Программатор микроконтроллеров семейства x51 фирмы Atmel

Существует достаточно большое количество программаторов для микроконтроллеров семейства x51 фирмы Atmel. Однако все они имеют ряд недостатков. Большинство программаторов предназначено для работы через LPT (параллельный) порт компьютера. Такое подключение имеет свои отрицательные черты: низкая нагрузочная способность порта, его высокая чувствительность к статическому электричеству, что не позволяет производить подключение/отключение программатора при включенном персональном компьютере (ПК).

Программатор и компьютер оказываются гальванически связанными, что требует повышенной осторожности при работе с программатором. Неудобством также является то, что к LPT порту подключается принтер и другие устройства, и поэтому перед началом программирования микроконтроллера приходится производить переключение устройств. Кроме того, большинство описанных программаторов реализуют только ограниченный набор функций.

В настоящей статье вниманию читателей предлагается описание варианта программатора, который подключается к ПК через свободный интерфейс RS-232C и имеет с ним гальваническую развязку, что позволяет подключать и отключать его при работающем компьютере.

Алгоритм программирования микроконтроллеров семейства x51

Алгоритм программирования микроконтроллеров (МК) семейства x51 описан в литературе [1—7]. Семейство x51 можно подразделить на два класса. Первый класс — МК типа AT89C1051/2051/4051 — выпускается в корпусе PDIP-20, второй класс — МК типа AT89C51/52/53/55/8252 — в корпусе PDIP-40. Алгоритмы программирования обоих классов МК достаточно близки, но отличаются тем, что для МК, относящихся к первому классу, ввиду ограниченного числа выводов микроконтроллеров, отсутствуют физические линии адресов, и адреса ячеек памяти переключаются путем подачи импульсов переключения адреса. Для МК второго класса адреса ячеек памяти задаются параллельным кодом на соответствующих выводах. Кроме того, микроконтроллеры каждого из классов имеют разные объемы памяти (табл. 1).

Для облегчения понимания принципиальной схемы программатора мы коротко рассмотрим оба алгоритма программирования и приведем временные диаграммы.

Как отмечалось выше, МК AT89C1051/2051/4051 имеют встроенный счетчик адресов, который сбрасывается в нулевое состояние при сбросе микроконтроллера по входу RST и наращивает адрес при подаче положительного импульса на вывод XLAT1. Микроконтроллеры AT89C51/52/53/55/8252 не имеют встроенного счетчика адресов, и адреса устанавливаются сигналами параллельного кода на соответствующих адресных линиях.

При программировании возможно выполнение шести команд для МК AT89C1051/2051/4051 и семи команд для МК AT89C51/52/53/55/8252 (добавляется команда записи третьего бита защиты). Выполняемая команда определяется кодом на четырех функциональных линиях F0…F3. В табл. 2 приведен перечень возможных команд.

Таблица 1. Объем Flash-памяти микроконтроллеров семейства x51 фирмы Atmel

PDIP20			PDIP40				
AT89C1051	AT89C2051	AT89C4051	AT89C51	AT89C52	AT89C53	AT89C55	AT89C8252
1K	2K	4K	4K	8K	12K	20K	8K

В отношении содержания таблицы необходимо сделать следующие пояснения:

1. символ “T” означает “отрицательный импульс” (+5 В => 0 => +5 В);
2. символ “H” означает высокий уровень или лог. 1 (+5 В);
3. символ “L” означает низкий уровень или лог. 0 (0 В);
4. команда очистки памяти требует длительности импульса программирования PROG/ более 10 мс;
5. некоторые микроконтроллеры AT89C51/52/53/55/8252 могут иметь напряжение программирования 5 В, о чем свидетельствует индекс “–5” в конце обозначения типа и соответствующий байт сигнатуры (см. [8]);
6. при программировании микроконтроллеров AT89C51/52/53/55/8252 на выв. RST должен быть подан уровень H, а на выв. PSEN/ — уровень L;
7. чтение сигнатуры МК AT89C1051/2051/4051 предполагает последовательное чтение байтов с адресами 0x00, 0x01, 0x02. Адрес 0x00 всегда содержит код 0x1E — код производителя Atmel. Адрес 0x01 содержит код микроконтроллера 0x11 (AT89C1051), 0x21 (AT89C2051), 0x41 (AT89C4051). Адрес 0x02 содержит код 0x00;
8. чтение сигнатуры МК AT89C51/52/53/55/8252 предполагает последовательное чтение байтов с адресами 0x30, 0x31, 0x32. Адрес 0x30 всегда содержит код 0x1E — код производителя Atmel. Адрес 0x31 содержит код микроконтроллера 0x51 (AT89C51), 0x52 (AT89C52), 0x53 (AT89C53), 0x55 (AT89C55), 0x72 (AT89S8252), Адрес 0x32 содержит код 0xFF, если напряжение программирования составляет 12 В, или 0x05, если напряжение программирования равно 5 В;

Таблица 2. Команды программирования микроконтроллеров семейства x51

№№	Функция	PROG/	Vpp	F0	F1	F2	F3
	AT89C1051/2051/4051 Наименование вывода, номер вывода	P3.2, 06	RST, 01	P3.3, 07	P3.4, 08	P3.5, 09	P3.7, 11
	AT89C51/52/53/55/8252 Наименование вывода, номер вывода	ALE, 30	ER/, 31	P2.6, 27	P2.7, 28	P3.6, 16	P3.7, 17
1	Запись кода ячейки памяти (Write Code Data)	T	H/12V	L	H	H	H
2	Чтение кода ячейки памяти (Read Code Data)	H	H	L	L	H	H
3	Запись 1-ого бита защиты (Write Lock Bit1)	T	H/12V	H	H	H	H
4	Запись 2-ого бита защиты (Write Lock Bit2)	T	H/12V	H	H	L	L
5	Запись 3-его бита защиты (Write Lock Bit3)	T	H/12V	H	L	H	L
6	Очистка памяти (Chip Erase)	T	H/12V	H	L	L	L
7	Чтение сигнатуры (Read Signature)	H	H	L	L	L	L

Таблица 3. Алгоритм программирования микроконтроллеров семейства x51

Последовательность операций при программировании микроконтроллеров		
№№	AT89C1051/2051/4051	AT89C51/52/53/55/8252
1	Подать питание на соответствующие выводы микросхемы. Подать на выв. Vpp (RST) и XLAT1 уровень лог. 0. Все остальные выводы должны быть свободными. Длительность этого состояния должна быть более 10 мс.	Подать питание на соответствующие выводы микросхемы. Подать на выв. RST уровень лог. 1. Подать на выв. PSEN/ уровень лог. 0. Все остальные выводы должны быть свободными. Длительность этого состояния должна быть более 10 мс.
2	Подать на выв. Vpp уровень лог. 1. Подать на выв. PROG/ уровень лог. 1.	
3	Установить комбинацию команды записи на линиях F0…F3.	
4	Установить код данных на выводах порта P1.	Установить код данных на выводах порта P0.
5	Установить на выв. Vpp напряжение 12 В (или уровень лог. 1, если микроконтроллер рассчитан на напряжение программирования 5 В).	
6	Сформировать на выв. PROG/ одиночный отрицательный импульс длительностью не менее 1,2 мс.	
7	Для проверки правильности записанного байта необходимо: перевести выв. Vpp в состояние лог. 1. Установить код команды чтения на линиях F0…F3. Считать код данных.	
8	Подготовить следующий байт. Подать одиночный импульс на выв. XLAT1 для увеличения адреса. Перейти к п. 3.	Подготовить следующий байт. Установить новый адрес. Перейти к п. 3.
9	Повторять пункты 3—8 до достижения последнего адреса Flash памяти программируемого микроконтроллера	
10	Последовательность выключения питания:	
	• установить на выв. XLAT1 уровень лог. 0;	• Установить на выв. Vpp уровень лог. 0;
	• установить на выв. Vpp уровень лог. 0;	• освободить все остальные выводы. Снять напряжение питания.
	• освободить все остальные выводы. Снять напряжение питания.	

9. при программировании МК AT89C1051/2051/4051 команда 5 отсутствует.

Собственно алгоритм программирования МК семейства x51 приведен в табл. 3.

Следует отметить, что существует два варианта задания времени программирования каждого байта. При фиксированном времени программирования длительность импульса программирования фиксирована и составляет более 1,2 мс. При изменяющемся времени программирования после подачи импульса программирования PROG/ необходимо анализировать уровень на выв. BUSY/ (P31 для микроконтроллеров AT89C1051/2051/4051 и P34 для AT89C51/52/53/55/8252). Переход уровня из состояния лог. 1 в состояние лог. 0 означает, что цикл программирования байта завершен и необходимо перевести уровень сигнала PROG/ в состояние лог. 1.

Второй вариант задания времени программирования позволяет сократить время программирования микросхемы, но усложняет программу. Поэтому в описываемом программаторе используется первый вариант задания времени программирования — фиксированный.

Временные диаграммы цикла программирования микроконтроллеров AT89C1051/2051/4051 показаны на рис. 1, а микроконтроллеров AT89C51/52/53/55/8252 — на рис. 2.

Внимательное рассмотрение приведенных выше данных позволяет сделать вывод о незначительных отличиях в последовательности программирования микроконтроллеров обоих семейств и возможности совмещения функций программирования в едином алгоритме.

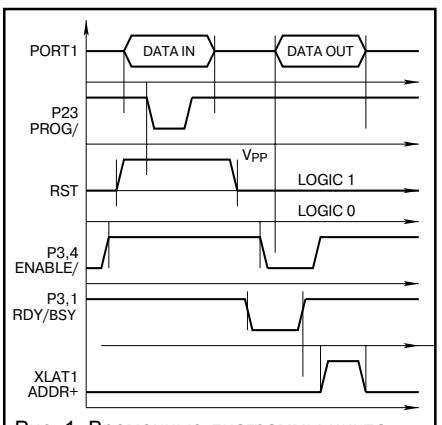

Рис. 1. Временные диаграммы цикла программирования микроконтроллеров AT89C1051/2051/4051

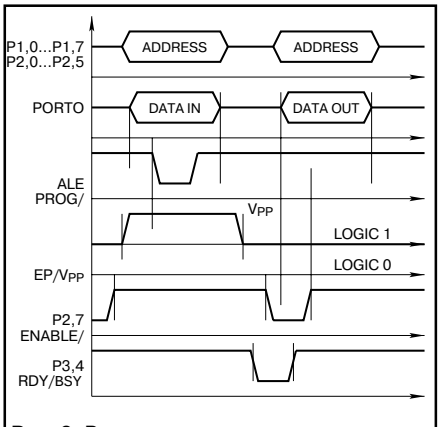

Рис. 2. Временные диаграммы цикла программирования микроконтроллеров AT89C51/52/53/55/8252

Принципиальная схема программатора

Структурная схема программатора приведена на рис. 3. Он содержит следующие основные узлы: узел последовательного интерфейса RS-232C для связи программатора с ПК (RS-232C Interface); узел микроконтроллера (Controller); узел панелей для подключения программируемых микросхем (Sockets), узел формирования напряжений программирования (Convertor); источник питания (Power Supply).

Принципиальная схема узла панелей для подключения программируемых микросхем представлена на рис. 4. Схема содержит две зажимные панели для корпусов DIP-20 (для микроконтроллеров AT89C1051/2051/4051) и DIP-40 (для микроконтроллеров AT89C51/52/53/55/8252). К панели DIP-40 подключены кварцевый резонатор и конденсаторы в соответствии с рекомендациями фирмы Atmel [8]. В этом же узле находится двухцветный светодиодный индикатор L-819 SRSG D фирмы Kingbright, предназначенный для индикации режимов работы программатора. Отсутствие свечения означает, что панельки обесточены (можно вставлять или вынимать микросхемы), зеленый цвет свечения — выполняются команды чтения, либо не производятся никакие действия, красный цвет свечения — выполняются команды записи или стирания. Узел выполнен в виде отдельной платы и крепится к верхней крышке прибора, в которой имеются отверстия под панели и светодиод. Узел соединяется с другими узлами с помощью плоского кабеля и штыревых разъемов PLD-40.

Узел последовательного интерфейса RS-232C для связи программатора с ПК показан на рис. 5. Он выполнен на трех микросхемах оптронов 4N35, обеспечивающих гальваническую развязку программатора с ПК. Напряжение изоляции — до 2,5 кВ. Два оптрона предназначены для передачи информации, третий — для реализации функции сброса. Линия RTS служит для формирования положительного напряжения, а линия DTR — отрицательного. Номиналы резисторов обеспечивают минимальную нагрузку на линии RS-232C при скорости 28800 бит/с. Описанная схема позволяет подключать программатор к включенному ПК.

Принципиальная схема узла микроконтроллера представлена на рис. 6. Он содержит собственно микроконтрол-

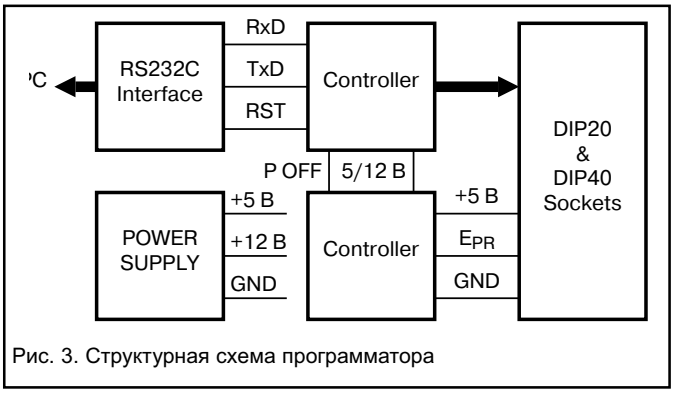

Рис. 3. Структурная схема программатора

лер D1 (AT89C2051), регистры D4—D7 (74HC573), шинный формирователь D3 (74HC245) и логику блокировки выходов D2 (74HC00). Регистры D4, D5 предназначены для формирования кода адреса, D6 — для формирования управляющих сигналов. Регистр D7 и шинный формирователь D3 образуют порт приема/передачи данных. Подобная архитектура узла контроллера имеет преимущества перед возможной архитектурой на одном контроллере AT89C51 (рис. 8): более низкое потребление (около 7 мА), малая стоимость, защита программатора при программировании некачественных микроконтроллеров. Кроме того, она может быть реализована на более доступных микросхемах и позволяет практически полностью обесточивать панельки. Выбор микросхем серии 74HC обусловлен их сверхнизким потреблением при высоком быстродействии. При отсутствии микросхем этой серии, их можно заменить, не изменяя схемы, на более доступные отечественные микросхемы серий 1555 или 555: D4–D7 — на ИР33, D3 — на АП6, D2 — на ЛА3. Естественно, это приведет к резкому возрастанию потребляемого тока. Можно также использовать регистры КР580ИР83 и шинный формирователь КР580ВА86, но это также вызовет возрастание тока потребления до 550 мА.

С другой стороны, наличие более мощных регистров и шинного формирователя может оказаться в некоторых случаях полезным. Автору пришлось иметь дело с несколькими партиями МК серий AT89C82 и AT89C53 произведенных в Корее и на Тайване, которые упорно “не хотели” программироваться при номинальном напряжении +12 В, однако безошибочно программировались при напряжении 12,3…12,42 В. Во втором случае при попытке программирования при номинальном напряжении резко возрастал потребляемый ток по входам адреса и функций. Очевидно, что в этом случае будет полезно использовать микросхемы серии 555 вместо серии 74HC.

На рис. 7 показан узел формирования напряжений программирования. Он реализован на микросхеме К155ЛН3. Элементы D1.1 и D1.2 образуют схему формирования напряжения программирования +12, +5 и +0,3 В. Остальные элементы коммутируют двухцветный светодиод, показанный на рис. 4. Узел управляется двумя сигналами от контроллера. Сигнал P5_12V в активном состоянии (лог. 1) обеспечивает напряжение программирования +5 В (устанавливается резистором R12), при этом включается зеленый светодиод. В пассивном состоянии (лог. 0) формируется напряжение +12 В, и включается красный светодиод. Второй управляющий сигнал POW_OFF в активном состоянии (лог. 1) выключает напряжение программирования (примерно +0,3 В) и светодиод.

Как уже отмечалось выше, узел микроконтроллера (рис. 6) можно реализовать вообще на одной микросхеме AT89C51. При этом надежность программатора снижается ввиду отмеченных выше причин, зато упрощается рабочая программа. Вариант такой реализации узла микроконтроллера показан на рис. 8. Назначения входов/выходов узла соответствуют схеме, приведенной на рис. 3.

Следует сказать несколько слов об источнике питания. В программаторе применен классический трансформаторный источник питания с двумя выходными напряжениями — +5 и +12 В. Потребляемый ток от источника напряжения +12 В составляет не более 20 мА, при этом можно использовать линейные стабилизаторы 78L12. Потребление сильно зависит от используемой в узле микроконтроллера элементной базы и может изменяться в пределах 100—700 мА. А если еще учитывать вероятность программирования некачественных микроконтроллеров, то необходимо иметь запас по току до 1,2 А от

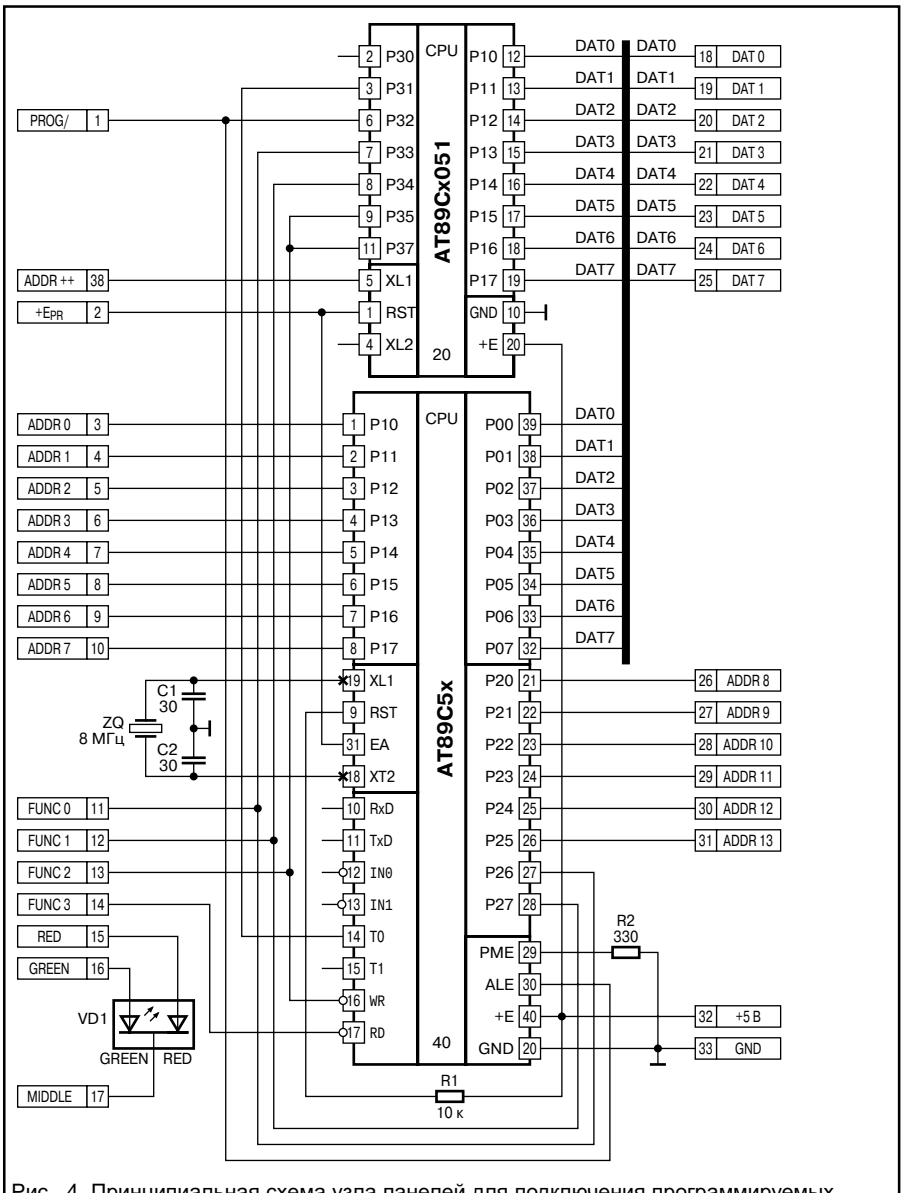

Рис. 4. Принципиальная схема узла панелей для подключения программируемых микросхем

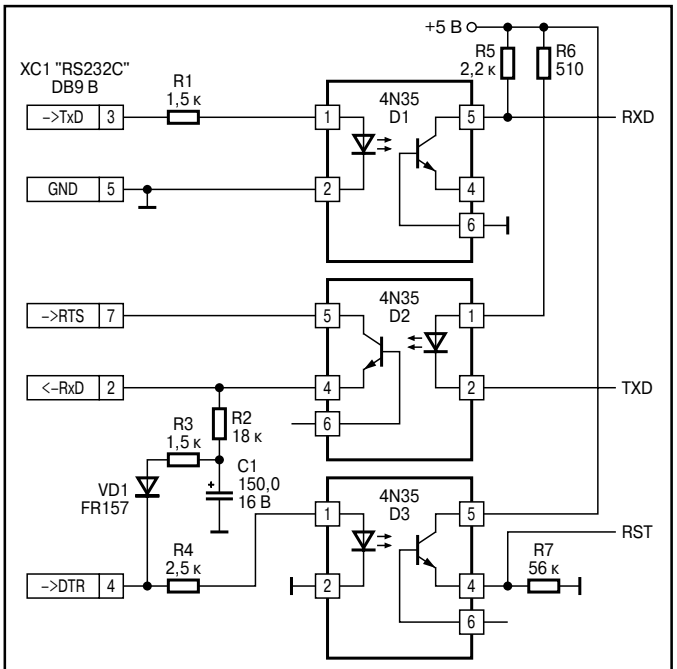

Рис. 5. Узел последовательного интерфейса RS-232C

источника +5 В. Кроме того, рекомендуется применять регулируемый источник напряжения +12 ±3 В со встроенным стрелочным вольтметром и, желательно, амперметром.

Рабочая программа для основного варианта узла микроконтроллера

Ниже приводится рабочая программа для основного варианта узла микроконтроллера (рис. 6). ПК работает с программатором в режиме командно-информационной сети. Это означает, что ПК является ведущим, а программатор — ведомым. Компьютер выдает команды, а программатор выполняет их. Программа написана на языке С и состоит из двух файлов. Первый файл, "prog.h", содержит определения кодов всех команд и микроопераций, выполняемых программатором. Ниже приведен фрагмент файла "prog.h".

Фрагмент 1

```
/*******************************************************/
/* PR0G.H       ver.15.00 Program For Programmer AT89C2051    */
/* Created by O. Nicolajchuk  27.10.97   Last Modify 05.02.99    */
/*******************************************************/
#include "..\include\io51.h"

#define             FALSE           0
#define             TRUE            1
#define             byte            unsigned char
#define             MHZ12           FALSE  /* Назначена частота
11.059MHz */
#define             OE_INP          P3.7            /* Линия от-
крытия данных */
#define             RG_Addr_L               P3.2            /    *
Строб регистра младшего байта адреса */
#define             RG_Addr_H               P3.3            /    *
Строб регистра старшего байта адреса */
#define             RG_Func                 P3.4            /    *
Строб регистра функций */
#define             RG_Data         P3.5            /* Строб реги-
стра данных */

/**** Определения вспомогательных команд и операций ****/

#define     CMD_SetAL               0x01            /* Установить
младший адрес */
#define     CMD_SetAH               0x02    /* Установить старший
адрес */
#define     CMD_SetFunc     0x03    /* Установить регистр функций */
#define     CMD_SetData             0x04    /* Установить регистр
данных записи */
#define     CMD_GetData     0x05    /* Прочитать данные */
#define     CMD_OutputOn    0x06            /* Выходы регистров
включить */
#define     CMD_OutputOff   0x07            /* Выходы регистров
выключить */

#define     CMD_RST_H       0x10    /* Установить напряжение PROG/
= 5V */
#define     CMD_RST_12      0x11    /* Установить напряжение PROG/
= 12V */

#define     CMD_Set_RS      0x12    /* Установить функцию"Read
Signature" */#define     CMD_Set_RCD     0x13    /* Установить функцию
"Read Code Data" */#define     CMD_Set_WCD     0x14    /* Установить
функцию "Write Code Data" */#define     CMD_Set_WLB1    0x15    /* Ус-
тановить функцию "Write Lock Bit1" */
#define     CMD_Set_WLB2    0x16    /* Установить функцию "Write Lock
Bit2" */
#define     CMD_Set_WLB3    0x17    /* Установить функцию "Write Lock
Bit3" */
#define     CMD_Set_CE      0x18    /* Установить функцию "Chip
Erase" */
#define     CMD_Prog_H      0x19    /* Установить Prog = 5V */
#define     CMD_Prog_L      0x1A    /* Установить Prog = 0V */
#define     CMD_NextAddr    0x1B    /* Установить следующий адрес */
#define     CMD_Reset               0x1C            /* Установить
адрес = 0 */
#define     CMD_Puls                0x1D    /* Установить следую-
щий адрес 2051 */

#define     CMD_RF                  0x22    /* Выполнить операцию
"Read Flash" */
#define     CMD_LB1                 0x23    /* Выполнить операцию
"Write Lock Bit1" */
#define     CMD_LB2                 0x24    /* Выполнить операцию
"Write Lock Bit2" */
#define     CMD_LB3                 0x25    /* Выполнить операцию
"Write Lock Bit3" */
#define     CMD_CE                  0x26    /* Выполнить операцию
"Chip Erase" */
#define     CMD_WF                  0x27    /* Выполнить операцию
"Write Flash" */

/**** Определения основных команд и операций ****/

#define     CMD_Speed               0x3F    /* Установить скорость */
#define     CMD_Speed2400 0x30              /* Установить 2400 */
#define     CMD_Speed4800 0x31              /* Установить 4800 */
#define     CMD_Speed9600 0x32              /* Установить 9600 */
#define     CMD_Speed19200          0x33            /* Установить
19200 */
#define     CMD_Speed28800          0x34            /* Установить
28800 */

#define     CMD_DoTest              0x7C    /* Выполнить тест */

/**** Резервирование регистров памяти ****/

byte  Ready;
byte  Rate;
byte  CMD;
byte        ADDR_L;
byte        ADDR_H;
byte        FUNC;
byte        DATA;

#if         MHZ12
code  StartSpeed            = 26;
code  SpeedMask             = 0x03;
#else
code  StartSpeed            = 24;
code  SpeedMask             = 0x1F;
#endif

/* End of PR0G.H */
```

Второй файл, "pr015.c", содержит собственно текст программы. Ниже приведен фрагмент файла "pr015.c".

Фрагмент 2

```
/*******************************************************/
/* PR015.C  ver.15.00   Program For Programmer AT89C2051   */
/* Created by O. Nicolaiciuc  27.10.97     Last Modify 05.02.99   */
/*******************************************************/
```

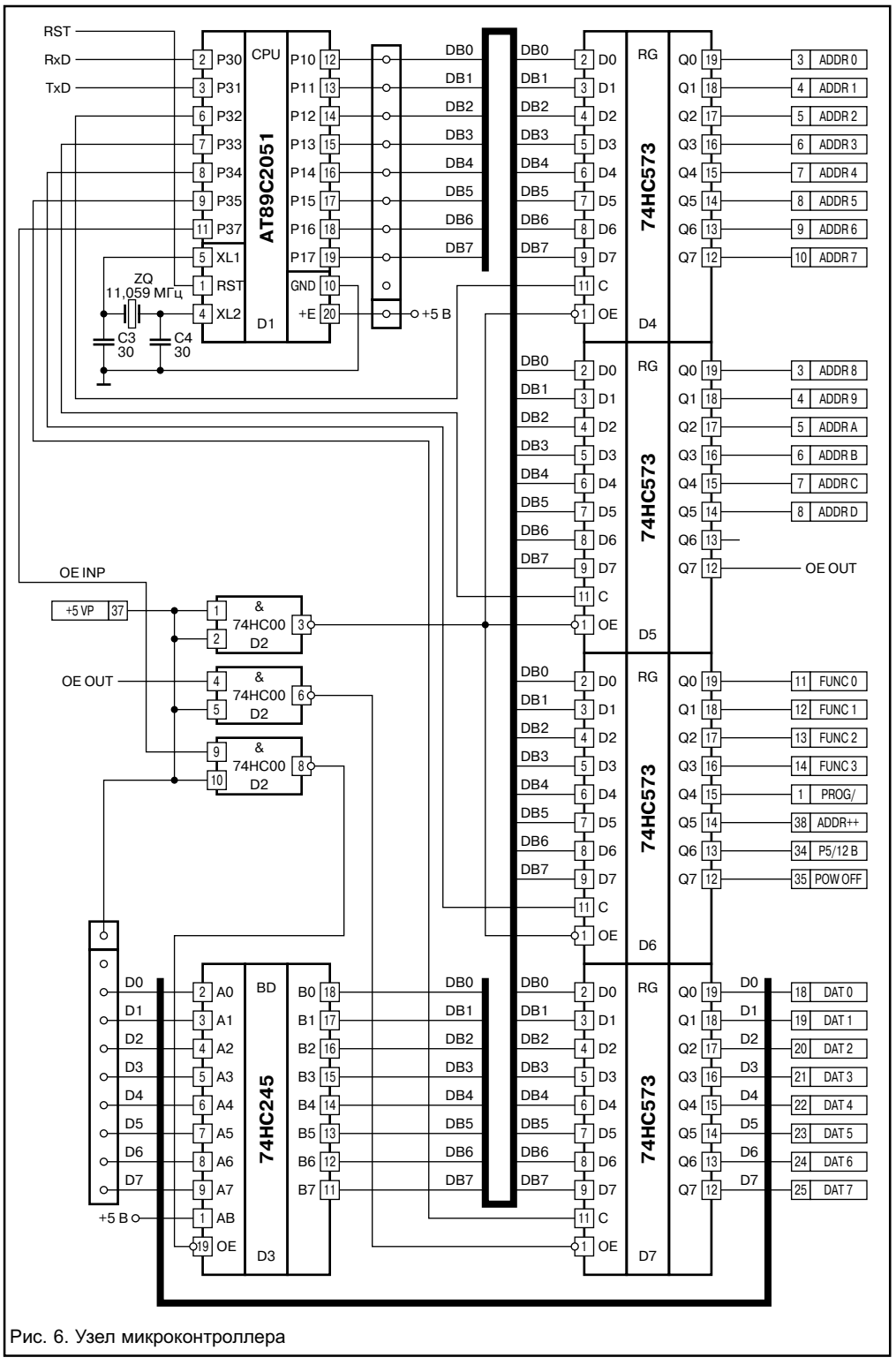

Рис. 6. Узел микроконтроллера

```
#include "prog.h"
interrupt [0x23] void SCON_int (void) /* Вектор прерывания RS232C */
{
      Ready=1;
}

void TimersInit (void)              /* Процедура инициализации таймера */
{
      EA=0;                         /* Запретить все прерывания */
      TMOD=0x21;                    /* Таймер 1 в режиме 2: 8-bit
счетчик с автозагрузкой */
                                    /* Таймер 0 в режиме 1: 16-bit
счетчик */
      EA=1;                         /* Разрешить все прерывания */
}

void SpeedInit (byte NewSpeed)  /* Инициализация скорости */
    {
    char ch;

      TR1     = 0;
/* Выключить таймер 1 */
      PCON    = PCON | 0x80;        /*
Двойная скорость */
      TH1     =          TL1        =
(NewSpeed ^ 0xFF)+1;
      ch      = SBUF;               /*
Очистить входной регистр */
      SCON    = 0x52;
      TR1     = 1;      /*    Включить
таймер 1 */
    }


    void SendB (byte SendByte)
/* Послать байт */
    {
      while (!TI);
/* Ожидать завершения посылки пре-
дыдущего байта */
      TI=0;               /*    Очистить
флаг */
      SBUF=SendByte;            /* По-
слать байт */
    }

    byte GetB (void)
/* Получить байт */
    {
    byte InputByte;

      while (!RI);                /     *
Ожидать приема байта */
      InputByte = SBUF;
/* Байт получить */
      RI = 0;
/* Флаг приема сбросить */
      return InputByte;           /* Байт
вернуть */
    }

    void Time (char Delay)
/* Задержка ~ (10+7*Delay) mkS */{
while (Delay) {Delay=Delay-1;}
}void Delay (char Step)
/* Задержка ~ Step mS */
    { while (Step—) {Time (160);}
      }

    void ChangeSpeed (void)
/* Установить скорость */
    {
    byte NewSpeed;
    byte R2400;
      NewSpeed= GetB();
/* Получить байт (код) скорости */
      if (Rate == NewSpeed)     /* Если
скорость старая, завершить подпрог-
рамму */
    {SendB (Rate); return;}
      Rate = NewSpeed & SpeedMask;
/* Проверить допустимость кода ско-
рости */
      SendB   (Rate);
      if          ((Rate&0x10)==0x10)
R2400=2;                  /*28800*/
      else if     ((Rate&0x08)==0x08)      R2400=3;
/*19200*/
      else if     ((Rate&0x04)==0x04)      R2400=6;                  /
*9600*/
      else if     ((Rate&0x02)==0x02)      R2400=StartSpeed/2;
/*4800*/
      else if     ((Rate&0x01)==0x01)      R2400=StartSpeed;
/*2400*/
      SpeedInit   (R2400);                /* Установить заданную скорость
*/
    }

    /***** Элементарные подпрограммы установки регистров *****/
    void SetAL          (void)    {     P1=ADDR_L;      RG_Addr_L=1;
RG_Addr_L=0; P1=0xFF;}
    void SetAH          (void)    {     P1=ADDR_H;      RG_Addr_H=1;
RG_Addr_H=0; P1=0xFF;}
```

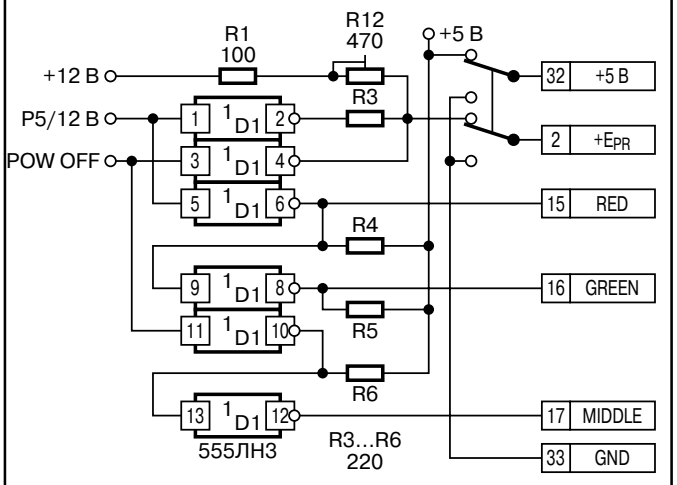

Рис. 7. Узел формирования напряжений программирования

```
   void SetFunc            (void)   { P1=FUNC;                RG_Func=1;
RG_Func=0;  P1=0xFF;}
   void SetData            (void)   { P1=DATA;                RG_Data=1;
RG_Data=0;  P1=0xFF;}
   /***** Подпрограммы переключения регистра данных *****/
   void OutputOn (void) { ADDR_H=(ADDR_H&0x7F)|0x80;    SetAH();}
   void OutputOff (void) { ADDR_H=(ADDR_H&0x7F);
SetAH();}
   /***** Подпрограммы установки регистра функций *****/
   void RST_H              (void)   { FUNC=(FUNC&0x3F)|0x40;
SetFunc();}
   void RST_12             (void)   { FUNC=(FUNC&0x3F);
SetFunc();}
   void Set_RS             (void)   { FUNC=(FUNC&0xF0);
SetFunc();}
   void Set_RCD  (void)     { FUNC=(FUNC&0xF0)|0x0C;
SetFunc();}
   void Set_WCD  (void)     { FUNC=(FUNC&0xF0)|0x0E;
SetFunc();}
   void Set_WLB1 (void)     { FUNC=(FUNC&0xF0)|0x0F;
SetFunc();}
   void Set_WLB2 (void)     { FUNC=(FUNC&0xF0)|0x03;
SetFunc();}
```

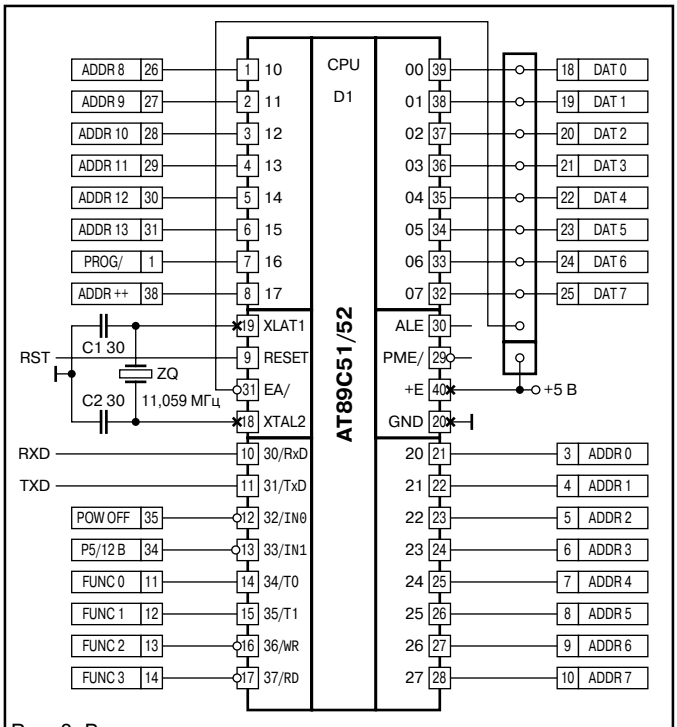

Рис. 8. Вариант реализации узла микроконтроллера на одной микросхеме

```
   void Set_WLB3(void)     { FUNC=(FUNC&0xF0)|0x05;
SetFunc();}
   void Set_CE             (void)   { FUNC=(FUNC&0xF0)|0x01;
SetFunc();}
   void Prog_H             (void)   { FUNC=(FUNC&0xEF)|0x10;
SetFunc();}
   void Prog_L             (void)   { FUNC=(FUNC&0xEF);
SetFunc();}
   void Puls_H             (void)   { FUNC=(FUNC&0xDF)|0x20;
SetFunc();}
   void Puls_L             (void)   { FUNC=(FUNC&0xDF);
SetFunc();}
   /***** Подпрограммы формирования импульсов *****/
   void Puls               (void)   { Puls_H(); Time(2); Puls_L();}
   void Prog               (void)  { Prog_L(); Time(15); Prog_H();}
   void DoTest  (void)                    /* Передать тестовую последова-
тельность */
   {
   byte i;

        Delay (1);
        for (i=0; i<255; i++)       SendB (i);
        SendB (0xFF);
   }

   void GetData (void)                    /* П/программа чтения данных */
   {
        P1              =0xFF;         /* Порт на ввод */
        OE_INP  =1;             /* Открыть шинный формирователь дан-
ных */
        Time            (1);           /* Задержка */
        DATA            =P1;           /* Считать данные */
        Time            (1);           /* Задержка */
        OE_INP  =0;             /* Закрыть шинный формирователь дан-
ных */
        SendB           (DATA); /* Передать считанные данные */
   }

   void NextAdd (void)                    /* Установить следующий адрес */
   {
        if      (ADDR_L<0xFF)  /* Если младший байт адреса < 0xFF */
                ADDR_L++;              /* Увеличить младший байт */
        else                           /* Иначе */
        {
                ADDR_L=0;              /* Младший байт обнулить */
                ADDR_H++;              /* Увеличить старший байт */
        }
        SetAL ();                      /* Установить младший адрес */
        SetAH ();                      /* Установить старший байт */
        Puls  ();                      /* Увеличить адрес микроконтроллеров
89Cxx51 */
   }

   void Counter_RST (void)                /* Сбросить счетчик адреса */
   {
        ADDR_L=0;       SetAL();       /* Обнулить младший байт адреса
*/
        ADDR_H=0x40; SetAH();   /* Обнулить старший байт адреса */
        FUNC    =0xC0;  SetFunc();     /* RST=L, ReadSignfnure, Prog_L,
Puls_L */
        OE_INP=0;                      /* Закрыть шинный формирова-
тель данных */
        Delay   (10);                  /* Задержка ~ 10 mS */
        Prog_H ();                     /* Установить начальные уровни…
*/
        RST_H ();
        Delay   (3);                   /* Задержка ~ 3 mS */
   }

   void ReadFlash (void)          /* Читать байт из Flash */{ Set_RCD();
                /* Установить код функции чтения */      P1=0xFF;
                /* Порт на ввод */OE_INP=1;                       /* От-
крыть шинный формирователь */
        Time(1);                       /* Задержка */
        DATA    =P1;                   /* Считать данные */
        Time(1);                       /* Задержка */
        OE_INP=0;                      /* Закрыть шинный формирова-
тель */
        NextAdd ();                    /* Установить следующий адрес */
        SendB   (DATA);                /* Передать считанный байт */
   }

   void RProg (void)                      /* Последовательность програм-
мирования */
   {
        RST_12();                      /* Установить напряжение про-
```

```
граммирования 12В */      Time (15);                    /* Задержка */
Prog();                   /* Импульс программирования */
OutputOff();              /* Закрыть регистр данных */      Delay
(4);                      /* Задержка */
        RST_H();          /* Установить напряжение программиро-
вания 5В */
        Time(5);          /* Задержка */
    }

    void WriteFlash (void)        /* Записать байт во Flash */
    {
        DATA    =GetB();          /* Получить байт для записи */
        Set_WCD();                /* Установить код функции запи-
си */
        SetData();                /* Установить данные */
        OutputOn();               /* Открыть регистр данных */

        RProg();                  /* Последовательность программирования
*/
    }

    void LockBit1 (void)          /* Записать "Lock Bit1" */
    {
        Counter_RST();            /* Сбросить счетчик адреса */
        Set_WLB1();               /* Установить код функции запи-
си "Lock Bit1" */
        RProg();                  /* Последовательность программирования
*/
    }

    void LockBit2 (void)          /* Записать "Lock Bit2" */
    {
        Counter_RST();            /* Сбросить счетчик адреса */
        Set_WLB2();               /* Установить код функции запи-
си "Lock Bit2" */
        RProg();                  /* Последовательность программирования
*/
    }

    void LockBit3   (void)        /* Записать Lock Bit3 */{
Counter_RST       ();             /* Сбросить счетчик адреса */
Set_WLB3();                       /* Установить код функции записи "Lock
Bit3" */
        RProg();                  /* Последовательность программирования
*/
    }

    void ChipErase (void)         /* Очистить (Стереть) Flash */
    {
        Counter_RST();            /* Сбросить счетчик адреса */
        Set_CE();                 /* Установить код функции  "Chip
Erase" */
        RST_12();                 /* Установить напряжение про-
граммирования 12В */
        Time(15);                 /* Задержка */
        Prog_L();                 /* Начать импульс стирания */
        Delay(10);                /* Задержка ~ 10 mS */
        Prog_H();                 /* Завершить импульс стирания */
        Time(15);                 /* Задержка */
        RST_H();                  /* Установить напряжение программиро-
вания 5В */
    }

    void main (void)              /* Монитор команд */
    {
    byte   a;

        Counter_RST ();           /* Установить счетчики адреса */
        TI=1;                     /* Установить флаги передачи и
приема */
        RI=0;
        TimersInit();             /* Инициализировать таймеры */
        SpeedInit (StartSpeed);   /* Установить начальную скорость 2400 */
        Delay(5);                 /* Задержка ~ 5 mS */

        while (1)                 /* Основной командный цикл */
        {
            while (!TI);          /* Ожидание приема …… */
            if (RI)               goto get;
            Ready=0;
            TI=0;
            ES=1;
            while (1)
            {
                PCON = PCON | 0x80;     /*idle/pdown*/
                if (Ready)        /* Если байт принят */
                {
                    ES        = 0;
                    Ready = 0;
                    TI        = 1;
                    goto get;
                }
            }
    get:  CMD    =GetB();         /* Записать байт команды */
          switch (CMD)
          {
              case CMD_SetAL:ADDR_L=GetB ();/* Set Low Address */
                                  SetAL ();
                                  break;
              case CMD_SetAH:     ADDR_H=GetB ();       /*  Set
High Address */
                                  SetAH ();
                                  break;
              case CMD_SetFunc:   FUNC=GetB ();  /* Set Functions
*/
                                  SetFunc ();
                                  break;
              case CMD_SetData:  DATA=GetB ();      /* Set Write Data
*/
                                  SetData ();
                                  break;
              case CMD_GetData:   GetData ();             /*  Get
Read Data */
                                  break;
              case CMD_OutputOn: OutputOn ();       /* Data Output
On */
                                  break;
              case CMD_OutputOff: OutputOff ();     /* Data Output
Off */
                                  break;
              case CMD_RST_H:     RST_H ();         /* Set RST = 5V
*/
                                  break;
              case CMD_RST_12:    RST_12 ();        /* Set RST = 12V
*/
                                  break;
              case CMD_Set_RS:    Set_RS ();              /*  Set
Function "Read Signature" */
                                  break;
              case CMD_Set_RCD: Set_RCD ();   /* Set Function "Read Code
Data" */
                                  break;
              case CMD_Set_WCD: Set_WCD (); /* Set Function "Write
Code Data" */
                                  break;
              case CMD_Set_WLB1: Set_WLB1 (); /* Set Function "Write Lock
Bit1" */
                                  break;
              case CMD_Set_WLB2: Set_WLB2 (); /* Set Function "Write Lock
Bit2" */
                                  break;
              case CMD_Set_WLB3: Set_WLB3 (); /* Set Function "Write Lock
Bit3" */
                                  break;
              case CMD_Set_CE:    Set_CE ();              /*  Set
Function "Chip Erase" */
                                  break;
              case CMD_Prog_H:    Prog_H ();              /*  Set
Prog = 5V */
                                  break;
              case CMD_Prog_L:    Prog_L ();              /*  Set
Prog = 0V */
                                  break;
              case CMD_NextAddr: NextAdd ();  /* Set Next Address */
                                  break;
              case CMD_Reset: Counter_RST ();  /* Set Address =0 */
                                  break;
              case CMD_Puls:  Puls ();          /* Set Next Address xx51
*/
                                  break;
              case CMD_RF:    ReadFlash ();     /* Read Flash */
                                  break;
              case CMD_LB1:   LockBit1 ();              /* Write Lock Bit1
*/
                                  break;
              case CMD_LB2:   LockBit2 ();              /* Write Lock Bit2
*/
                                  break;
              case CMD_LB3:   LockBit3 ();              /* Write Lock Bit3
*/
                                  break;
              case CMD_CE:    ChipErase ();     /* Chip Erase */
```

```
                                        break;
                case CMD_WF:        WriteFlash ();              /* Write Flash */
                                        break;
                case CMD_Speed:         ChangeSpeed ();  /* Set Speed */
                                        break;
                case CMD_Speed2400: SpeedInit (StartSpeed); /* 2400 */
                                        break;
                case CMD_Speed4800: SpeedInit (StartSpeed/2); /* 4800 */
                                        break;
                case CMD_Speed9600: SpeedInit (6);          /*9600 */
                                        break;
                case CMD_Speed19200:SpeedInit (3);          /* 19200 */
                                        break;
                case CMD_Speed28800:SpeedInit (2);          /* 28800 */
                                        break;
                case CMD_DoTest:        DoTest ();                  /*  Test
*/
                                        break;
                default:            SendB (0xEE);       /* Error */
                                        SendB (CMD);
                                        break;
                }
        }
    }
    /***************     END OF PROGRAM  ****************************/
```

Рабочая программа верхнего уровня также написана на языке Си. Она предназначена для работы в среде операционной системы MS-DOS, либо в среде эмуляции MS-DOS операционных систем Win95/98.

Программа имеет достаточно большие возможности. Отметим лишь некоторые из них.

1. Программа обеспечивает все возможные команды, связанные с программированием — выбор программируемого микроконтроллера, чтение сигнатуры и ее распознавание, проверку Flash на чистоту, чтение Flash в основной программный буфер, сравнение содержимого Flash и буфера, запись массива Flash с произвольного адреса и произвольной длины, запись битов защиты, стирание Flash.
2. Программа обеспечивает чтение данных для записи во Flash в бинарном и шестнадцатеричном (Intel) видах.
3. Прочитанные из Flash в программный буфер данные также могут быть записаны на диск.
4. Программа имеет второй “стековый” программный буфер. Между основным и “стековым” буферами возможны операции чтения, записи, сравнения и обмена данными.
5. Программа имеет настройку скорости обмена и номера порта.

Окно программы имеет вид, схожий с окном Norton Commander, занимает достаточно мало места (менее 100 кБ). Загрузить ее можно по адресу **http://www.dian.ru/programs/index.html.**

Автором разработан и изготовлен ряд различных программаторов, основанных на изложенных в данной статье принципах. Многие из них успешно эксплуатируются уже более 6 лет и показали неплохие эксплуатационные качества. Очевидно, что по этому принципу можно разработать и изготовить программаторы для других микроконтроллеров и микросхем, а также создать специализированные или универсальные устройства отладки и эмуляции.

Олег Николайчук,
onic@ch.moldpac.md

Литература

1. *Atmel Corporation Microcontroller Data Book, Atmel. Oct.,1995.*
2. *Doc500.pdf http://www.atmel.ru http://www.atmel.com*
3. *Doc501.pdf http://www.atmel.ru http://www.atmel.com*
4. *Doc504.pdf http://www.atmel.ru http://www.atmel.com*
5. *Doc505.pdf http://www.atmel.ru http://www.atmel.com*
6. *Doc507.pdf http://www.atmel.ru http://www.atmel.com*
7. *89C55.pdf http://www.atmel.ru http://www.atmel.com*
8. *Doc510.pdf http://www.atmel.ru http://www.atmel.com*