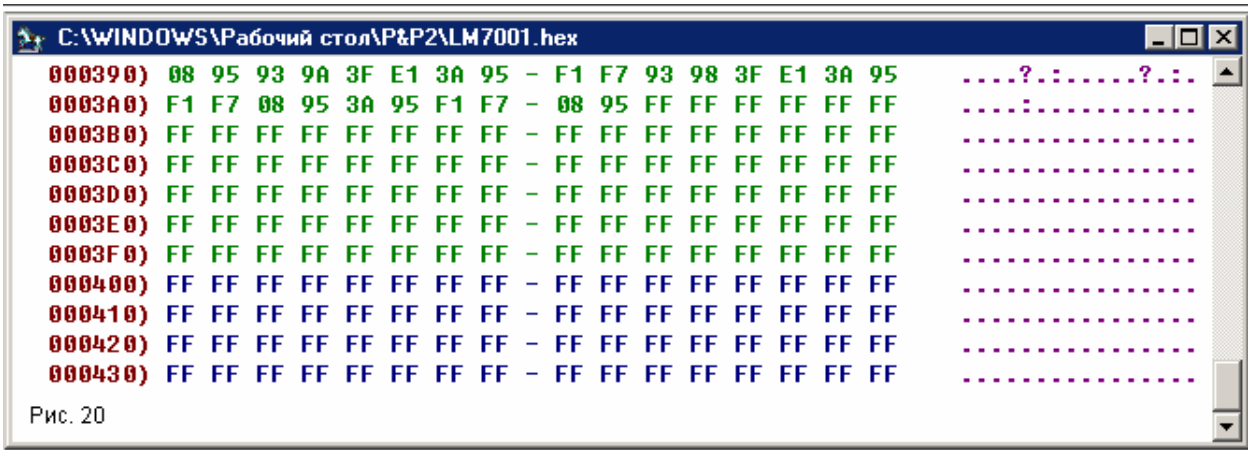
В списке будут содержаться лишь те файлы, имена которых имеют расширение, указанное в окне "Тип файлов". Нажав в нем кнопку со стрелкой вниз, можно перейти к файлам с другими расширениями или (выбрав "*") получить список всех файлов, имеющихся в папке. Файл будет загружен после двойного щелчка по его 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 (**рис. 19**).

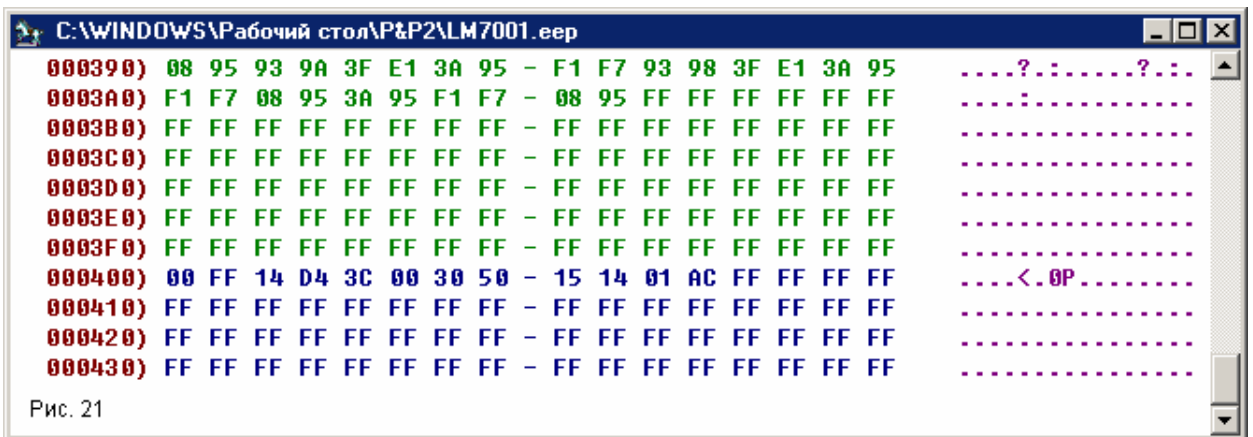


Она состоит из строк, начинающихся с отделенного скобкой адреса первого байта строки. Затем следуют шестнадцатиричные значения 16-ти байтов с последовательно возрастающими адресами (для удобства они разделены тире на две группы по восемь) и далее — символьное представление тех же байтов.

Обратите внимание на нижнюю часть таблицы (рис. 20). 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)..." (рис. 21). 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.

"Радио", 2001, № 6, с. 53, 54

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 шестнадцатичный 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. 2.06c выводит их правильно, в полном соответствии с "кодовой страницей 1251", как показано на **рис. 22**.

```

C:\WINDOWS\Рабочий стол\ASCII.hex
000000) 00 01 02 03 04 05 06 07 - 08 09 0A 0B 0C 0D 0E 0F .....Ъ.....
000010) 10 11 12 13 14 15 16 17 - 18 19 1A 1B 1C 1D 1E 1F .....
000020) 20 21 22 23 24 25 26 27 - 28 29 2A 2B 2C 2D 2E 2F  !"#$%&'()*+,-./
000030) 30 31 32 33 34 35 36 36 - 38 39 3A 3B 3C 3D 3E 3F 0123456689:;<=>?
000040) 40 41 42 43 44 45 46 47 - 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
000050) 50 51 52 53 54 55 56 57 - 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[\]^_
000060) 60 61 62 63 64 65 66 67 - 68 69 6A 6B 6C 6D 6E 6F `abcdefghijklmnopqrstuvwxyz
000070) 70 71 72 73 74 75 76 77 - 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{|}~.
000080) 80 81 82 83 84 85 86 87 - 88 89 8A 8B 8C 8D 8E 8F ЪГ,ф,г,д,е,ж,з,и,к,л,м,н,о,п,р,с,т,у,ф,х,ц,ч,ш,щ,ъ,ы,э,я
000090) 90 91 92 93 94 95 96 97 - 98 99 9A 9B 9C 9D 9E 9F ъ`"#$%&'()*+,-./:;<=>?@ABCDEFGHIJKLMNO
0000A0) A0 A1 A2 A3 A4 A5 A6 A7 - A8 A9 AA AB AC AD AE AF ъг,ф,г,д,е,ж,з,и,к,л,м,н,о,п,р,с,т,у,ф,х,ц,ч,ш,щ,ъ,ы,э,я
0000B0) B0 B1 B2 B3 B4 B5 B6 B7 - B8 B9 BA BB BC BD BE BF °±ііггµ¶·ёёё»»јѕѕі
0000C0) C0 C1 C2 C3 C4 C5 C6 C7 - C8 C9 CA CB CC CD CE CF АБВГДЕЖЗИЙКЛМНОП
0000D0) D0 D1 D2 D3 D4 D5 D6 D7 - D8 D9 DA DB DC DD DE DF РСТУФХЦШЩЬЫЭЮЯ
0000E0) E0 E1 E2 E3 E4 E5 E6 E7 - E8 E9 EA EB EC ED EE EF абвгдежзийклмноп
0000F0) F0 F1 F2 F3 F4 F5 F6 F7 - F8 F9 FA FB FC FD FE FF рстуфхцшщщьыьэя

```

Рис. 22

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, находящейся на FTP-сервере журнала "Радио" по адресу <ftp://ftp.radio.ru/pub/2003/08/check/chkhex.exe>. Достоинство этой программы в том, что она следит за правильностью ввода кодов и сообщает об ошибках. Но можно ввести данные вручную и непосредственно в буфер программатора.

Запустите PonyProg и выполните все, что необходимо для его настройки на программирование микроконтроллера нужного типа. Далее вместо загрузки файла выберите в главном меню пункт "Утилиты", а в открывшемся подменю — "Очистить буфер". Эту же операцию можно выполнить, нажав комбинацию клавиш "Ctrl-C".

На экране появится окно default. Буферы FLASH-памяти и EEPROM заполнены байтами 0FFH. Именно в таком состоянии находятся ячейки памяти

микроконтроллера до записи в них информации или после ее стирания. Выберите пункт главного меню "Буфер", а в нем — "Редактирование буфера".

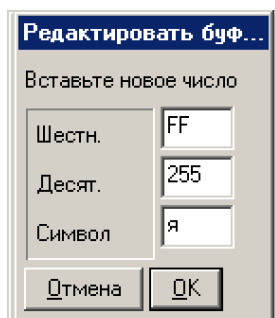


Рис. 23

В "шестнадцатиричной" части окна у ячейки с нулевым адресом появится мигающий курсор. Если теперь нажать левую кнопку мыши или клавишу Enter, будет открыто окно редактирования содержимого ячейки, показанное на **рис. 23**.

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 ОК 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, изображенное на **рис. 24**, 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

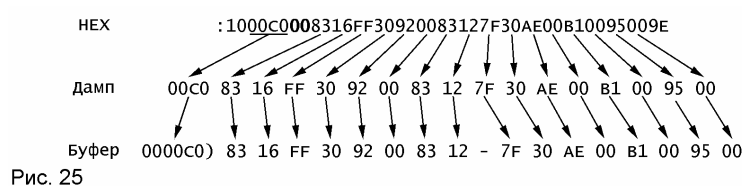


Рис. 24

ОК, 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 PonyProg, 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. Соответствие между ними и буфером иллюстрирует **рис. 25**.



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 (на рис. 25 подчеркнут) 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 с 4200H, 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. 26**.

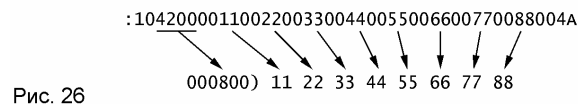


Рис. 26

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, 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, как показано на **рис. 27**.

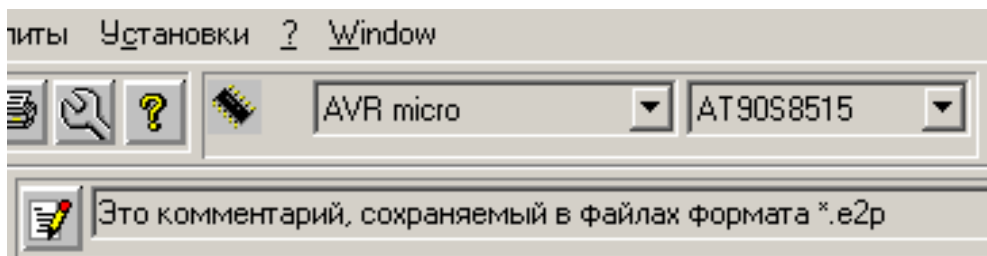


Рис. 27

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

"Радио", 2004, № 7, с. 53, 54

Не забудьте 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. Показанное на **рис. 28** относится к PIC16F628.

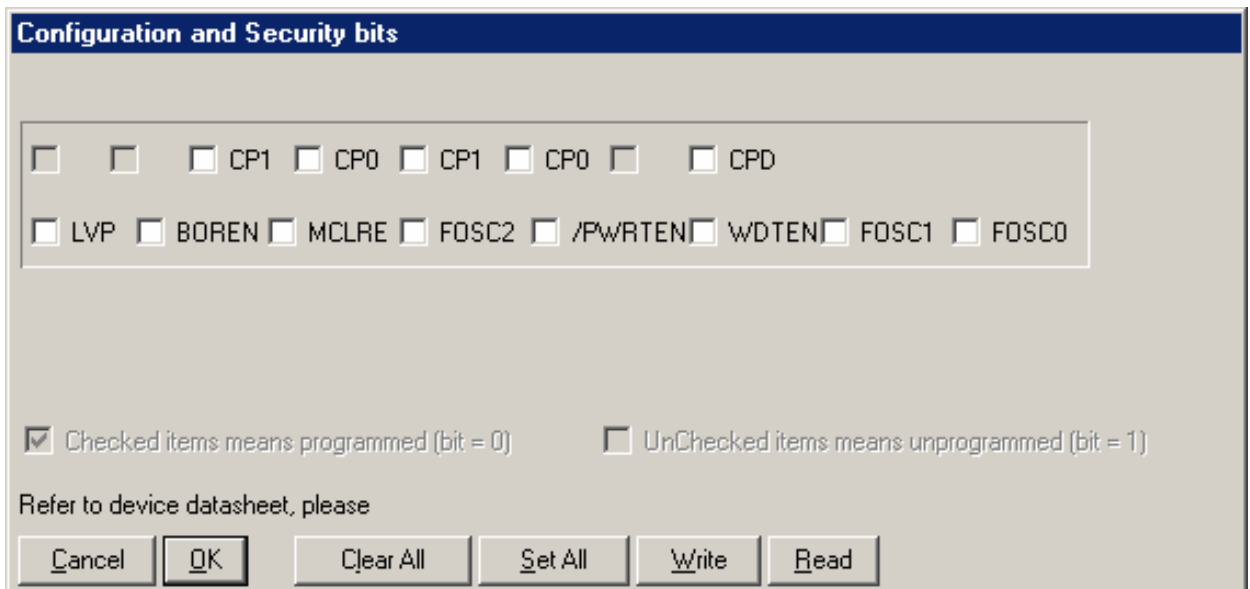


Рис. 28

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 "дубли" 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.

MCLR — вывод 4 микросхемы служит входом сигнала установки микроконтроллера в исходное состояние MCLR (1) или обычным цифровым входом RA5 (0).

/PWRTE — 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.

FOSC2—FOSC0 — 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 удастся 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", **рис. 29**). 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

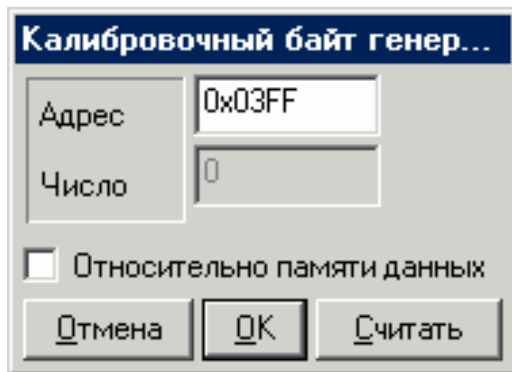


Рис. 29

отлаженной. Надеюсь, в новых версиях PonyProg недочеты будут устранены.

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.

Хотелось бы обратиться к авторам конструкций на микроконтроллерах. Описывая их, не забывайте рассказывать об особенностях программирования конфигурации, а если необходимо, и установки частоты внутреннего генератора.

Осторожно, утилиты!

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. Пример показан на **рис. 30**: а — до; б — после исполнения данного пункта. 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
  
```

Рис. 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. 3, 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.

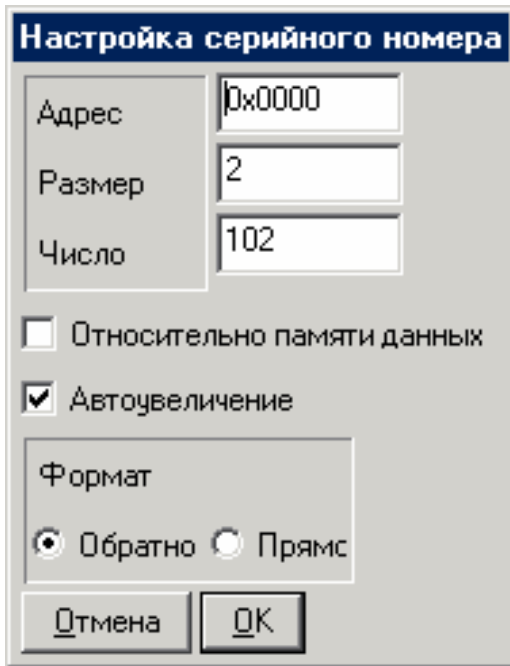


Рис. 31

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. 31**. 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 "Прямс" 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. 32**, текущее значение CRC выведено в нижней части (строке состояния) главного окна PonyProg вместе с выбранным типом

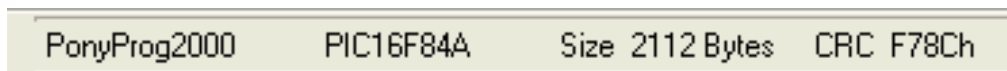


Рис 32

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 всем ячейкам буфера, либо повторить операции по подготовке данных заново.

Программируем, наконец

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 содержащейся в ней ранее информации, необходимо выбрать в меню "Команды" пункт "Стереть" и получить сообщение,

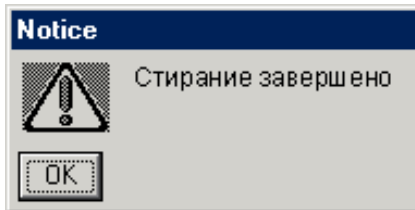


Рис. 33

показанное на **рис. 33**. 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 от того, какая область памяти представляет интерес. На экране появится информация о ходе считывания (**рис. 34**), а затем — о его завершении (**рис. 35**).

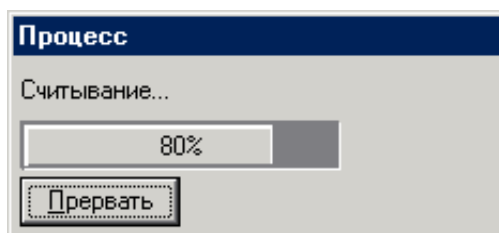


Рис. 34

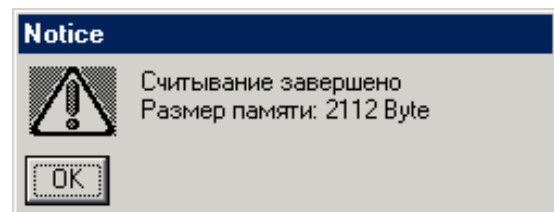


Рис. 35

Теперь можно, вернувшись в меню "Файл", сохранить информацию, как это делалось при ручной подготовке данных (см. "Радио", 2004, № 6, с. 54). Там в тексте (колонка 3, строка 27 снизу) вкралась опечатка. Следует читать: "... будет сохранен весь буфер либо только FLASH, либо только EEPROM". После этого память микросхемы очищают командой "Стереть", а ненужное более окно закрывают.

осле 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. 36**) — немного запоздалое, так как все "предыдущее содержимое" уже стерто.

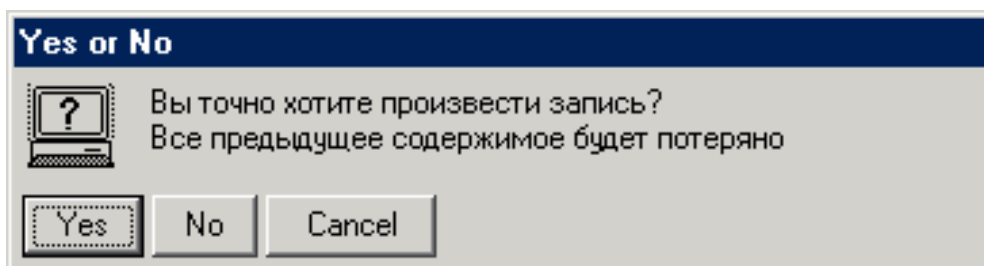


Рис. 36

Запись начнется после нажатия на кнопку "Yes". О ее ходе сообщит окно,

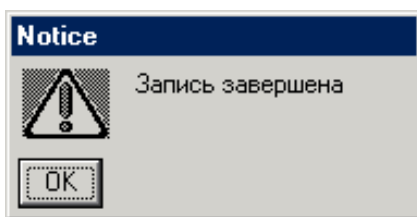


Рис. 37

подобное показанному на рис. 34, 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. 37**). 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

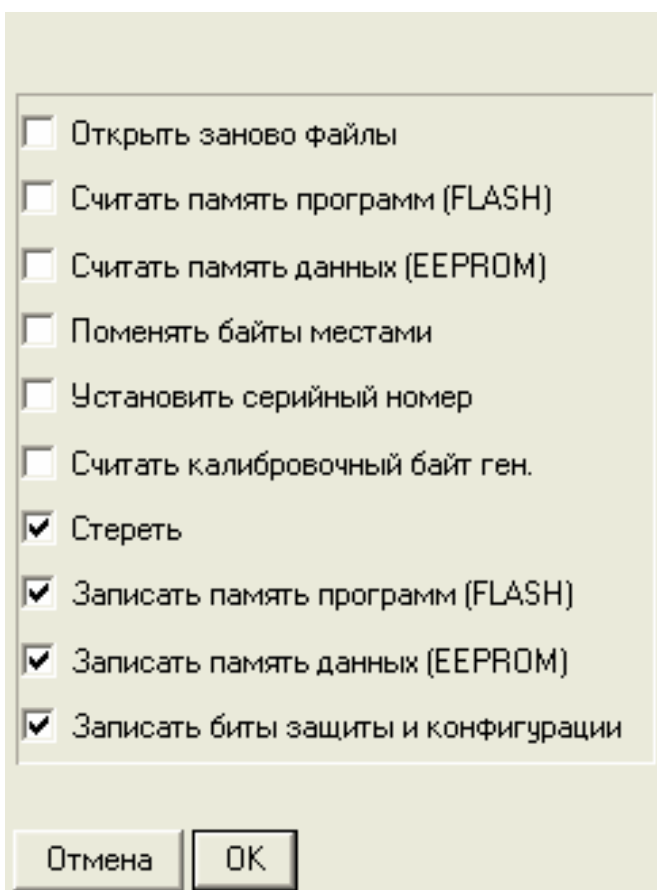


Рис. 38

ячейке этой памяти — 0x3FFF. В буфере PonyProg под этот код отведено 16 разрядов (два байта), значение кода в которых после очистки буфера — 0xFFFF. 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. 38**. Теперь при каждом выборе пункт а "Программирование" отмеченные команды будут выполнены автоматически в той последовательности, в которой они перечислены в окне.

На этом описание оболочки программирования PonyProg закончено. Следующие разделы будут посвящены описанию особенностей программы аналогичного назначения IC-Prog.

Устанавливаем IC-Prog

"Радио", 2004, № 9, с. 51, 52

Оболочка программирования IC-Prog занимает одно из первых мест по популярности среди радиолюбителей, так как способна работать с большим числом как программируемых микросхем, так и адаптеров различных типов. Интересы автора программы (его зовут Bonny Gijzen) лежат, по-видимому, в области микроконтроллеров PICmicro, поэтому приспособлена IC-Prog больше к программированию именно этих микроконтроллеров, в отличие от PonyProg, развивающейся в сторону микроконтроллеров фирмы Atmel. Хотя с программированием и тех и других обе программы успешно справляются.

"Скачать" IC-Prog можно с сайта его автора по адресу <<http://www.ic-prog.com/icprog105C.zip>>. В этом архиве находится единственный файл icprog.exe последней на момент подготовки статьи версии 1.05с. С выходом новых версий адрес может измениться. Если на компьютере установлена операционная система Windows-98 или Windows Millenium, файл достаточно запустить на исполнение.

Для операционных систем Windows 2000, Windows XP потребуется еще один файл, icprog.sys, находящийся в архиве по адресу <<http://www.ic->

prog.com/icprog_driver.zip>. Файл драйвера необходимо поместить в одну папку с исполняемым `icprog.exe`. Перед запуском IC-Prog в этих ОС необходимо, щелкнув по имени файла правой кнопкой мыши, выбрать пункт "Свойства" ("Properties") и установить совместимость (compatibility) с Windows 2000. Для владеющих английским языком будет полезен файл помощи, находящийся по адресу <<http://www.ic-prog.com/icprog.chm>>.

После запуска программы на экране появится окно, похожее на изображенное на **рис. 39**, но с надписями на английском языке. Выберем в меню "Settings" (настройки) пункт "Options" (опции), а в нем — закладку "Language" (язык), где в свою очередь выберем Russian, как показано на **рис. 40**. После выполнения предложенного программой перезапуска она станет русскоязычной. Некоторые несуразности вроде слова "команды" с удвоенной буквой м, оставим на совести переводчика.

При первом запуске IC-Prog в среде Windows 2000/XP необходимо в том же меню "Settings—Options" открыть закладку "Misc" (разное) и отметить пункт "NT/2000 Driver".

Первое бросающееся в глаза отличие IC-Prog от PonyProg — отдельные окна буферов программной памяти и памяти данных микроконтроллера. К тому же буфер программы — 16-разрядный. Коды команд длиной 12 (как у PIC12C508) 14 (как у PIC16F84) или 16 (как у микроконтроллеров серии PIC18) двоичных разрядов отображаются в нем четырехразрядными шестнадцатиричными числами. Для сравнения на **рис. 41** приведены строки буферов PonyProg (а) и IC_Prog (б), содержащие один и тот же фрагмент программы микроконтроллера PIC16F84.

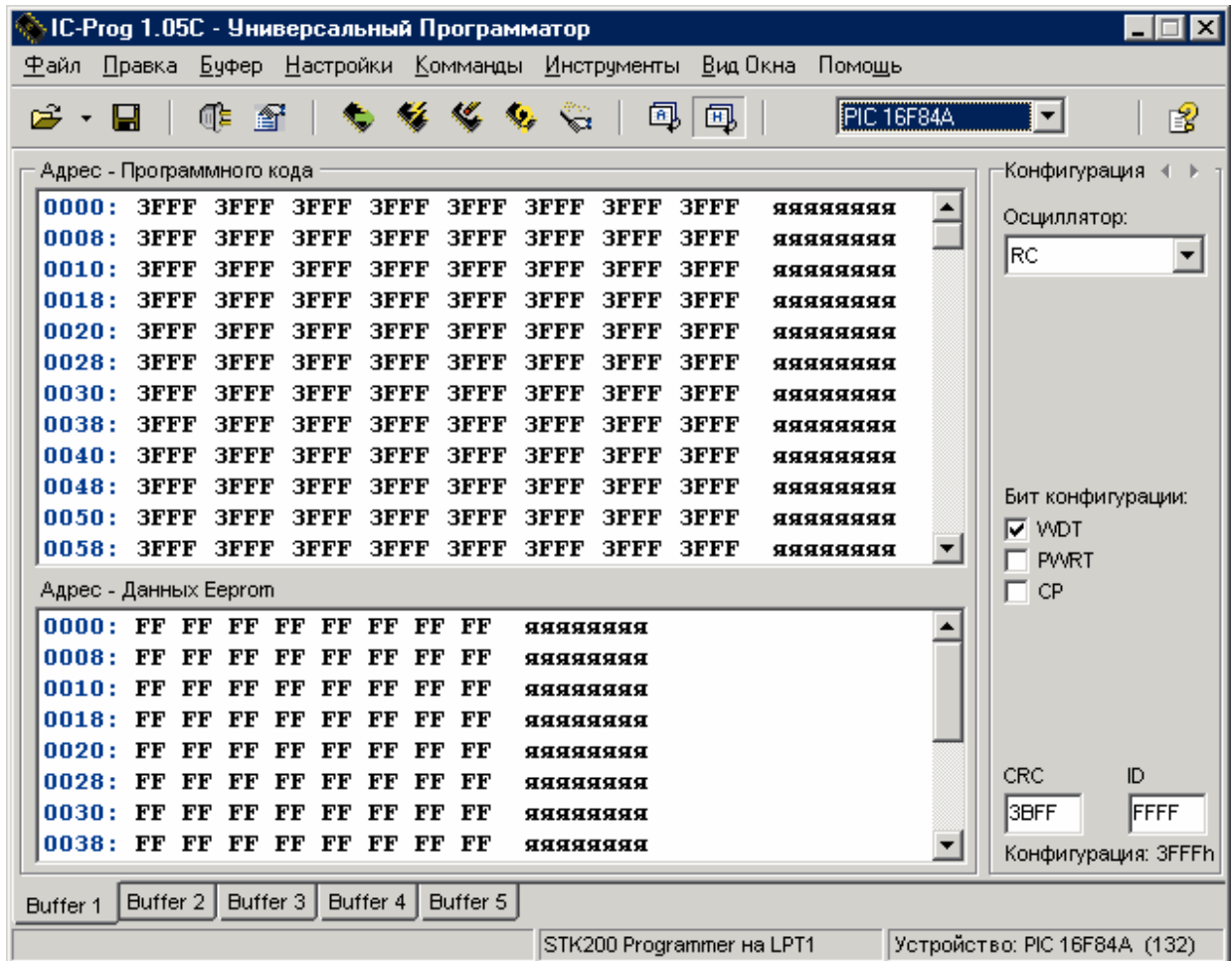


Рис. 39

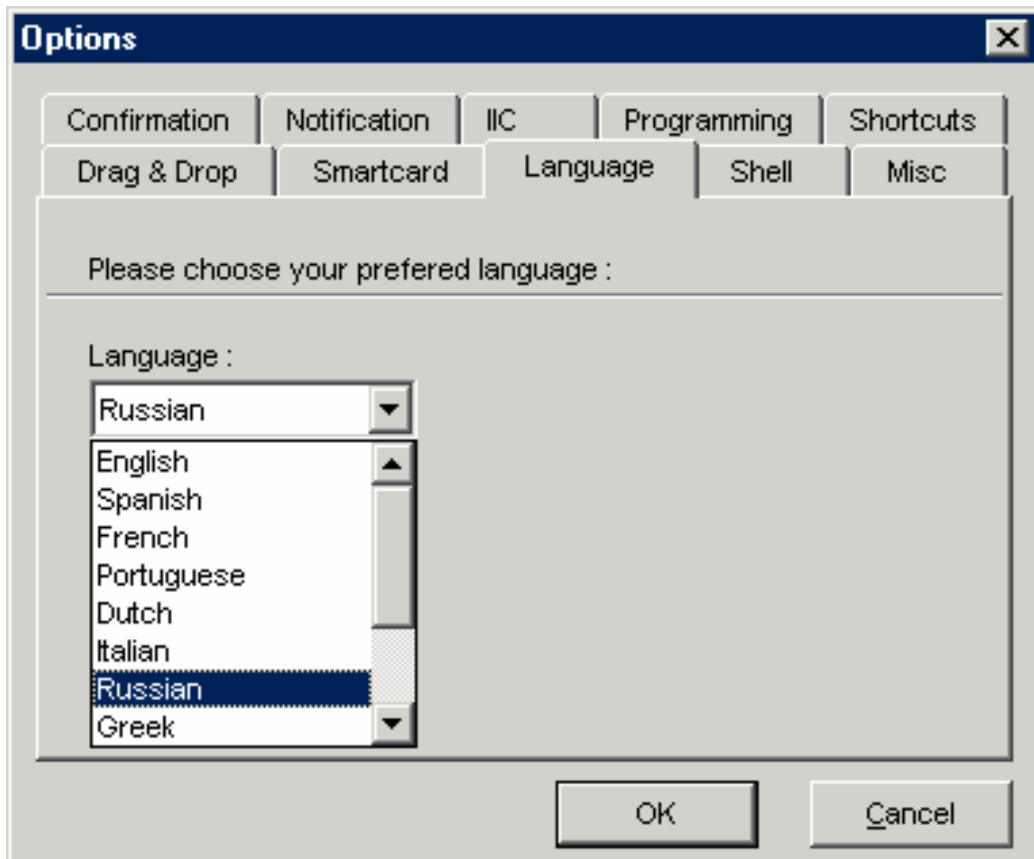


Рис. 40


```

000000) 05 28 FF FF FF FF FF FF - 05 28 85 01 86 01 07 30
000010) 9F 00 83 16 01 30 85 00 - 00 30 86 00 8E 11 83 12

```

а)

```

0000: 2805 3FFF 3FFF 3FFF 2805 0185 0186 3007
0008: 009F 1683 3001 0085 3000 0086 118E 1283

```

б)

Рис. 41

Отличия заметить несложно. Адреса ячеек буфера IC-Prog соответствуют действительным адресам 14-разрядных ячеек памяти микроконтроллера (а не больше их в два раза, как в PonyProg). Порядок следования шестнадцатиричных цифр соответствует старшинству разрядов, а не байтов. Содержимое "пустых" ячеек (3FFFFH) отображается правильно.

При байтовой, как у микроконтроллеров серии AT90, организации памяти в каждом слове буфера заполняется только младший байт. Старший становится нулевым. Пример для микроконтроллера AT90S1200 приведен на **рис. 42**. Как и на предыдущем рисунке а) — буфер PonyProg, б) — буфер IC-prog.

```

000000) 1F EF 18 BB 11 27 17 BB - 13 E8 12 BB 1C E7 11 BB
000010) 22 27 66 27 55 27 1D E0 - B3 D1 14 E0 AB D1 01 2F

```

а)



```

0000: 001F 00EF 0018 00BB 0011 0027 0017 00BB
0008: 0013 00E8 0012 00BB 001C 00E7 0011 00BB
0010: 0022 0027 0066 0027 0055 0027 001D 00E0
0018: 00B3 00D1 0014 00E0 00AB 00D1 0001 002F

```

б)

Рис. 42

Интересная особенность буфера памяти программ IC-Prog — возможность дизассемблировать находящиеся в нем коды и увидеть их в форме мнемоник команд микроконтроллера. Достаточно нажать на экранную кнопку  или выбрать в меню "Вид Окна" пункт "Ассемблерный", чтобы окно превратилось в показанное на **рис. 43**. Его содержимое соответствует шестнадцатиричному на рис. 41, а. Возвращают окно в прежнее состояние с помощью экранной кнопки  или пункта меню "Вид Окна — Шестнадцатиричный". К сожалению, дизассемблер "знаком" только с системой команд микроконтроллеров серии PIC16. К ним можно

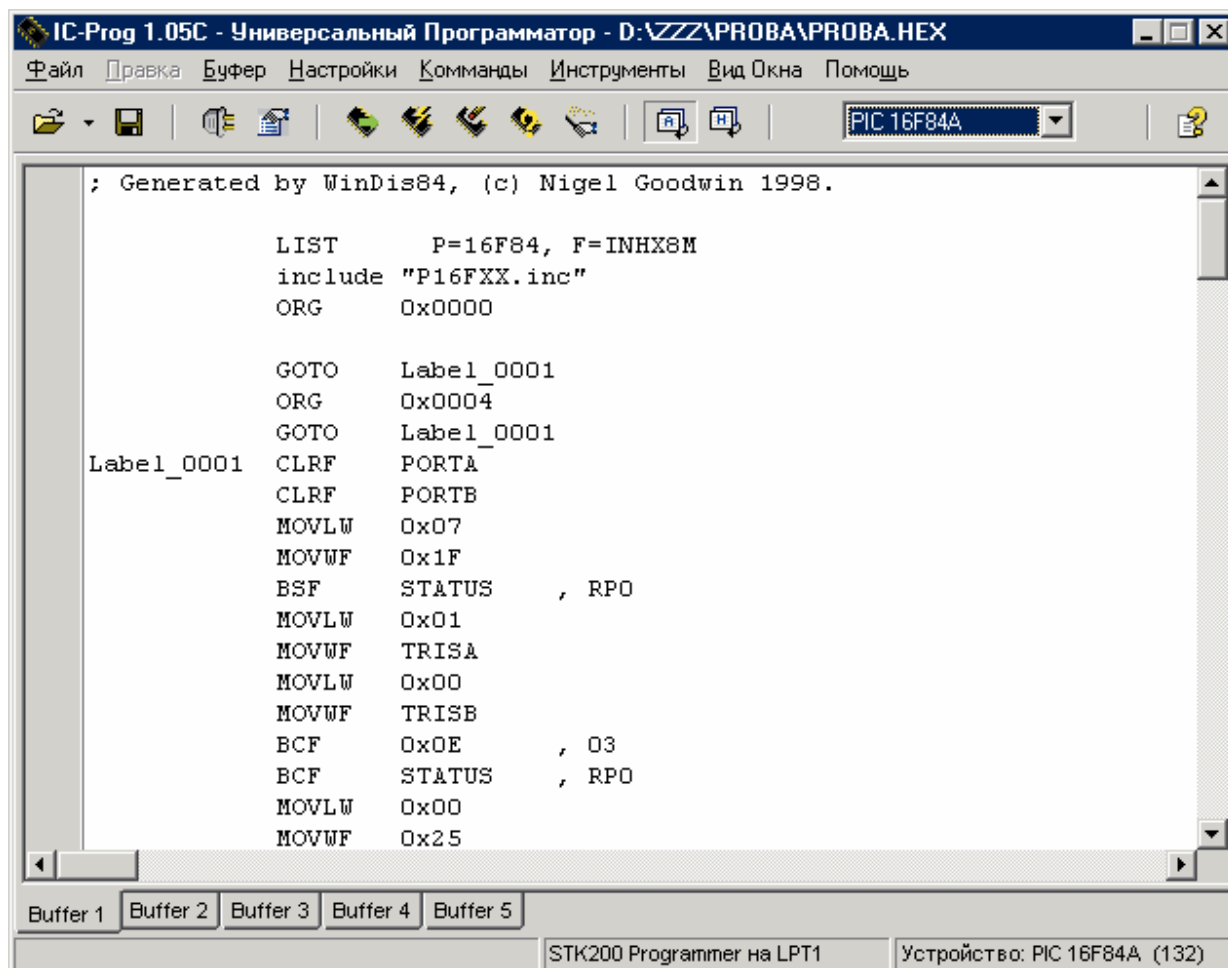


Рис. 43

добавить PIC12, если поместить в папку с файлом icprog.exe файл динамической библиотеки подпрограмм disasm.dll, "скачанный" по адресу <<http://www.ic-prog.com/disasm.dll>>.

Буфер памяти данных (EEPROM) у программы IC-Prog байтовый. "Лишних", не используемых при программировании байтов между используемыми в нем нет. И адреса ячеек буфера совпадают с действительными адресами ячеек EEPROM микроконтроллера.

Впрочем, все эти особенности на правильность заполнения буферов содержимым HEX-файлов или соответствующих областей памяти микроконтроллера не влияют. А вот при ручном вводе кодов учитывать их необходимо.

Обратите внимание на правую часть окна IC-Prog. Здесь находится табло "Конфигурация". Вид его зависит от выбранного для программирования микроконтроллера. Например, на **рис. 44** оно показано в варианте для PIC16F628. Предусмотрена возможность выбрать тип генератора (осциллятора, ER CLKOUT

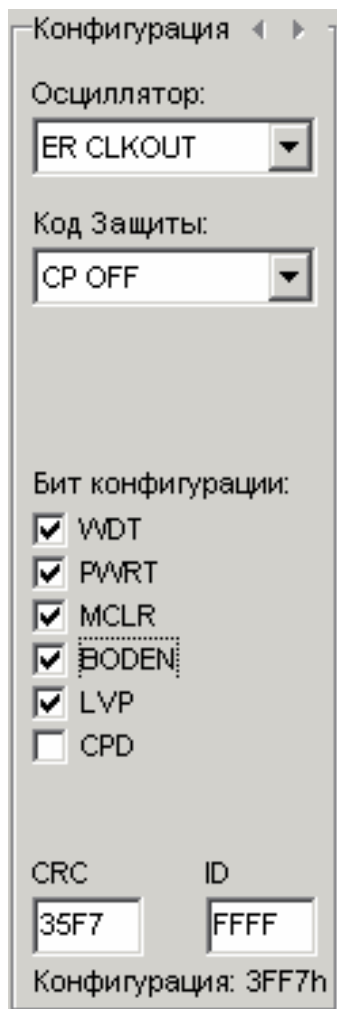


Рис. 44

расшифровывается как "установка частоты внешним резистором с выходом тактовых импульсов"), включить защиту кода (CP OFF — защита выключена) и задать значения других разрядов слова конфигурации. В нижней части окна выведено шестнадцатиричное значение CRC — циклического контрольного кода всего содержимого памяти микроконтроллера. Редактировать его значение нельзя, программа автоматически обновляет его при любом изменении содержимого любой ячейки буферов программной памяти, EEPROM, слова конфигурации и так называемого идентификационного (ID) кода. Для последнего в микроконтроллерах PICmicro отведена специальная область памяти, куда пользователь имеет возможность записать произвольное четырехразрядное шестнадцатиричное значение. ID код остается доступным для чтения даже при включенной защите, что позволяет при необходимости "опознать" хранящую его микросхему. Для его отображения и редактирования на табло "Конфигурация"

предусмотрено специальное

"Радио", 2004, № 10, с. 51

окно.

В нижней части табло можно увидеть значение слова конфигурации в шестнадцатиричном формате. А если дважды "щелкнуть" мышью в произвольном месте поля табло, будет открыто окно (**рис. 45**), позволяющее присвоить слову конфигурации шестнадцатиричное, десятичное или символьное (ASCII) значение, не занимаясь индивидуальной установкой отдельных двоичных разрядов.

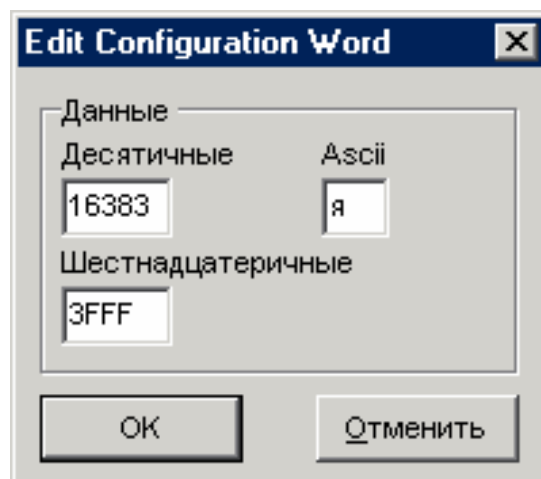


Рис. 45

Подключаем адаптер

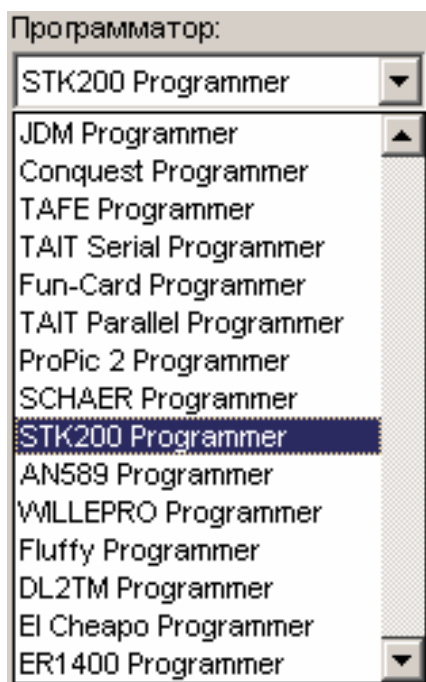



Рис. 46

Уже отмечалось разнообразие адаптеров, с которыми способна работать программа IC-Prog. Доступ к их списку, показанному на **рис. 46**, можно получить после выбора в меню "Настройки" пункта "Программатор", нажатия клавиши F3 или экранной кнопки . Со схемами этих "официально поддерживаемых программаторов" можно ознакомиться, открыв страницу <http://www.ic-prog.com/programmers.html>³.

Настройка программы на адаптер из списка во многом аналогична выполняемой в PonyProg. Достаточно указать COM или LPT порт, к которому подключен адаптер и способ общения программы с

портом: "напрямую" или через драйверы операционной системы. Исключение составляет настройка программной задержки, необходимой для формирования импульсов программирования. В PonyProg для этого предусмотрена операция автоматической калибровки, в IC-Prog задержку регулируют вручную. Обычно подходит значение, предлагаемое по умолчанию. На очень быстрых компьютерах его увеличивают, если при чтении данных из микроконтроллера и его

³ Все схемы адаптеров имеются и на FTP-сервере редакции по адресу <ftp://ftp.radio.ru/pub/2004/09/adapters/adapters.zip> — Прим. ред.

программировании наблюдаются сбои. Если компьютер "медленный", задержку можно уменьшить, что увеличит скорость программирования. Критерий допустимости уменьшения — отсутствие сбоев.

Для программирования микроконтроллеров PICmicro предназначено большинство "поддерживаемых" адаптеров. Они перечислены в **табл. 4**. Для тех, описания которых уже опубликованы в журнале, приведены "координаты" схем в формате "год-номер-страница".

Таблица 4

Адаптеры	Цепь			
	MCLR	CLOCK	DATA	DATA_IN
Подключаемые к параллельному порту (LPT)				
Conquest	D2 (4)	D1 (3)	D0 (2)	BUSY (11)
TAIT Serial	D3 (5)	D1 (3)	D0 (2)	ACK (10)
ProPic 2	D3 (5)	D1 (3)	D0 (2)	ACK (10)
SHAER	D2 (4)	D1 (3)	D0 (2)	BUSY (11)
AN589 (2004-1-55, рис. 2)	D4 (6)	D1 (3)	D0 (2)	ACK (10)
WILLERPROM	STROBE (1)	D1 (3)	D0 (2)	BUSY (11)
EI Cheapo	INIT (16/31)	STROBE (1)	SLCT_IN (17/36)	SELECT (13)
Подключаемые к последовательному порту (COM)				
JDM (2004-2-51, рис. 3)	TXD (3/2)	RTS (7/4)	DTR (4/20)	CTS (8/5)
SI-prog (2003-5-25, рис. 1)	TXD (3/2)	RTS (7/4)	DTR (4/20)	CTS (8/5)

В графах таблицы приведены названия цепей портов, к которым подключают адаптеры и номера соответствующих контактов интерфейсных разъемов. Для порта LPT — DB25M, установленного на компьютере и CEN-36, обычно подключаемого к принтеру. Как правило, номера совпадают. В противном случае они разделены дробной чертой (в знаменателе для CEN-36). Названия цепей в заголовке таблицы соответствуют приведенным в "Радио", 2004, № 1, с. 5, табл. 2.

В табл. 4 не включены адаптеры "TAIT Parallel" и "Fluffy", имеющие специфический интерфейс, зато в ней имеется адаптер SI-prog — основной для PonyProg. О настройке IC-Prog на работу с ним будет рассказано ниже.

Аналогичная предыдущей **табл. 5** содержит данные адаптеров для программирования микроконтроллеров AVR (серий AT90S, ATtiny, ATmega) и AT89 с последовательным интерфейсом программирования. В ней всего два "поддерживаемых" адаптера, оба подключают к порту LPT. Для адаптера SI-prog (точнее его части, предназначенной для микроконтроллеров AVR) указаны номера контактов разъема на плате адаптера. Об особенностях его применения также будет рассказано ниже.

Таблица 5

Адаптер	Цепь (2004-2-52, табл. 3)				
	RESET	MOSI	MISO	SCK	VCC
Fun-Card	D4 (6)	D5 (7)	ACK (10)	D6 (8)	D0—D2 (2—3)
STK200 (2004-2-52-8)	D7 (9)	D5 (7)	ACK (10)	D4 (6)	Внеш.
SI-prog (2001-7-19-5)	2/3	5	6	4	Внеш.

Микроконтроллеры AT89S1051, AT89S2051, AT89S4051 программируют с помощью адаптеров "Tafe" и "DL2TM". Последний был описан в "Радио", 2004, №3, с. 52 (рис. 12). Оба подключают к порту LPT. К нему же подключают адаптер "ER1400", предназначенный исключительно для одноименных микросхем памяти.

"Радио", 2004, № 11, с. 51, 52

Важная особенность программы IC-Prog — выбрав в меню "Настройки" пункт "Тест Программатора" (в результате будет открыто окно, показанное на **рис. 47**), можно вручную изменять логические уровни напряжения на выводах порта компьютера, к которому подключен адаптер программирования. Это позволяет с помощью осциллографа или вольтметра убедиться в правильности прохождения

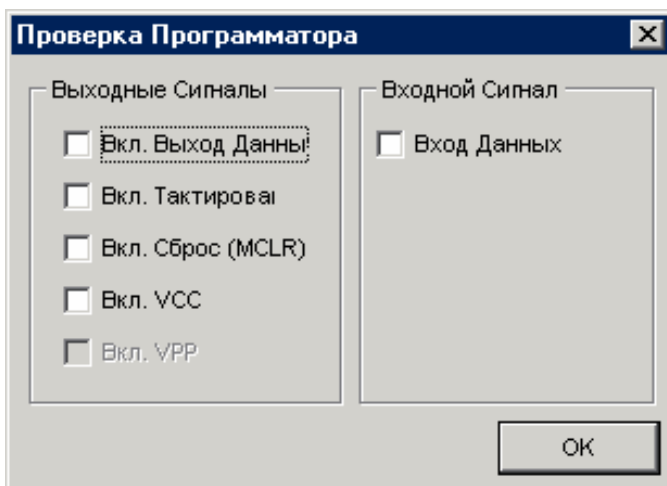


Рис. 47

сигналов от разъема порта до панели программируемой микросхемы. Сигнал "Выход Данных" — это DATA (см. табл. 4) или MOSI (см. табл. 5), "Тактирование" — соответственно CLOCK или SCK, "Сброс" — MCLR или RESET, "VCC" — управление питанием программируемой микросхемы. Если "окошко" сигнала отмечено "галочкой", на выводе порта будет установлен высокий, в противном случае — низкий уровень.

Проверяется и правильность восприятия компьютером сигнала "Вход Данных" (DATA_IN или MISO). В адаптерах для микроконтроллеров PICmicro линии DATA и DATA_IN связаны с одним и тем же контактом панели микросхемы, так как цепь передачи данных у этих микроконтроллеров двусторонняя. Поэтому при исправном адаптере любое изменение состояния линии DATA влечет

изменение состояния линии DATA_IN, что фиксирует "галочка", появляющаяся и исчезающая на панели "Входной Сигнал".

Чтобы получить тот же эффект в адаптерах для микроконтроллеров фирмы Atmel, необходимо временно соединить перемычкой контакты панели, на которые выведены сигналы MOSI и MISO (см. табл. 3 — "Радио", 2004, № 2, с. 52). Можно, конечно, и, не устанавливая перемычки, подавать на контакт MISO напряжение соответствующего уровня. Например, поочередно соединять его с общим проводом и плюсом источника питания микроконтроллера.

Учтите, все установки уровней на линиях порта действуют только до тех пор, пока окно (рис. 47) открыто. Закрывание окна возвращает порт в исходное состояние.

В радиоловительской литературе и в Интернете можно найти множество схем адаптеров программирования, которых нет в списке "официально поддерживаемых" программой IC-Prog. Тем не менее большинство из вполне пригодно для работы с этой программой. Необходимо лишь, проанализировав схему адаптера, найти указанные в табл. 4 или 5 цепи. Вполне возможно, что их подключение линиям портов компьютера совпадет с одним из упомянутых в этих таблицах адаптеров. Если точного соответствия найти не удалось, — не беда, проблема решается изготовлением соответствующего переходника.

Сравнивая схемы подключаемого и выбранного "эквивалентного" адаптера, обратите внимание на число инверсий сигналов на пути от выводов порта до выводов программируемой микросхемы. Если оно одинаково или разность числа инверсий четная, все в порядке. В противном случае поставьте "галочки" у соответствующих пунктов "Инверсия..." на панели "Параметры сигналов" окна "Настройка Программатора".

При подсчете числа инверсий обратите внимание, что многие микросхемы, используемые в качестве буферных, имеют похожие названия и одинаковую цоколевку, но различаются как раз наличием или отсутствием инверсии сигналов. Например, элементы микросхемы SN7406N (К155ЛНЗ) инвертируют сигналы, а SN7407N (К155ЛП9) — нет.

Как видно из табл. 4, адаптеры JDM и SI-prog используют для связи с компьютером одни и те же линии порта COM. Поэтому, настроив IC-prog не работу с JDM, вместо него можно подключить к порту адаптер SI-prog. Но для успешного программирования этого недостаточно. На пути сигнала данных от порта к программируемой микросхеме в SI-Prog (см. "Радио", 2001, № 7, с. 21, рис. 8)

имеется инвертор на транзисторе VT2, отсутствующий в JDM. Это учитывают установкой "галочки" "Инверсия Данных Вывода". В обратном направлении сигнал распространяется без инверсии в обоих случаях, так как в адаптере JDM (см. рис. 3 — "Радио", 2004, № 2, с. 51) каскад на транзисторе VT2, включенном по схеме с общей базой, неинвертирующий.

Опытным путем установлено, что необходимы также "Инверсия VCC" и "Прямой доступ к портам". Последнее связано с тем, что при работе через стандартный драйвер Windows изменение логического уровня на линии TXD происходит с большой задержкой, что нарушает алгоритм программирования.

В итоге окно "Настройка Программатора" для адаптера SI-prog, подключенного к порту COM, должно принять вид, показанный на **рис. 48**.

Программа IC-prog может работать и с адаптерами из комплекта SI-prog, подключенными к порту LPT компьютера. Плата согласования с портом COM ("Радио", 2001, № 6, с. 25, рис. 2) в этом случае не нужна. В окне "Настройка Программатора" следует выбрать адаптер STK200.

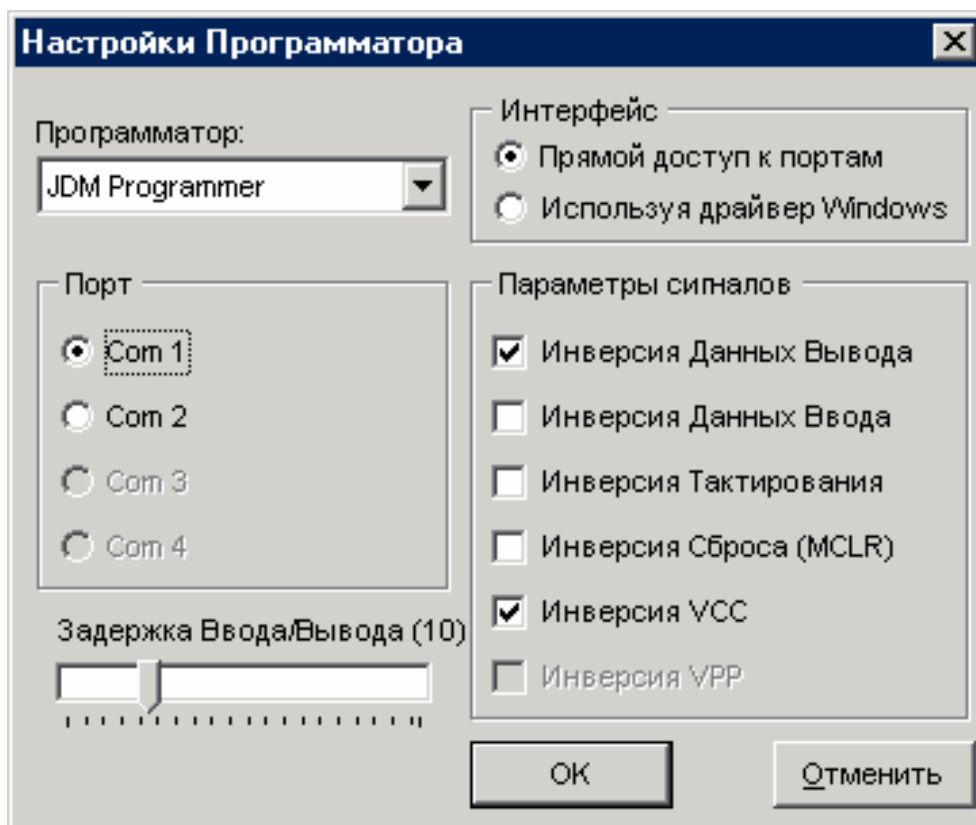


Рис. 48

Адаптеры для микроконтроллеров AT89 и AT90, ATtiny (см. "Радио", 2001, №7, с. 19, рис. 5), ATmega (там же, с. 20, рис. 7) и PICmicro можно подключить непосредственно к порту компьютера по схеме, показанной на **рис. 49**, но лучше

все-таки применить в качестве "промежуточного звена" адаптер STK200 соединяя с ним адаптеры SI-prog по схеме, изображенной на **рис. 50**.

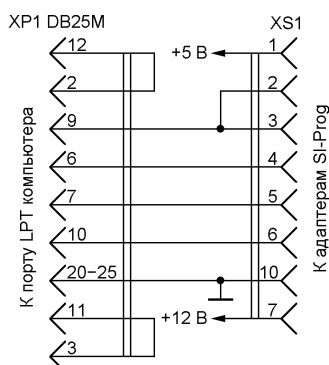


Рис. 49

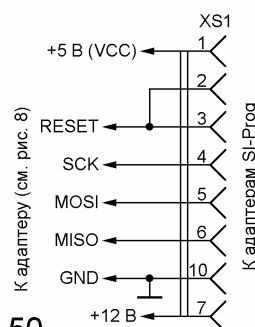


Рис. 50

Схема STK200 приведена на рис. 8 ("Радио", 2004, № 2, с. 52). Кварцевый резонатор ZQ1 в данном случае не нужен. Он уже имеется на плате предназначенного для программирования микроконтроллеров Atmel адаптера из комплекта SI-prog. Чертежи печатной платы и фотоснимок адаптера STK200 показаны на рис. 9 и 10 ("Радио", 2004, № 3, с. 51).

Вероятно, аналогичным образом можно подключить к порту LPT и другие адаптеры из комплекта SI-prog (см. "Радио", 2001, №7, с. 21, рис. 10—14), но на практике такая возможность не проверялась.

Напряжения +5 В и +12 В подают от внешних источников, причем последнее необходимо только для программирования микроконтроллеров PICmicro и лишь в случае, если в предназначенном для них адаптере не установлена батарея напряжением 9 В.

На **рис. 51** показано, как должно выглядеть окно "Настройка Программатора" для программирования микроконтроллеров серии PICmicro с помощью адаптера из комплекта SI-prog, подключенного к порту LPT. Для микроконтроллеров фирмы Atmel инвертировать сигнал данных не следует. Однако имеющийся в предназначенном для них адаптере переключатель SA1, изменяющий полярность сигнала сброса, в данном случае не действует. Поэтому нужную (неодинаковую для разных микроконтроллеров) полярность этого сигнала устанавливают программно, оперируя "галочкой" "Инверсия Сброса".

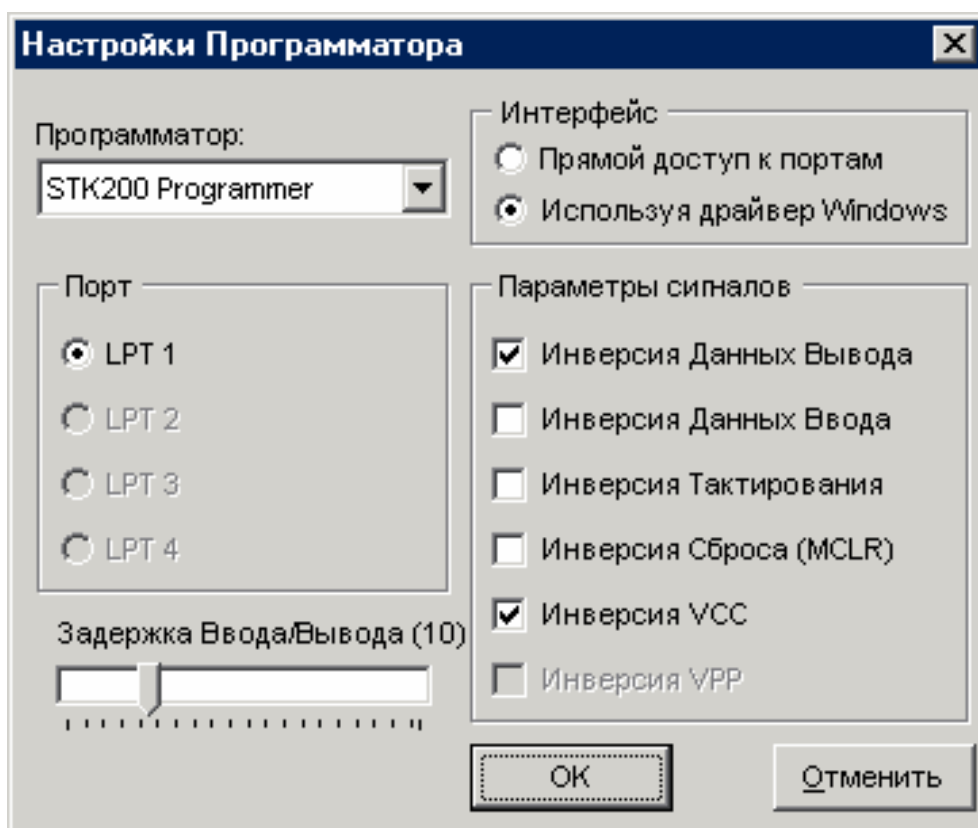


Рис. 51

"Радио", 2004, № 12, с. 47—49

Выбираем тип программируемой микросхемы

Есть несколько способов выполнить эту операцию. Первый из них — открыть, как показано на **рис. 52**, в меню "Настройки" пункт "Микросхемы", выбрать в нем семейство микросхем, затем — нужную микросхему из появившегося на экране списка. Учтите, под общим названием "Flash μ C" скрываются микроконтроллеры AT89C1051, AT89C2051 и AT89C4051, "SPI μ C" — AVR и AT89S, "Serial μ C" — микроконтроллеры серии P87 фирмы Philips, совместимые по структуре и системе команд с 8051. Имеются также микроконтроллеры фирмы Scenix, совместимые с микроконтроллерами PICmicro ("Microchip PIC"), но более скоростные.

Выбрать микросхему для программирования можно и из списка, выпадающего, как показано на **рис. 53**, из имеющегося в основном окне программы окошка с названием микросхемы, выбранной ранее. Этот способ менее удобен, так как на экране видна лишь небольшая часть общего списка и искать нужную микросхему приходится довольно долго.

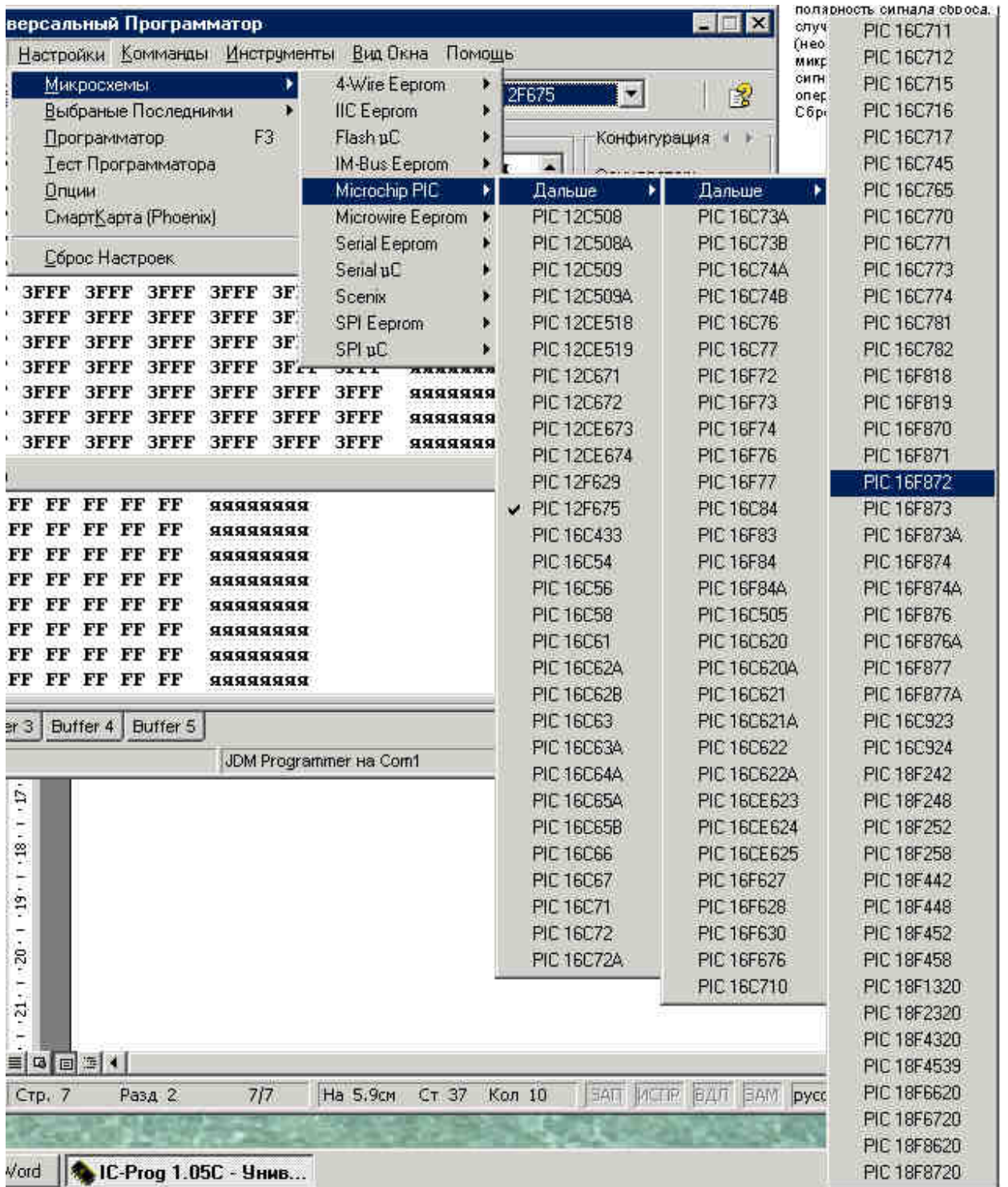


Рис. 52

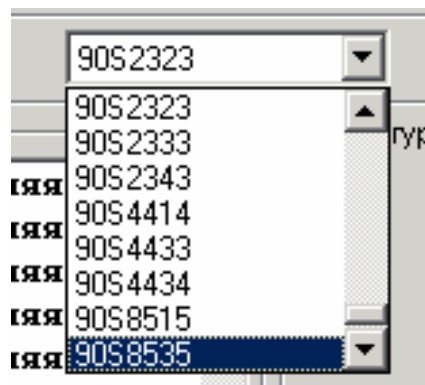


Рис. 53

Открыв в меню "Настройки" окно "Выбранные последними" (**рис. 54**), можно увидеть список из восьми микросхем, с которыми уже приходилось работать, и выбрать из него нужную. Это очень удобно, если приходится работать с небольшим числом разных микроконтроллеров, постоянно переходя от одного к другому.

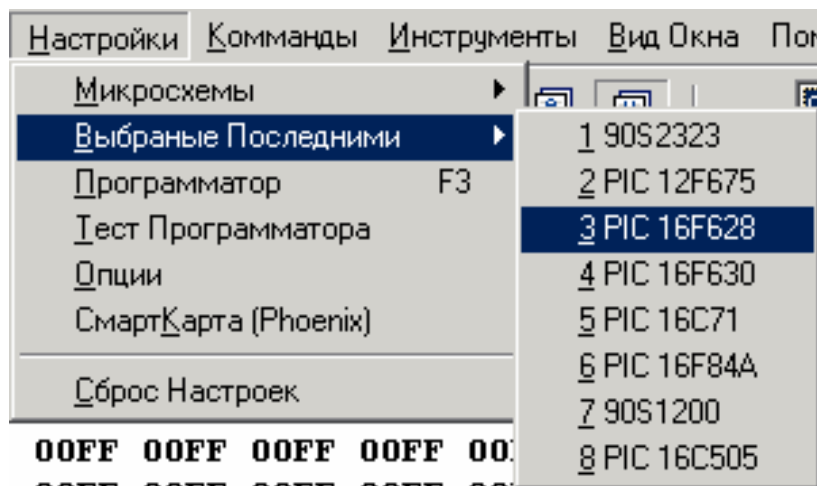


Рис. 54

Основные параметры выбранной микросхемы можно узнать из "подсказки", подобной показанной на **рис. 55**. Она будет выведена на экран при выборе пункта "Информация о микросхеме" в меню "Вид Окна". Стоит отметить и пункт "Положение Установки" этого меню⁴.

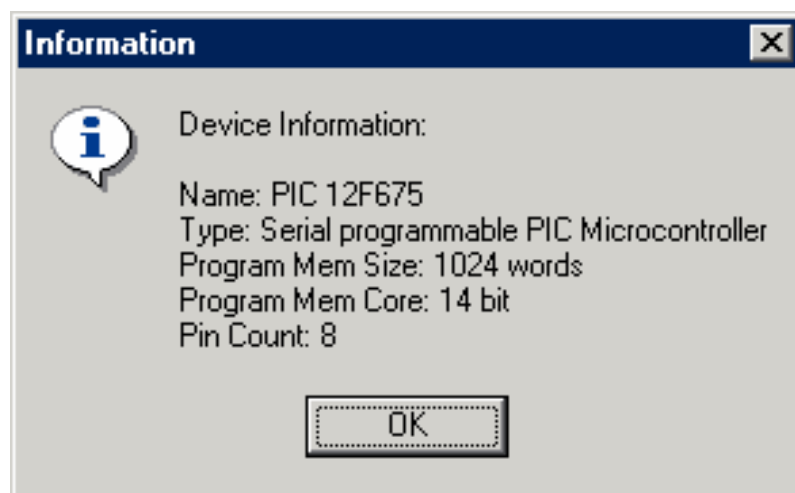


Рис. 55

⁴ Хочется принести читателям извинения за весьма неуклюжие названия меню и их пунктов. Но ничего не поделаешь, так их перевел с английского Алекс Кокаико, русифицировавший программу. — **Прим. автора**

Если выбранная микросхема может быть запрограммирована с помощью выбранного адаптера, на экране появится окно, подобное изображенному на **рис. 56**. Наглядно показано, каким образом следует устанавливать эту микросхему в панель адаптера.

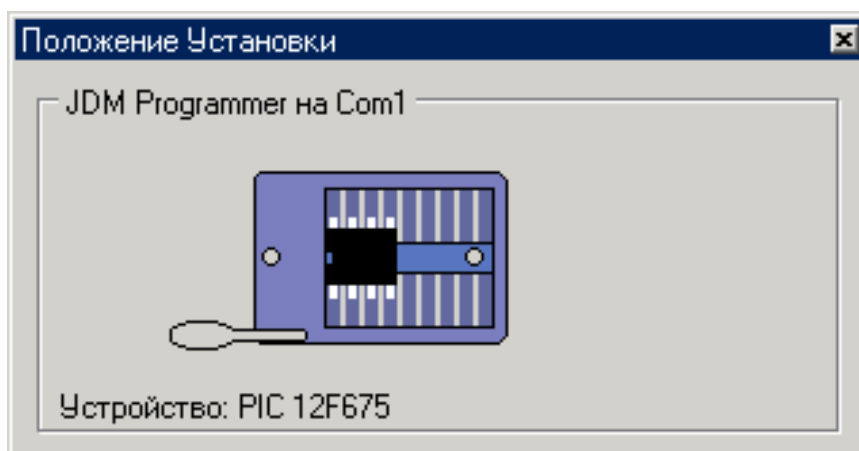


Рис. 56

Окно, показанное на **рис. 57**, говорит о том, выбранная комбинация микросхема-адаптер допустима, но в схему последнего нужно внести изменения или даже подключить вместо него другой. Именно так выглядит это окно при программировании микроконтроллера AT90S2323 с помощью адаптера SI-prog, соединенного с портом COM компьютера.

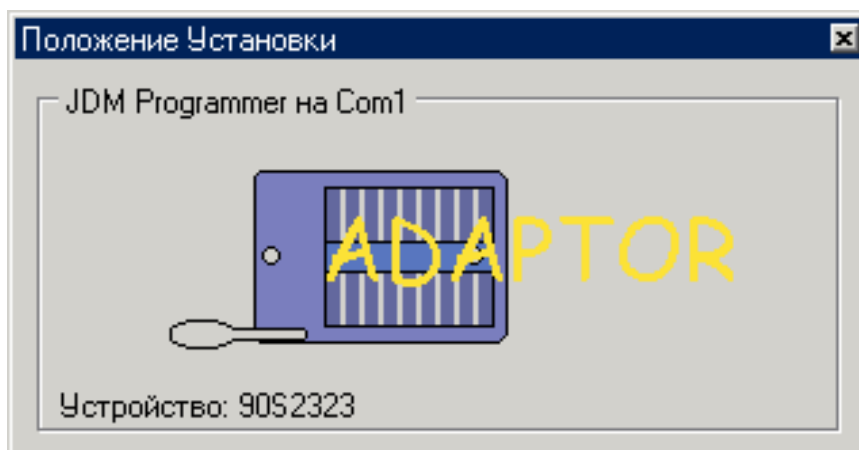


Рис. 57

В более простом случае, например, для программирования 28-выводного микроконтроллера PIC16F873 с помощью оснащенного 18-выводной панелью адаптера JDM, достаточно изготовить переходник, надлежащим образом соединяющий выводы микроконтроллера с контактами панели. Разработать схему переходника поможет табл. 2 ("Радио", 2004, № 1, с. 55).

Но самое неприятное из окон показано на **рис. 58**. Оно свидетельствует о полной несовместимости микросхемы с выбранным адаптером.

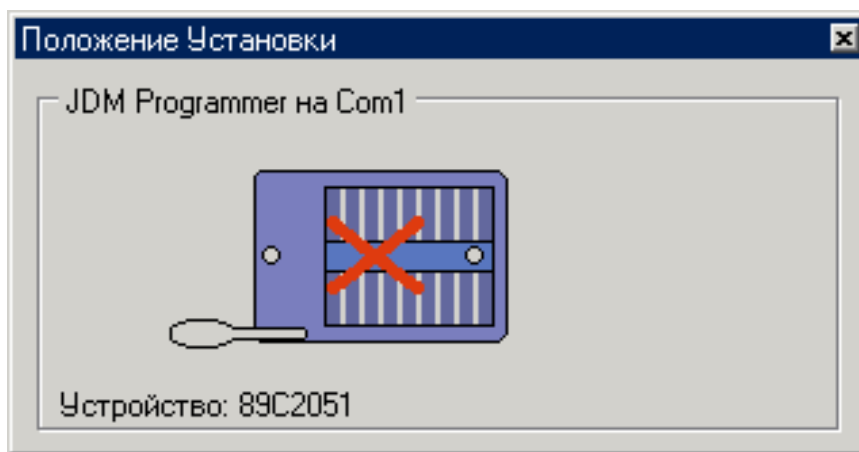


Рис. 58

Загружаем данные и программируем

Для загрузки буфера памяти программ достаточно выбрать в меню "Файл" пункт "Открыть Файл..." и указать в нем имя нужного файла. Аналогичный пункт для загрузки буфера EEPROM называется "Открыть Файл Данных...". Формат файла программа опознает автоматически. HEX-файлы для микроконтроллеров PICmicro обрабатываются корректно. Содержащаяся в них информация автоматически попадает в нужные буферы (памяти программ, EEPROM, конфигурации).

Пункт "Файлы Открытые Последними", позволяет избежать продолжительных поисков на дисках компьютера файлов, с которыми недавно уже приходилось работать. Файл, выбранный из выведенного на экран списка, будет загружен в ту область памяти (программы или EEPROM), в которую его загружали ранее.

Открыв на экране папку, в которой находится нужный файл, можно просто "перетащить" его значок мышью в окно буфера программы IC-prog. Но этот прием действует лишь в том случае, если он включен соответствующей "галочкой" на закладке "Перетаскивание мышью" пункта "Опции" меню "Настройка". На этой же закладке нужно выбрать тип "перетаскиваемых" файлов (раздельно для памяти программ и EEPROM). Автоматического распознавания в данном случае не происходит.

В программе IC-prog имеется пять независимых "комплектов" буферов, в которых храниться информация для программирования микроконтроллера. Операции выполняемые с одним "комплексом" буферов никак не влияют на содержимое других. Однако при любой смене типа программируемой микросхемы

программа IC-prog автоматически очищает **все** буферы. Поэтому загружать информацию в любой из них следует лишь после завершения всех операций по выбору микросхемы.

По умолчанию активен буфер под номером 1. К другим переходят, открывая соответствующую закладку в нижней части окна. Это же можно сделать, одновременно нажав клавишу **Ctrl** и цифру, соответствующую номеру буфера, или выбрав в меню "Буфер" пункт "Активный буфер"

Если выбрать пункт "Сравнить" того же меню, откроется окно, показанное на **рис. 59**. В данном случае после нажатия кнопки "Compare" программа сравнит содержимое буферов 1 и 2. Если оно идентично, процедура завершится сообщением "Buffers compared succesfully!". В случае несовпадения соответствующее слово или байт в окне активного буфера будет выделено цветом, а над или под ним — строка с тем же начальным адресом из другого буфера (**рис. 60**).

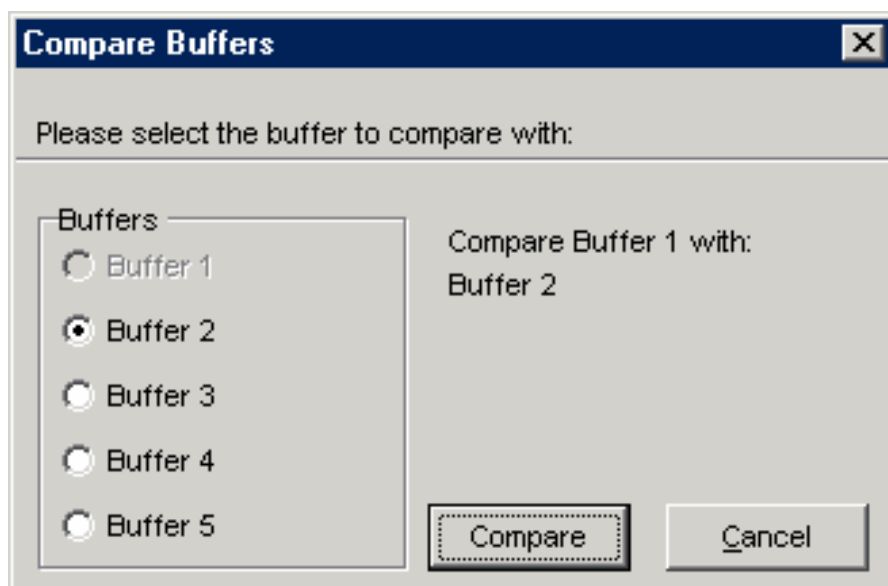


Рис. 59

```

0050: 1185 1D86 2857 200C 1C9D 2851 2821 1085 ...†W.ќQ!...
0040: 0D97 202D 0D90 202D 1003 1003 111D 202D --□-....-
0048: 3002 0088 3039 0089 21EF 109D 1585 218C .€9%тк..Њ
0050: 1185 1D86 2857 20BC 1C9D 2851 2821 1085 ...†WјќQ!...

```

Рис. 60

После нажатия появившейся в окне "Compare Buffers" кнопки "Next" получим аналогичную информацию о следующем несовпадении, а если их больше нет, — сообщение "Buffer compare done!". Если ошибок слишком много, процесс сравнения можно прервать с помощью кнопки "Cancel".



Чтобы исправить ошибки (или ввести новые данные), достаточно установить курсор в нужное место окна буфера и набрать новое значение на клавиатуре. Команды, имеющиеся в меню "Правка", позволяют заполнить одним и тем же значением ячейки всего буфера ("Заполнить Буфер", "Заполнить буфер Данных") или его части ("Заполнить Буфер из...", "Заполнить буфер Данных из..."⁵).

В последнем случае программа попросит указать не только значение кода, но и граничные адреса области памяти, в которую он должен быть записан.

Другие команды из меню "Правка" позволяют скопировать предварительно выделенную часть содержимого буфера и перенести ее, например, в другой буфер.

Операции, относящиеся к собственно программированию, сосредоточены в меню "Команды". Все они, приведены в табл. 6 с кратким описанием выполняемых действий. Команды можно подавать не только выбором пунктов меню, но и нажатиями указанных в таблице функциональных клавиш или экранных кнопок с соответствующими значками.

Таблица 6

Команда	Клавиша	Значок	Операция
Читать Все	F8		Копирование содержимого всех программируемых областей памяти микросхемы, в том числе ячеек конфигурации в активные буферы.
Программировать Все	F5		Загрузка текущего содержимого активных буферов во все программируемые области памяти микросхемы, в том числе в ячейки конфигурации.
Программировать Конфигурацию	F4	Нет	Программирование конфигурации микросхемы значениями, указанными в соответствующем окне
Стереть микросхему			Очистка всех областей программируемой памяти микросхемы
Проверка стирания	F6	Нет	Проверка, что вся программируемая память микросхемы находится в незапрограммированном состоянии.
Сравнить с буфером			Сравнение содержимого программируемой памяти микросхемы и активного буфера

Если во время предыдущего программирования в микроконтроллере была включена защита содержимого памяти от копирования, исполнение команды "Читать Все" приведет лишь к заполнению текущего буфера информацией, не совпадающей с истинным содержимым памяти микроконтроллера. В некоторых случаях буфер окажется заполнен байтами 0x7F, в других — нулями, в третьих последовательность байтов 0x11, 0x22, 0x33 и так далее.

⁵ Почему "из" — на совести "русификатора" программы. — Прим. автора

Состояние отвечающих за защиту памяти ячеек конфигурации микроконтроллеров серии PICmicro программатор правильно считывает и отображает на панели "Конфигурация" главного окна. К сожалению, для микроконтроллеров серий AT89 и AT90 это невозможно, о состоянии их защиты от копирования удается судить только по "правдоподобности" результата исполнения команды "Читать Все". Отключает защиту лишь полное стирание содержимого памяти (команда "Стереть Все"), причем даже из этого правила есть исключения.

В результате стирания все разряды всех программируемых ячеек памяти получают значение лог. 1. Командой "Проверка Стирания" убеждаются, что это действительно так и микросхема пригодна для загрузки новой информации.

На этом наш рассказ о программаторах и управляющих ими программах закончен. Естественно, он не мог охватить особенностей всех существующих аппаратных и программных средств этого назначения. Однако основы устройства и работы большинства из них очень близки к изложенным и мы надеемся, при работе с ними у читателей не возникнет непреодолимых трудностей.

До новых встреч!

ЛИТЕРАТУРА

1. **Долгий А.** Разработка и отладка устройств на МК. Часть 1. — Радио, 2001, № 5, с.17—19.
2. **Долгий А.** Как проверить HEX-файл. — Радио, 2003, № 8, с. 27, 28.
3. **Балахтарь А.** Программатор с питанием от LPT-порта для микроконтроллера KP1878BE1. — Радио, 2004, № 1, с. 29, 30.
4. **Долгий А.** Как проверить PonyProg. — Радио, 2003, № 5, с. 25, 26.
5. **Рюмик С.** Параллельный программатор. Радио, 2004, № 2, с. 28—30.
6. **Долгий А.** Разработка и отладка устройств на МК. Часть 3. — Радио, 2001, № 7, с.19—21.