Микроконтроллеры AVR. Ступень 1

С.М. Рюмик, г. Чернигов

Тот, кто мягко ступает,
далеко продвинется на своем пути.
Китайская пословица

Продолжаем начатый в прошлом году рассказ о микроконтроллерах (МК). Надеемся, что это поможет радиолюбителям шире применять их на практике.

Электронщики в шутку говорят, что после третьей изученной МК-платформы легко перейти на четвертую. Тем же, кто добросовестно выполнял задания из цикла статей об МК семейства MCS-51 (РА 3-12/2004), можно поставить галочку за первую пройденную платформу. "Пройденную" - это не значит "забытую" или "устаревшую", скорее, прибавившую знания в схемотехнике и программировании.

Если представить процесс изучения МК в виде покорения горной вершины, то первые "10 шагов" были сделаны по "равнине" платформы MCS-51. На очереди подъем по ступеням "предгорья" платформы AVR фирмы Atmel.

Почему в качестве второй выбрана именно платформа AVR? Специалисты определили ряд здоровых принципов выбора МК: не гнаться за экзотикой, не связываться с единственным семейством, не экономить на средствах отладки и программирования. Для платформы AVR все перечисленное выполняется. Вдобавок, соблюдается преемственность, поскольку ранее пройденный 8-разрядный МК AT89C2051 тоже был разработан на фирме Atmel.

Начинать изучение МК сразу с "модных" AVR, означало бы отход от основных педагогических принципов - постепенности и последовательности. К примеру, лет 20...30 назад среди вузов электронного профиля Украины существовало понятие "школа". Солидные преподавательские составы "школ" Харьковского института радиоэлектроники, радиофакультетов Киевского политехнического и Харьковского авиационного институтов гарантировали базовые знания студентов. Молодые специалисты, прошедшие "школу", легко становились программистами, разбирающимися в схемотехнике, или радиоинженерами, умеющими составлять программы.

Точно так же и в изучении МК. Без прохождения "школы" платформы MCS-51 трудно будет понять нюансы в архитектуре AVR, поскольку не с чем сравнивать, не имея опыта.

История появления AVR

Фирма Atmel была основана в 1984 г. в знаменитой Кремниевой долине (Калифорния, США). В середине 90-х годов ее основной продукцией стали микросхемы памяти и перепрограммируемые МК платформы MCS-51. По сравнению с аналогичными изделиями фирм Intel, Philips, Temic, OKI, Siemens, микросхемы Atmel были дешевле, ни в чем не уступая им по качеству. Одна из составляющих успеха - создание филиалов производства в странах Юго-Восточной Азии.

Всем хороши были МК платформы MCS-51 за исключением энергопотребления и производительности. Там, где использовалось маломощное (батарейное) питание и требовалась высокая скорость обработки данных, разработчики предпочитали PIC-контроллеры фирмы Microchip Technologies, МК серии H8/300 фирмы Hitachi и МК фирмы Dallas Semiconductor.

Ситуация в корне изменилась в 1996 г., когда было объявлено о начале серийного производства принципиально новых 8-разрядных контроллеров платформы AVR.

У архитектуры AVR скандинавская родословная. В 1995 г. два норвежских изобретателя Альф Боген и Вегард Воллен предложили фирме Atmel концепцию нового МК. Идея была принята. Базовые принципы и система команд разрабатывались в норвежском отделении фирмы Atmel совместно со шведскими программистами фирмы IAR Systems. Имена разработчиков вошли в название платформы в виде двух первых заглавных букв - **A**lf Bogen / **V**ergard Wollan / **R**isc architecture.

Достоинства AVR: быстродействующий RISC-процессор, FLASH-память с низковольтным напряжением программирования, внутреннее перезаписываемое ЭСПЗУ, мощные выходные порты, широкий диапазон питающего напряжения. И все это при малом потреблении тока, высокой скорости, а главное, при низкой цене. По совокупному интегральному параметру "энергопотребление - производительность - цена" AVR-контроллеры оказались лучшими в мире.

Классификация AVR

Платформа AVR насчитывает 4 семейства: "classic", "tiny", "mega", "LCD". В 1997 г. в каталоге фирмы Atmel впервые появились четыре "classic"-МК с маркировкой AT90Sxxxx. В каталогах 1999 г. были представлены уже три семейства: "classic" (AT90S), "tiny" (ATtiny), "mega" (ATmega). В дальнейшем развитие "классического" семейства было заморожено в связи с большой номенклатурой МК и их самодостаточностью.

С 2000 г. начался перевод производства с технологических норм 0,5 мкм на 0,35 мкм. Изменения коснулись в основном семейств "tiny" и "mega". Разработчики не только уменьшили размеры кристаллов, но заодно увеличили тактовые частоты, объем ПЗУ, ввели новые интерфейсы, снизили удельное энергопотребление и исправили некоторые ошибки, проявлявшиеся при эксплуатации. Новые МК получили другие обозначения и позиционировались в качестве замены один к одному устаревшим микросхемам.

В 2004 г. МК, имеющие выводы для подключения ЖК-индикаторов, были выделены в отдельное семейство "LCD AVR". В октябре 2004 г. последним двум микросхемам семейства "classic" присвоен статус EOL (End-Of-Life), т.е. они не рекомендуются для новых разработок и в середине 2005 г. окончательно уйдут с производства. Из четырех семейств остались только три (**табл.1**), зато какие!

Первоначально все микросхемы AVR заметно различались друг от друга по числу выводов корпуса: "tiny" - 8 выводов, "classic" - 40-44 вывода, "mega" - 64 вывода. В дальнейшем грани стерлись, но тенденция осталась. Для DIP-микросхем, которые чаще всего применяются в любительской практике, действует ряд: 8, 20, 28, 40 выводов.

Еще один нюанс. Первые цифры в названии AVR-контроллеров обозначают объем FLASH-ПЗУ в килобайтах. Например, ATtiny15L (1 Кб), ATtiny26 (2 Кб), AT90S4414 (4 Кб), ATmega8515 (8 Кб), ATmega162 (16 Кб), ATmega32 (32 Кб), ATmega 6450 (64 Кб), ATmega128 (128 Кб).

Программатор AVR

Практически все AVR-контроллеры имеют функцию внутрисистемного программирования ISP (In-System Programming). Это означает, что для зашивки кодов программы не требуется извлекать МК из платы и устанавливать его в панель программатора. Теперь "гора сама идет к Магомету", т.е. компьютер через специальный адаптер подключается к разъему ISP, установленному на плате изделия (**рис.1**). Адаптер нередко устроен таким образом, что по окончании программирования он автоматически отключается от выводов МК, не мешая работе остальных узлов. Процесс многократных экспериментов и перепрошивок, по сравнению с AT89C2051, теперь идет гораздо быстрее.

Шина связи МК с адаптером содержит 6 сигналов (**табл.2**): три входных, один выходной и два по питанию. Информация передается в последовательном виде по протоколу SPI (Serial Programming Interface). Чтобы не запутаться в обозначениях, надо запомнить простое мнемоническое правило: выходной сигнал MISO единственный, который имеет в конце букву "O" (output). А слово "MOSI" по звучанию похоже на японское приветствие телефонных абонентов (вместо "Алло").

Каждый из информационных сигналов подключается к определенному выводу МК, точнее, к линии порта, имеющей альтернативное на-

Таблица 1

Семейство AVR	Обозначения микросхем	Параметры
"tiny"	ATtiny13, 15L, 2313, 25, 26, 28L, 45, 85	1-8 Кб ПЗУ, 64-512 байт ОЗУ, DIP8-TQFP32
"mega"	Atmega48, 8, 88, 8515, 8535, 16, 162, 165, 168, 32, 325, 3250, 64, 645, 6450, 128	4-128 Кб ПЗУ, 256-4096 байт ОЗУ, DIP28-TQFP64
"LCD"	Atmega169, 329, 3290, 649, 6490	16-64 Кб ПЗУ, 1024-4096 байт ОЗУ, TQFP64-TQFP100

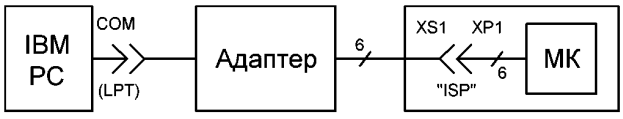

рис.1

Таблица 2

Сигнал	Расшифровка	Функция	Назначение
SCK	**S**erial **C**loc**K**	Вход МК	Тактовый сигнал в МК
MOSI	**M**aster **O**ut – **S**lave **I**n	Вход МК	Информационный сигнал в МК
MISO	**M**aster **I**n – **S**lave **O**ut	Выход МК	Информационный сигнал из МК
GND	**Gro**u**ND**	Общий	Общий провод
$\overline{RES}$	**RES**et	Вход МК	Лог."0" – программирование
VCC	**V**oltage **C**ommon **C**ollector	Питание	Напряжение питания 2,7...5,5 В

звание MISO, MOSI или SCK (**табл.3**). Двойное назначение выводов заимствовано из платформы MCS-51. При всем разнообразии микросхем AVR, выводы SPI у них строго закреплены, даже два варианта корпусов DIP-40 имеют различие только по выводам питания.

Разъемы XP1, XS1 (рис.1) 10-контактные, соответственно вилка на плату BH-10 и розетка на плоский кабель IDC-10F. К сожалению, унификация в распайке выводов отсутствует. В **табл.4** приведены наиболее часто встречающиеся варианты. Первый из них разработан фирмой Altera для программирования ПЛИС через адаптер ByteBlaster. Два последующих варианта применяются в отладочном комплексе STK200 (STK300) фирмы Atmel и в ее внутрисистемном программаторе AVR910 [1].

Практическое следствие: если на чужой плате с AVR-контроллером находится 10-контактный разъем под ISP, то не надо спешить подключать к нему свой программатор, ведь цоколевка может не совпадать.

Преимущество распайки а-ля "ByteBlaster" заключается в универсальности, поскольку одним и тем же устройством можно прошивать МК и программировать ПЛИС. Однако это требует специального программного обеспечения. Чтобы отделить "зерна от плевел", предлагается использовать две другие распайки (**рис.2,а, б**). Название "STK200" стало "де-факто" промышленным стандартом. Вариант "AVR910" позволяет использовать разъемы с меньшим числом контактов (жаль, что IDC-6 не существует!). Кроме того, он абсолютно безвреден при случайной установке 10-контактного разъема "задом наперед": ни МК, ни адаптер гарантированно не выйдут из строя.

Для любительских конструкций выбор одного из двух вариантов распайки непринципиален. Однако в целях единообразия на схемах адаптеров будет приведен вариант "AVR910". Разумеется, это всего лишь расположение контактов в разъеме, который никак не влияет на электрические параметры.

Выбор схемы адаптера

Первым делом напрашивается мысль изготовить фирменный адаптер по схеме, приведенной в [1]. Однако наличие в нем контроллера AT90S1200, который надо предварительно запрограммировать на каком-то другом программаторе, плюс неунифицированное программное обеспечение делают эту работу малопривлекательной.

С другой стороны, в Интернете, журнальных и книжных публикациях растиражированы клоны более простых, но не менее эффективных схем. Легко найти к ним и свободное программное обеспечение. Многочисленные похвальные отзывы людей, повторивших эти схемы, убеждают в правильности направления поиска.

Выбор электрической схемы адаптера зависит от трех факторов:

управляющей компьютерной программы, предназначенной для прошивки МК;

типа порта, используемого в компьютере (COM или LPT);

требуемого уровня сервисных и защитных функций.

Управляющих программ, допускающих работу с AVR, известно много. Например, AVreal (http://www.ln.com.ua/~real/avreal), IC-Prog (http://www.ic-prog.com), PonyProg (http://www.lancos.com), Willem Eprom (http://www.willem.org). Все они бесплатные и поддерживают широкую номенклатуру AVR-контроллеров. Любую из них можно использовать в дальнейшей работе, изготовив адаптер по приводимым на сайтах схемам. И все-таки, придется взять одну базовую программу, хотя бы для того, чтобы у читателей меньше возникало вопросов: "А почему не работает?".

Выбор остановим на программе PonyProg (автор Claudio Lanconelli, Италия). Во-первых, поддержка PonyProg заложена в Си-компиляторах и отладчиках. Во-вторых, с этой программой читатели уже работали в "Шаге 9", РА 11/2004, файл "mk9.zip", http://www.ra-publish.sea.com.ua.

Тип порта подключения к компьютеру определяется системной конфигурацией. Если LPT-порт постоянно занят под принтер, то выбирают COM-порт, и наоборот. Дополнительные аргументы. Длина соединительного кабеля между адаптером и COM-портом может составлять 5...8 м, а для LPT-порта - только 1,5...2 м. С другой стороны, схемы LPT-адаптеров проще в конструкции и содержат меньше деталей. Для справки, существуют промышленные адаптеры AVR с подключением к USB (ориентировочная цена 50-100 дол.), но в "гаражных условиях" такой прибор не сделаешь.

Сервисные и защитные функции предоставляют пользователю альтернативу выбора. Что лучше, сделать более простой или более сложный адаптер - решать каждому для себя. Критерий истины - это практика. Хотя именно она подсказывает, что лучше иметь не один адаптер, а два - на смену и для перепроверки.

Электрические схемы адаптеров

На сайте разработчика PonyProg http://www.lancos.com/siprogsch.html размещены схемы LPT- и COM-

Таблица 3

Корпус микросхемы	Сигналы интерфейса SPI и выводы МК					
	VCC	GND	$\overline{RES}$	SCK	MOSI	MISO
DIP-8	8	4	1	7	5	6
DIP-20	20	10	1	19	17	18
DIP-28	7	8	1	19	17	18
DIP-40 (1)	40	20	9	8	6	7
DIP-40 (2)	10	11	9	8	6	7

Таблица 4

Контакт разъема IDC-10	Интерфейс		
	Byte Blaster	STK 200	AVR 910
1	SCK	MOSI	MISO
2	GND	VCC	VCC
3	MISO	NC	SCK
4	VCC	GND	MOSI
5	RES	RES	RES
6	NC	GND	GND
7	NC	SCK	NC
8	NC	GND	NC
9	MOSI	MISO	NC
10	GND	GND	NC

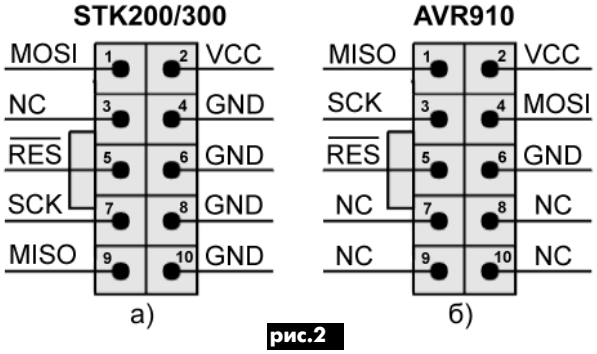

рис.2

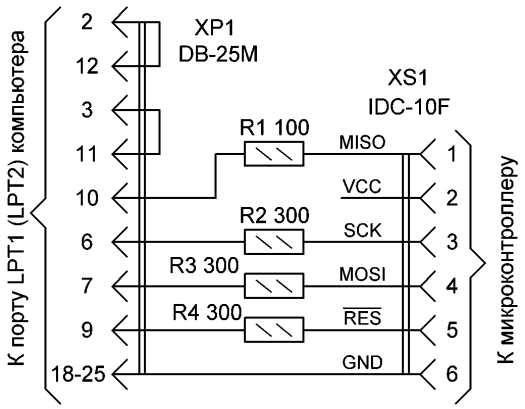

рис.3

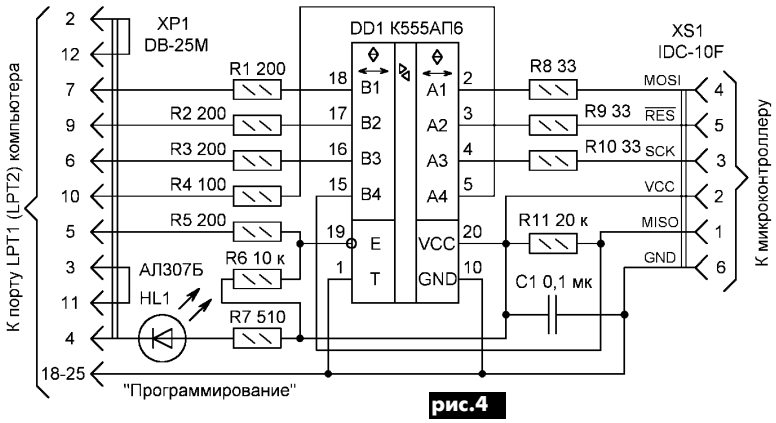

рис.4

адаптеров. Примем их за основу, но с модификациями, повышающими степень защиты и сервис.

Простейший вариант LPT-адаптера (**рис.3**) содержит резисторы R1-R4, которые ограничивают экстратоки и уменьшают "звон" на фронтах импульсных сигналов. Закороченные выводы 2, 12 и 3, 11 разъема XP1 позволяют программе идентифицировать наличие аппаратной части.

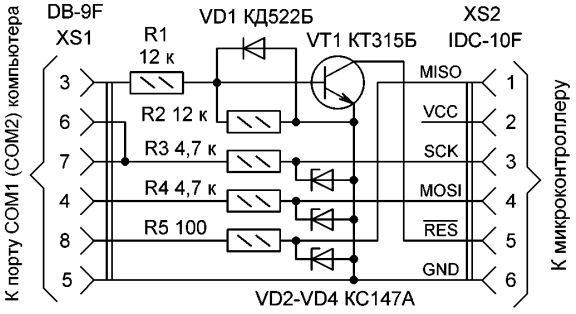

рис.5

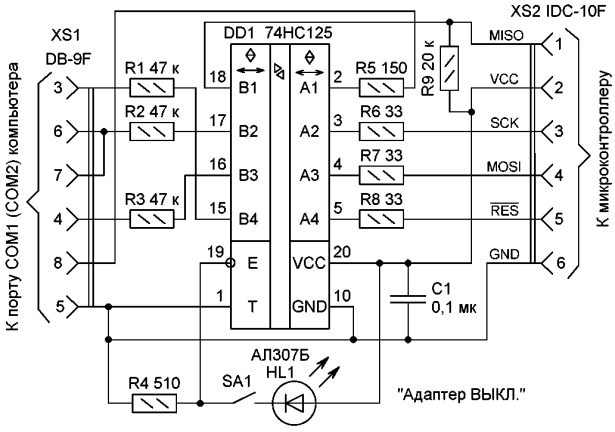

рис.6

Длина соединительного кабеля должна быть как можно меньше. Желательно использовать ленточный кабель, причем информационные сигналы необходимо чередовать с "земляными". Несмотря на простоту, адаптер устойчиво работает с подавляющим большинством компьютеров.

На **рис.4** показана схема более интеллектуального LPT-адаптера, выходы которого автоматически переводятся в высокоимпедансное состояние по окончании программирования. Путь прохождения управляющего сигнала: контакт 5 разъема XP1, резистор R5, вывод 19 микросхемы DD1. Смену уровней лог."0" или лог."1" производит сама программа PonyProg. Она же в процессе программирования выставляет лог."0" на контакте 4 разъема XP1, заставляя светиться индикатор HL1.

Наличие мощных буферных повторителей в микросхеме DD1, с одной стороны, улучшает крутизну фронтов сигналов, а с другой - косвенно защищает компьютер от аварийных перенапряжений. Резисторы R1-R5, R8-R10 улучшают согласование импедансов, а в критической ситуации могут послужить еще и плавкими предохранителями.

Резистор R6 обеспечивает уровень лог."1" на входе E микросхемы DD1 при отстыковке кабеля от LPT-порта. Тем самым выходы буферов переводятся в отключенное состояние от линий интерфейса SPI в МК. Питание микросхемы DD1 (VCC=5 В) подается от платы программируемого устройства. Конденсатор C1 блокировочный. Он должен располагаться вблизи выводов 10, 20 микросхемы DD1.

Адаптер с буферной логической микросхемой хорошо применять в условиях повышенных индустриальных помех. Впервые подобное устройство было разработано фирмой Kanda Systems в противовес отладочным комплектам STK200, STK300, совпадая с ними по распайке выходного разъема. С тех пор подобные адаптеры часто называют Kanda STK200/300.

На **рис.5** показан простой адаптер для COM-порта. Назначение элементов: резисторы R1, R3-R5 ограничивают токи, стабилитроны VD2-VD4 ограничивают напряжение (4,7 В), резистор R2 закрывает транзистор VT1 при отсоединении кабеля от компьютера. Диод VD1 ограничивает напряжение отрицательной полярности, поступающее из COM-порта. В других аналогичных схемах его не устанавливают, надеясь на высокое допустимое напряжение транзисторов BC547 (Philips), которые используются вместо VT1.

Схема неприхотлива в деталях и хорошо зарекомендовала себя на практике, по крайней мере, при управлении от программы PonyProg.

Более сложный COM-адаптер (**рис.6**) включает в себя буферную микросхему DD1, выходы которой переводятся в высокоимпедансное состояние вручную переключателем SA1. Небольшая тонкость: входы микросхемы DD1 не защищены стабилитронами, хотя уровни сигналов COM-порта колеблются от –10 до +10 В. Причина заключается в боль-

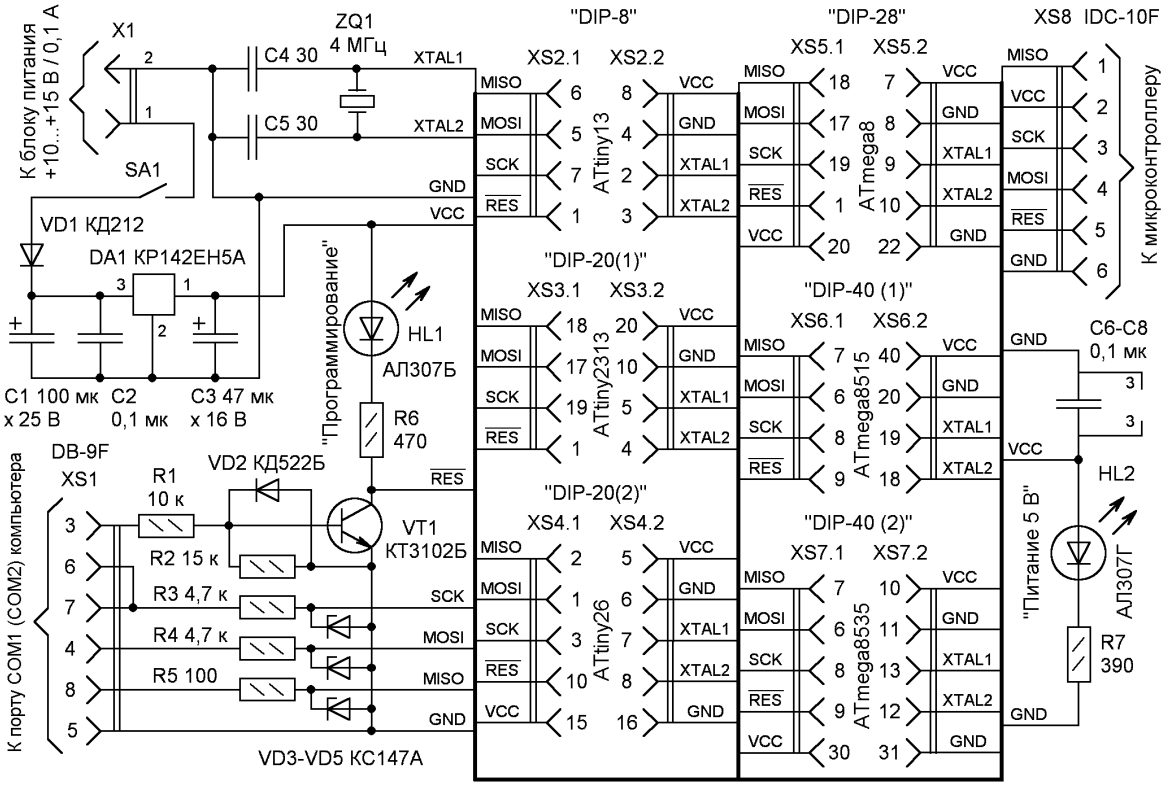

рис.7

шом сопротивлении резисторов R1-R3 и наличии внутренних диодов по входам микросхемы DD1.

Питание 5 В (VCC) подается от платы программируемого устройства. Конденсатор C1 уменьшает импульсные помехи. Резисторы R5-R9 демпфируют выбросы на фронтах импульсов. Если индикатор HL1 погашен, то можно программировать МК. Наличие свечения означает, что микросхему DD1 как будто бы изъяли из панели. Ручное отключение выходов адаптера иногда предпочтительнее программного, например при поисках причин неисправностей.

Приведенные схемы не являются догмой. Допускается изготовить другой вариант адаптера, например, как показано на сайтах http://iron.fire.usi.ru, http://evm.wallst.ru/main/prog, http://www.ln.com.ua/~real/avreal/adapters.html#STK.

Универсальный адаптер

При программировании большого количества разнообразных AVR-контроллеров или при их входном контроле удобно иметь универсальный адаптер с панелями под микросхемы в корпусах DIP-8, DIP-20, DIP-28, DIP-40 (**рис.7**). Все выводы МК с маркировками MISO, MOSI, SCK, RES, VCC, GND соединяются параллельно. Разумеется, одновременно на таком адаптере можно запрограммировать только одну микросхему, вставляемую в панель XS2-XS7. Какие из двух разновидностей 20- или 40-выводных микросхем вставлять в панели "DIP-20(1)", "DIP-20(2)", "DIP-40(1)", "DIP-40(2)", необходимо смотреть по справочным данным DATASHEET на сайте фирмы Atmel http://www.atmel.com.

Кварцевый резонатор ZQ1 совместно с конденсаторами C4, C5 входит в типовую схему включения задающего генератора МК. Некоторые типы МК могут быть запрограммированы и без резонатора, от своего внутреннего RC-генератора, но "кашу маслом не испортишь". Единственное требование - для устойчивой генерации резонатор ZQ1 должен располагаться как можно ближе к выводам XTAL1, XTAL2 МК.

Сопряжение с COM-портом выполнено аналогично схеме, показанной на рис.5. Индикатор HL1 светится только в процессе программирования. Индикатор HL2 указывает на наличие напряжения 5 В, которое формируется стабилизатором DA1. Конденсаторы C1-C3, C6-C8 блокировочные. Диод VD1 защищает адаптер от неверной подачи питающего напряжения. К разъему X1 подключается любой малогабаритный блок питания, например, в корпусе "сетевая вилка", обеспечивающий напряжение 10...15 В при токе 100 мА.

Для радиолюбителей универсальный адаптер представляет альтернативу программирования через разъем ISP, хотя он тоже предусмотрен (XS8). Реальный случай - в миниатюрной конструкции для разъема ISP физически нет места. Или в схеме не хватает входных/выходных линий, и необходимо задействовать MISO, MOSI, SCK в режиме полноценных портов.

Опытный мастер обязательно предусмотрит на плате панель под МК и в любой момент сможет вынуть микросхему из устройства, чтобы запрограммировать внешним программатором. Такой подход гораздо практичнее, чем запаивание AVR "намертво" (вдруг захочется переставить МК в другую конструкцию).

Параллельный программатор

Все ранее рассмотренные адаптеры используют сигналы последовательного интерфейса SPI. Нередко комплекс, состоящий из управляющей программы и адаптера (рис.3-7), называют последовательным программатором AVR.

Если существует последовательный, значит, должен быть и параллельный программатор AVR. На сайте ChaN (http://elm-chan.org/works/avrx/report_e.html, Япония) приведена схема подобного устройства для 20- (**рис.8**) и 8-выводных (**рис.9**) микросхем. Там же

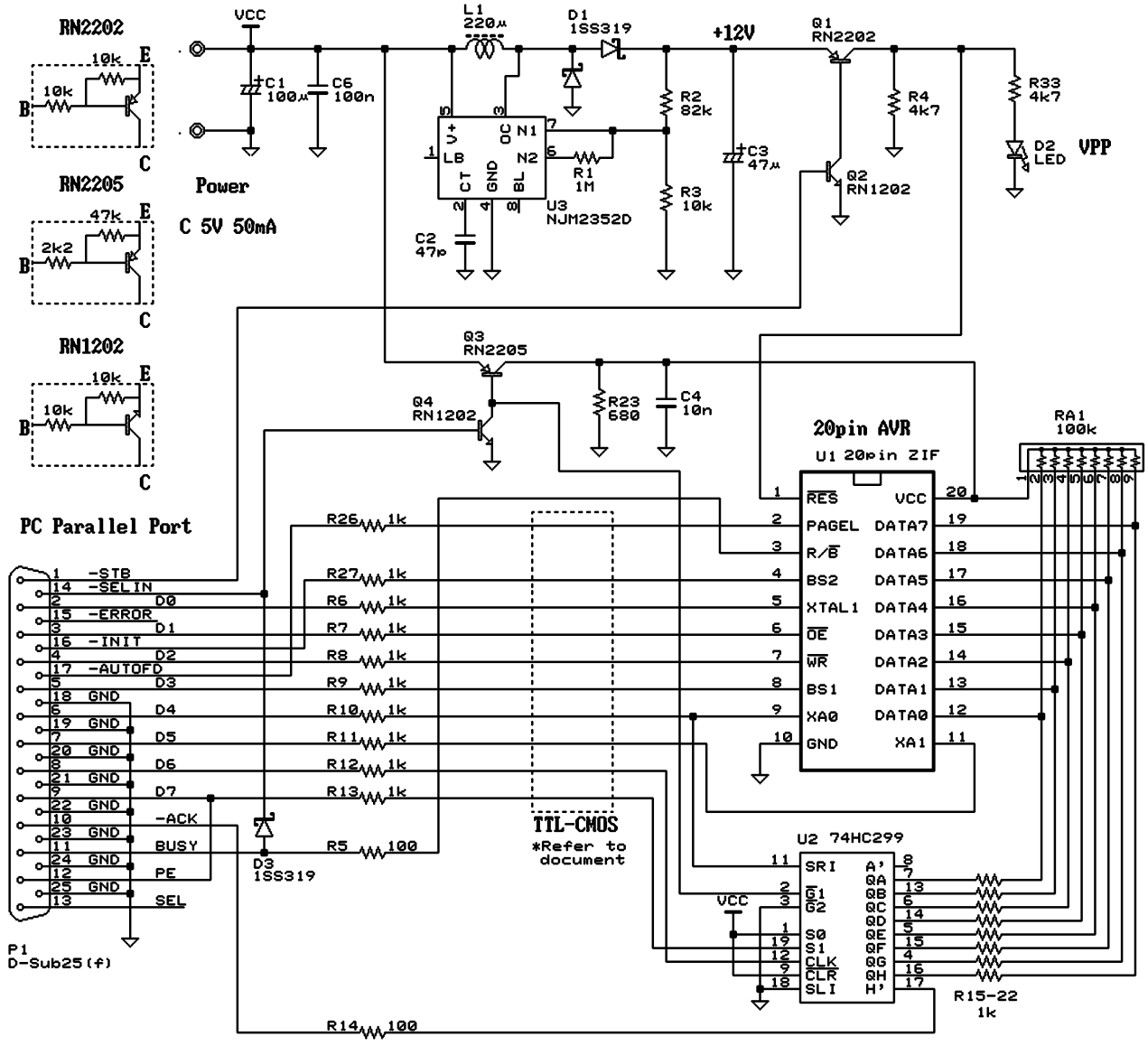

рис.8

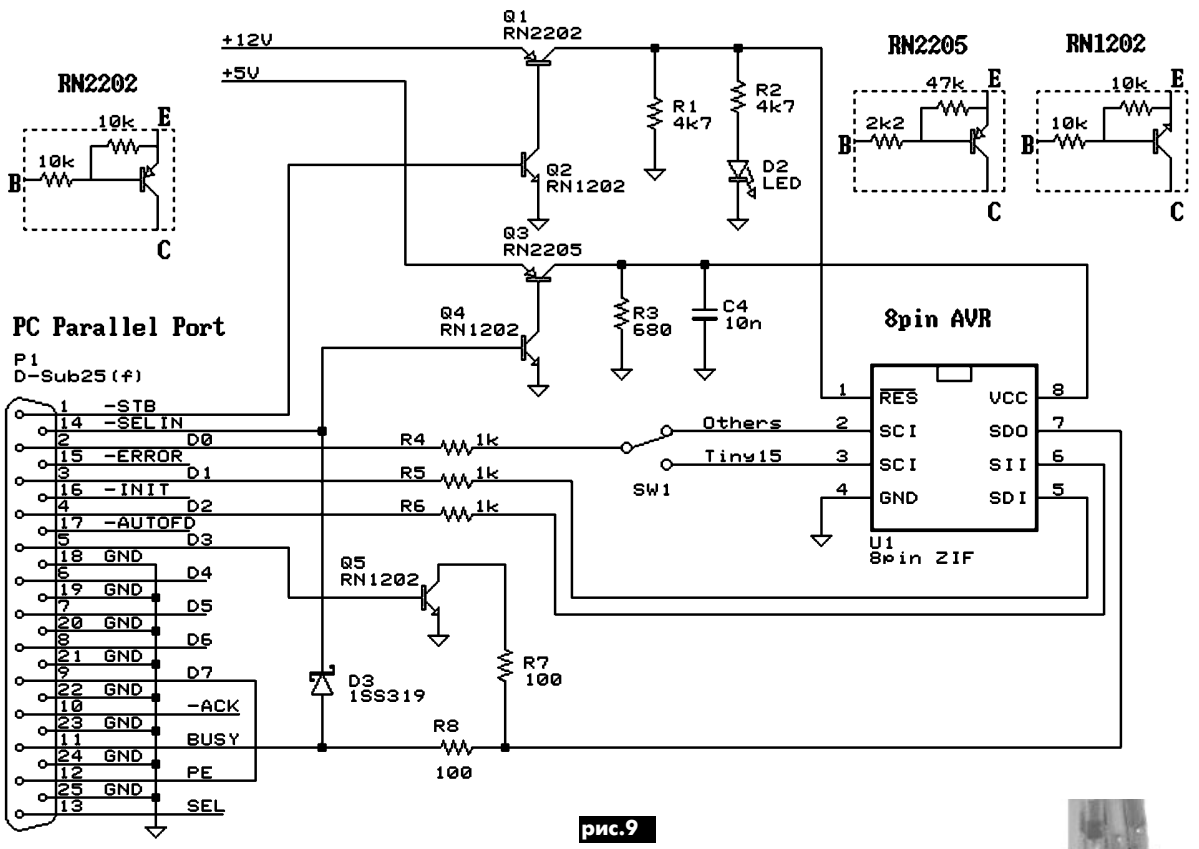

рис.9

выложена управляющая программа под DOS (http://elm-chan.org/works/avrx/avrxtool.zip, 69 Кб) и Win-9x/XP (http://elm-chan.org/works/avrx/avrxtool32.zip, 91 Кб). Микросхемы в корпусах DIP-28, DIP-40 программируются через переходник к 20-контактной панели.

Параллельный программатор позволяет прошивать МК не хуже последовательного, но использует значительно большее число линий связи и повышенное до 12 В напряжение. Как следствие, сложная схемотехника и уникальное программное обеспечение. Главное преимущество параллельного программатора заключается в способности восстановления неосторожно установленных программных битов, переводящих вход RESET МК в дополнительную линию порта. Это обычно случается у новичков при экспериментах на последовательном программаторе. После такой "трепанации" интерфейс ISP отключается, ведь входа сброса RESET как такового уже не существует. Последовательный программатор выдает сообщение о бракованном МК при его абсолютной годности.

Параллельный программатор, как скорая помощь, "вылечивает" МК, позволяя вновь устанавливать программные биты в нужное положение. К счастью, в программе PonyProg опасные биты изначально заблокированы, поэтому даже при ошибочных действиях ввести МК в режим непрограммируемости нельзя.

Если подытожить, то параллельный программатор - это сложная, не совсем обязательная, но довольно полезная в хозяйстве радиолюбителя вещь.

Конструкция и детали

Обычно адаптеры выполняют на отдельной печатной плате, соединенной шлейфом проводов длиной 20...30 см с 10-контактным разъемом ISP и 1,5...2-метровым кабелем с компьютером. Поскольку деталей немного, то используют "макетку" и монтаж тонкими проводами МГТФ-0,1. Плату закрывают корпусом или обвязывают скотчем.

Иногда детали адаптера размещают прямо в разъеме LPT- или COM-порта. Работоспособность такого устройства требует поверки, особенно если не применяются буферные микросхемы. Если на компьютере вместо 9-контактного установлен 25-контактный COM-разъем, то в схемах, показанных на рис.5, 6, меняют распайку выводов, например: 3-2, 4-20, 5-7, 6-6, 7-4, 8-5 (DB9-DB25).

Резисторы, конденсаторы, стабилитроны, диоды - любые малогабаритные. Буферные микросхемы в Интернете рекомендуют из серии HC, например 74HC245 вместо К555АП6 (рис.4). При этом в разрыв цепи VCC устанавливают диод Шотки анодом к контакту 2 розетки XS1. Его назначение - не допустить попадания напряжения 3 В, "просачивающегося" от LPT-порта в МК при отсутствии на последнем питания 5 В. Не рекомендуется замена КР1533АП6, КР1554АП6 ввиду неустойчивого программирования при подключении к цепи GND общего провода осциллографа.

В универсальном адаптере (рис.7), при его интенсивной работе, лучше установить панели XS2-XS6 с нулевым усилием прижатия, хотя это и дорого.

В параллельном программаторе (рис.8) преобразователь DC-DC на микросхеме U3 NMJ2352D можно заменить внешним источником питания 12 В, 50 мА. Его положительный вывод подключают к плюсовой, а отрицательный - к минусовой обкладкам конденсатора C3. Элементы L1, D1, U3, C2, R1-R3 при этом удаляют. Замена микросхемы 74HC299 - КР1554ИР24. "Цифровые" транзисторы RN1202, RN2202, RN2205 можно заменить двумя обычными резисторами и транзисторами КТ3102, КТ3107, в зависимости от типа структуры. Схемы соединения и номиналы резисторов показаны на рис.8, 9.

рис.10

Розетка IDC-10F - однократного применения, о чем часто забывают или не знают. Случается, что при обжимке плоского кабеля в розетку некоторые выводы закорачиваются. Тогда обламывают пластмассовые детали верхней части розетки, аккуратно припаивают провода непосредственно к ее контактам и заливают всю конструкцию компаундом.

Допустимо вообще отказаться от 10-контактных разъемов и применить любые другие имеющиеся соединители с числом контактов не менее 6, например магнитофонные пары СГ-6, СШ-6. Для домашних разработок это не составит проблем, единственное, что воспользоваться в другом месте таким адаптером не удастся.

Еще одно оригинальное решение предложено в Японии: сделать разъем конструктивным и "надевать" его прямо на выводы МК (**рис.10**). Главное, чтобы конструкция не смещалась относительно выводов микросхемы и досрочно не разрушилась, как карточный домик.

Практическое задание. Собрать AVR-адаптер по одной из схем, показанных на рис.3-6, или по аналогичной, допускающей работу с программой PonyProg. По возможности сделать универсальный и параллельный программаторы.

Литература

1. AVR910: In-System Programming. - Atmel, 2000, http://sinbad.narod.ru/isp.htm (русский перевод).

Микроконтроллеры AVR. Ступень 2

С.М. Рюмик, г. Чернигов

Что знаете доброго, того не забывайте,
а чего еще не знаете, тому учитесь.
Владимир Мономах

В предыдущей статье цикла (РА 1/2005) приводились несложные схемы адаптеров для программирования микроконтроллеров (МК) семейства AVR. Надеемся, что один из них успешно собран и готов для дальнейшей работы.

У электронщиков, специализирующихся на проектировании микроконтроллерных устройств, существует термин "быстрый старт". Относится он к случаю, когда надо в короткий срок опробовать в работе новый тип МК и заставить его выполнять простейшую задачу. Цель состоит в том, чтобы, не углубляясь в подробности, освоить технологию программирования и быстро получить конкретный результат. Полное представление, навыки и умения появятся позже в процессе работы.

Обычно фирма-разработчик МК для таких экспериментов поставляет так называемый "стартовый набор" Starter Kit, который представляет собой печатную плату с разъемами, кнопками управления, индикаторами, интерфейсными схемами и, разумеется, с установленным МК. К стартовому набору прилагается управляющая программа и инструкция по программированию.

Фирма Atmel для МК семейства AVR выпустила несколько подобных наборов: STK-100, STK-200, STK-300, STK-500, AVR Embedded Internet, AVR Butterfly. Первые три из них сейчас сняты с производства и заменены STK-500 (**рис.1**). Это наиболее подходящий для работы универсальный комплекс, который обслуживает подавляющее большинство типов МК AVR. Единственный его недостаток – стоимость $80-100.

Можно ли самостоятельно собрать упрощенный аналог набора "быстрого старта", ориентируясь на подручные средства, минимальную цену и бесплатно распространяемые программы? Ответ положительный, при этом вся задача разбивается на несколько этапов:

сборка адаптера программатора;

освоение работы с компилятором;

изготовление макетной платы "быстрого старта";

компиляция демо-программы, зашивка кодов в МК и наблюдение практического результата.

Те, кто выполнил домашнее задание из "Ступени 1", могут считать первый этап пройденным. Остальные этапы будут рассматриваться далее.

Компилятор языка Си для AVR

Какие только языки программирования не используют для составления AVR-программ! В Интернете имеются ссылки на Ассемблер, Ada, Basic, Си, Forth и даже Pascal. Но на практике чаще всего используют Ассемблер и Си. Первый из них предпочтителен для создания критичных во времени процедур, а второй – для быстрой разработки программ. Если учесть, что в архитектуру AVR изначально были заложены принципы оптимизации Си-процедур, то альтернативы этому алгоритмическому языку нет.

В **таблице** приведен перечень наиболее популярных Си-компиляторов. Самым мощным из них считается компилятор фирмы IAR Systems. Это немудрено, ведь ее сотрудники в середине 90-х годов принимали непосредственное участие в разработке системы команд AVR.

Из других компиляторов больше на слуху ICCAVR и CodeVision. Их полные версии требуют оплаты, что для начинающих радиолюбителей не в подъем. Пользоваться демо-версиями можно лишь на первых порах для решения ограниченного круга задач с урезанными функциями. На практике же поступают по-славянски изобретательно, тем или иным способом добывая патчи к программам. Интернет-форумы так и пестрят просьбами: "Намыльте лекарство для вижн AVR", что в переводе означает: "Вышлите мне, пожалуйста, по электронной почте пароль к Си-компилятору CodeVision AVR".

Но зачем ломиться в открытую дверь, нарушая чьи-то авторские права? Существуют бесплатно распространяемые Си-компиляторы, и возможности у них колоссальные. Если выбирать из двух представленных в таблице freeware-компиляторов, то предпочтение следует отдать WinAVR. Во-первых, он поддерживает больше типов МК, во-вторых, его автором является коллектив профессиональных программистов, в-третьих, он солиднее по выполняемым функциям и, наконец, его версии постоянно обновляются.

Осторожный читатель вправе заметить, что "бесплатный сыр бывает только в мышеловке". Рассеять сомнения поможет теоретическая подкованность.

Классификация программного обеспечения

Абсолютно бесплатных программ не бывает. Каждая из них имеет свою цену, другое дело, какую именно. Например, за скачивание с Интернета freeware-программы надо заплатить деньги провайдеру. Для перезаписи бесплатного софта с компьютера на компьютер надо приобрести дискету или CD. И даже если кто-то подарит вам безвозмездно программу, то время установки ее на компьютер можно пересчитать через износ вычислительной техники.

Информация в Интернете делится на три большие группы: бесплатная, условно-бесплатная и платная.

Freeware (сокращение от FREE softWARE) – бесплатные программы с возможностью многократного тиражирования и неограниченным сроком действия. Правила хорошего тона предписывают не изменять название программы, а также указывать авторов при ссылке на нее.

Shareware (сокращение от SHARE softWARE) – условно-бесплатные программы. На компьютерном жаргоне их называют "шароварными". Авторы подобных программ вводят определенные ограничения. Например, действует лимит на

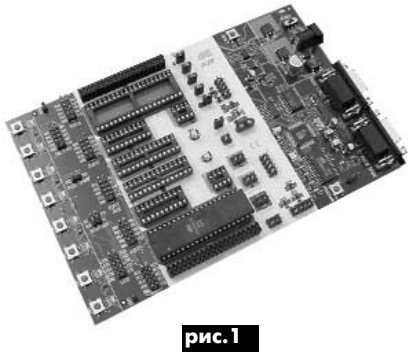

рис.1

Компилятор	Фирма, страна	Сайт разработчика	Версия (цена)	Файл
C-AVR	SPJ Systems, Индия	http://www.spjsystems.com/cavr.htm	1.0.0 ($149-225)	Демо 1,4 Мб
CodeVision AVR	HP InfoTech, Румыния	http://www.hpinfotech.ro/html/cvavr.htm	1.24.4a ($90-150)	Демо 3,5 Мб
CrossWorks	Rowley Associates, Англия	http://www.rowley.co.uk/avr/index.htm	1.0 ($790)	7,9 Мб, 30 дней
IAR Embedded Workbench	IAR Systems, Швеция	http://www.iar.com	3.20c (более $1000)	58 Мб, 30 дней
ICCAVR	ImageCraft, США	http://www.imagecraft.com/software	7 ($199-749)	6,2 Мб, 45 дней
SmallC for AVR	Ron Kreymborg, Австралия	http://www.jennaron.com.au/smallc/smallc.html	2.4.7, бесплатно	126 Кб
WinAVR (GCC)	Eric Weddington (коллективный проект)	http://winavr.sourceforge.net	20040720, бесплатно	12,9 Мб

количество дней работы, на число функций, запусков. Бывает так, что к выходному файлу дописывается ремарка "Не зарегистрирован!". Чтобы заменить demo- или trial-версию полноценной программой, необходимо выслать автору денежное вознаграждение.

Commercial – коммерческие платные программы, распространяемые за деньги, и, как правило, немалые. Взамен фирма гарантирует техническую поддержку, ответы на вопросы, своевременное обновление версий. За примерами далеко ходить не надо – это все Windows, MS Office, AutoCAD, Adobe Photoshop и многое другое. Коммерческие продукты защищены авторскими правами, и, по идее, их нелегальные копии нельзя использовать в производстве собственных изделий или для оказания платных услуг.

Внутри каждой группы программ существуют подразделы. Для freeware действует следующая классификация.

Adware (ad-sponsored software, bannerware) – программа, в которой содержится баннерная реклама, автоматически скачиваемая с Интернета. Эта программа, с точки зрения пользователя, является бесплатной. Однако ее автор получает определенные дивиденды от рекламодателей.

Donationware (Beer-ware, cakeware) – занимают промежуточное положение между freeware и shareware. От первого типа они отличаются включением в текст программы просьбы автора прислать ему "кто, сколько может" или угостить чем-нибудь вкусным, а от второго типа – отсутствием реально действующих ограничений.

Spyware (trackware, Big Brotherware) – программа-шпион, которая по запросу извне может отсылать ее автору информацию о параметрах компьютера, адресах e-mail, паролях. Часто такие программы автоматически попадают в компьютер при посещении Интернет-магазинов и рекламных сайтов. Подобным способом коммерсанты пытаются выяснить вкусы своих посетителей, их количество и географию размещения. Если дело ограничивается только этим, то spyware абсолютно безвредны. Неприятность грозит со стороны хакеров, когда какая-нибудь простенькая бесплатная программка, обеспечивающая дозвон до провайдера, может коварно утаскивать ваши пароли и отсылать по Сети своему создателю.

Postcardware – для запуска такой программы требуется пройти регистрацию. Для этого надо отправить автору почтовую карточку, открытку или электронное письмо, а взамен получить пароль. Цель автора – узнать, насколько популярна его программа, и какому контингенту пользователей она интересна.

Причины, побуждающие разрабатывать freeware-программы

Можно ли считать всех авторов бесплатного софта альтруистами по убеждению? Ознакомившись с причинами, побуждающими людей "раздавать" программы, каждый может сделать свои выводы [1].

1. Программа получилась небольшой по выполняемым функциям, интересной только узкому кругу лиц или потребовала на ее создание мало времени. Уважающий себя программист не станет подражать "собаке на сене" и отдаст свой скромный труд на всеобщее пользование.

2. Бесплатная программа может стать хорошим "портфолио" (англ. портфель) при поиске работы. Это своеобразная визитная карточка и аттестат зрелости. К примеру, открыл страничку в Интернете, поместил на ней программу, которая вызвала интерес. Со временем ссылки на страничку появляются на ведущих поисковиках мира, программу награждают призами на конкурсах. Как следствие, автор начинает получать заказы на разработку коммерческих программ.

3. На базе бесплатной программы могут создаваться виртуальные клубы по интересам. Появление новых знакомств, интересный форум, свежие идеи, консультации, взаимовыручка – вот "капитал" автора. Очень часто организуются интернациональные группы программистов, которые поочередно совершенствуют программный продукт, переводят его на разные языки.

4. Попытка распространять вирусы под видом бесплатных программ. К сожалению, такие случаи встречаются на практике, поэтому каждую новую скачанную через Интернет программу следует проверять "антивирусом" до ее запуска на компьютере, а не после...

5. Бесплатная программа может являться пробной альфа- или бета-версией коммерческого продукта, но с еще не до конца исправленными ошибками. Часто они не фатальные и серьезно не мешают в работе.

6. Некоторые фирмы разрабатывают одновременно freeware- и commercial-программы, например, первую из них для индивидуальных пользователей, а вторую – для корпоративных сетей. При этом бесплатная версия служит рекламой, на ней отрабатываются новые алгоритмы и устраняются ошибки за счет читательских отзывов.

7. Размещение программы в Интернете может служить защитой от плагиата. В "свободное плавание" отправляют версию программы, в которой четко фиксируется основная идея алгоритма, указывается авторство и дата. При обнаружении плагиата, о нем предупреждаются системные администраторы и поисковые сайты.

8. В разряд бесплатных могут переводиться нормально работающие, но устаревшие версии коммерческих продуктов. Расчет на психологию потребителя: если бесплатная версия понравится, то с большой долей вероятности могут купить и ее улучшенный вариант.

9. Некоторые программисты при временном отсутствии работы по специальности, чтобы не терять форму, работают над свободно распространяемыми проектами. При этом отзывы, критика и пожелания пользователей являются мощным стимулом к изучению новых приемов программирования, что, безусловно, пригодится в будущем.

10. Создание бесплатной программы может служить альтернативным ответом одних программистов другим. Принцип по-спортивному простой: "...и мы умеем не хуже". Классический пример – это операционная система Linux, задуманная как некоммерческая альтернатива Windows.

Существует целый класс бесплатных программ, авторы которых сознательно распространяют их без ограничений, руководствуясь принципами GNU. К этой категории принадлежит и WinAVR. Поскольку данное направление включает в себя целое философское учение, то имеет смысл рассмотреть его подробнее.

Принципы GNU

В 1984 году группа дипломированных специалистов во главе с Ричардом Столлменом (Richard Stallman) основала Проект GNU (GNU Project). Это свободная в распространении Unix-подобная операционная система, включающая в себя множество свободных подпрограмм. Название "GNU" расшифровывается как рекурсивный акроним словосочетания "GNU's Not Unix" ("GNU – это не Unix"). На эмблеме Проекта (**рис.2**) изображен стилизованный шарж антилопы гну, созвучный со словом "gnu".

рис.2

Со временем начали разрабатывать и другие программы на базовом принципе "свобода программного обеспечения" (free software), который означает право пользователя свободно запускать, копировать, распространять, изучать, изменять и улучшать. В английском языке слово "free" переводится двояко – "бесплатный" и "свободный". В системе GNU используется второе понятие, которое имеет четыре разновидности:

свобода 0 – возможность запускать программу в любых целях;

свобода 1 – возможность изучения логики работы программы и адаптации ее к своим нуждам;

свобода 2 – возможность свободного распространения копий программы;

свобода 3 – возможность улучшать программу и публиковать свои улучшения без уведомления кого-либо.

Программа считается свободной от ограничений при выполнении всех четырех свобод. Необходимым условием осуществления свободы 1 и 3 является доступ к исходным текстам программы. При выполнении свободы 2 пользователь вправе взимать плату за копирование программы (и только!), хотя может распространять ее бесплатно.

Сторонники идеи GNU выработали своеобразный "кодекс чести", в котором нет места "пиратству и плагиату". Существует даже список слов и выражений, не приемлемых в сообществе. Принципам GNU может соответствовать любая программа любого жанра любой операционной системы. Знак авторского права "копирайт" рекомендуется заменять "копилефтом", единственным ограничением которого является запрет накладывать какие-либо ограничения при распространении программы.

Если внимательно проанализировать философию GNU, то ее основная идея заключается в коллективном накоплении банка программ и алгоритмов, которым мог бы воспользоваться любой желающий.

А как же быть с оплатой труда программистов? Принципы GNU не запрещают брать деньги за услуги по тиражированию свободных программ и за услуги по адаптации программного обеспечения для нужд конкретного потребителя. Не секрет, что в большинстве случаев оплачивают программы те люди, которые не умеют их составлять самостоятельно. Если человек с трудом отличает "фортран" от "ситроена", то ему не поможет знание исходного кода. А вот у программистов между собой не должно быть тайн.

С изложенной теорией можно соглашаться или спорить, но она реально существует и поддерживается Free Software Foundation, Inc. (http://www.gnu.org, США, Бостон, русская страница http://www.gnu.org/home.ru.html). Более того, наблюдается тенденция к увеличению числа приверженцев GNU, особенно после подключения к Интернету программистов из стран с невысоким достатком.

Освоение WinAVR

Первое знакомство с WinAVR обычно рекомендуют начинать с изучения англоязычного файла помощи http://winavr.sourceforge.net/download/install_config_WinAVR.pdf, 297 Кб, автор Colin O'Flynn. Не уменьшая значение этого документа, можно предложить другой подход - более простой, краткий и логически не менее понятный.

Первоначально следует скачать со страницы http://sourceforge.net/projects/winavr последнюю по времени версию пакета WinAVR (по состоянию на январь 2005 г. это 20040720). Дата версии зашифрована в названии файла по принципу год-месяц-число, в частности, http://ovh.dl.sourceforge.net/sourceforge/winavr/WinAVR-20040720-install.exe означает 20 июля 2004 г. Объем файла почти 13 Мб, поддерживается докачка с 10 равноценных зеркал из разных уголков планеты. В Интернете потребуется 2...3 часа работы при крейсерской скорости скачивания 5...7 Мб/час на городских телефонных линиях.

После запуска файла WinAVR-20040720-install.exe на выполнение, предлагается выбор языка инсталляции, среди прочих имеются русский и украинский. О том, что WinAVR (произносится "whenever") является свободно распространяемым пакетом, свидетельствует лицензия GNU GPL (General Public License). По умолчанию программа будет установлена в папку C:\WinAVR, занимая объем 69,9 Мб. Менять этот путь не следует в целях облегчения идентификации дальнейших действий.

По окончании инсталляции в системный файл "autoexec.bat" Win9x автоматически вводятся две новые строки путей поиска, а на рабочем столе Windows-9x/2000/XP создаются ярлыки к 7 разным инструментам пакета: Avr Insight, Avr-libs, GNU manuals, MFile, Programmers Notepad, Readme, TkInfo. Для начала лучше переместить их в отдельно созданную папку или удалить со стола в корзину. Доступ к ним все равно остается через меню "Пуск – Программы – WinAVR".

На этом шаге многие программисты в недоумении прекращают работу с компилятором. Дело в том, что единого запускающего файла не видно, а хаотический вызов на выполнение отдельных рабочих инструментов ни к чему толком не приводит. Разбираться же в объемных англоязычных файлах помощи не у каждого хватает терпения. Очевидно, с этими негативными моментами связано не столь широкое распространение WinAVR, как других Си-компиляторов.

А ларчик открывается просто: текстовый редактор Programmers Notepad (PN), автор Simon Steele, и является по совместительству главной оболочкой Си-компилятора. Пусть никого не смущает поддержка в нем разных языков программирования, начиная от Visual Basic до Java, и наличие отдельного авторского сайта http://www.pnotepad.org. Надо помнить, что пакет WinAVR собран из отдельных разработок, каждая из которых может иметь самостоятельное применение. К слову сказать, в будущих версиях WinAVR разработчики планируют ввести многоязыковую поддержку и отдельный графический интерфейс для комплексного создания новых проектов.

Итак, пакет WinAVR успешно установлен на компьютер. Теперь очередь поискать среди его файлов примеры составления Си-программ. В папке C:\WinAVR\examples\demo\ находится одна из них под названием "demo.c". Поскольку распространяется она на условиях Beer-ware (разновидность freeware), то нет ограничений против русификации комментариев и более плотного форматирования текста, как показано в **листинге**. Автора программы, немца Йорга Вунша (Joerg Wunsch), остается лишь поблагодарить за работу и гостеприимно пригласить в древний град Чернигов на кружку пенистого напитка...

Разбираться в тонкостях построения данной Си-программы нет надобности. Это сложный для начинающих пример, скорее образец того, как делают люди. Главное, что не требуется самому писать первую тестовую программу, в которую по неопытности можно внести казусные ошибки.

Составление электрической схемы для "быстрого старта"

Программа "demo.c", приведенная в листинге, хоть и демонстрационная, но вполне работоспособна на макете. Подсказки для построения схемы устройства разбросаны прямо по ее тексту, не пропустить бы. В частности, устройство должно содержать один светодиод, подключенный между выводом OC1 (OC1A) МК и общим проводом GND. Светодиод, по замыслу разработчика программы, должен периодически изменять свою яркость от максимума до минимума, используя широтно-импульсное регулирование длительности.

Идентифицировать, где находится вывод OC1 (OC1A), можно, только зная название МК. Судя по заголовку Си-программы, она универсальна и без изменений может работать с AT90S2313, 2333, 4414, 4433, 8515, 8535, ATmega8, 64,128,163.

Но как же программист указывает компилятору тип МК? Для этого к каждой Си-программе в WinAVR прилагается так называемый make-файл. Он должен находиться в той же папке, что и Си-программа. В нашем случае, действительно, в папке C:\WinAVR\examples\demo\ расположен файл без расширения под названием "makefile" длиной 1770 байтов. Если просмотреть его содержимое любым текстовым редактором, то в третьей сверху строке будет видна надпись: "MCU_TARGET = at90s2313", из чего следует однозначный вывод – программа "demo.c" будет откомпилирована под МК AT90S2313.

Справочные данные на микросхему находятся в DATASHEET http://www.atmel.com/dyn/resources/prod_doc-

Листинг

```
/* "THE BEER-WARE LICENSE" (Revision 42). Если вы считаете
 * эту программу полезной для себя, то при встрече вы можете
 * угостить меня пивом. Joerg Wunsch <joerg@FreeBSD.ORG>
 * ----------------------------------------------------------
 * Демонстрационная программа. Индикатором служит светодиод,
 * подключенный между выводами OC1/OC1A и GND. Яркость све-
 * чения управляется ШИМ (то увеличивается, то уменьшается).
 * =demo.c, v 1.1.2.2  2004/05/25  08:55:24  joerg_wunsch=*/
#include <inttypes.h>              /*Библиотека типов данных*/
#include <avr/io.h>                /*Библиотека ввода-вывода*/
#include <avr/interrupt.h>          /*Библиотека прерываний*/
#include <avr/signal.h>              /*Библиотека сигналов*/
#if defined(__AVR_AT90S2313__)  /*Если обнаружен AT90S2313*/
#  define OC1 PB3                 /*Линии PB3 назначено имя OC1*/
#  define OCR OCR1        /*Регистру OCR1 назначено имя OCR*/
#  define DDROC DDRB    /*Регистру DDRB назначено имя DDR0C*/
#elif defined(__AVR_AT90S2333__)||defined(__AVR_AT90S4433__)
#  define OC1 PB1                 /*Линии PB1 назначено имя OC1*/
#  define DDROC DDRB    /*Регистру DDRB назначено имя DDR0C*/
#  define OCR OCR1        /*Регистру OCR1 назначено имя OCR*/
#elif defined(__AVR_AT90S4414__)||defined(__AVR_AT90S8515__)
    ||defined(__AVR_AT90S4434__)||defined(__AVR_AT90S8535__)
    ||defined(__AVR_ATmega163__)  /*Если обнаружены эти МК*/
#  define OC1 PD5                 /*Линии PD5 назначено имя OC1*/
#  define DDROC DDRD    /*Регистру DDRD назначено имя DDR0C*/
#  define OCR OCR1A      /*Регистру OCR1A назначено имя OCR*/
#elif defined(__AVR_ATmega8__) /*Если обнаружен МК ATmega8*/
#  define OC1 PB1                 /*Линии PB1 назначено имя OC1*/
#  define DDROC DDRB    /*Регистру DDRB назначено имя DDR0C*/
#  define OCR OCR1A      /*Регистру OCR1A назначено имя OCR*/
#  define PWM10 WGM10 /*Регистру WGM10 назначено имя PWM10*/
#  define PWM11 WGM11 /*Регистру WGM11 назначено имя PWM11*/
#elif defined(__AVR_ATmega64__)||defined(__AVR_ATmega128__)
#  define OC1 PB5                 /*Линии PB5 назначено имя OC1*/
#  define DDROC DDRB    /*Регистру DDRB назначено имя DDR0C*/
#  define OCR OCR1A      /*Регистру OCR1A назначено имя OCR*/
#  define PWM10 WGM10 /*Регистру WGM10 назначено имя PWM10*/
#  define PWM11 WGM11 /*Регистру WGM11 назначено имя PWM11*/
#else                    /*Если обнаружен неизвестный тип МК*/
#  error "Don't know what kind of MCU you are compiling for"
#endif             /*Досрочный выход при неизвестном типе МК*/
#if defined(COM11)
#  define XCOM11 COM11   /*Рег. COM11 назначено имя XCOM11*/
#elif defined(COM1A1)
#  define XCOM11 COM1A1 /*Рег. COM1A1 назначено имя XCOM11*/
#else
#  error "need either COM1A1 or COM11"
#endif   /*Досрочный выход при отсутствии COM1A1 или COM11*/
enum { UP, DOWN };             /*Переменные UP, DOWN для ШИМ*/
volatile uint16_t pwm;                  /*Переменная 16 бит*/
volatile uint8_t direction;              /*Переменная 8 бит*/
/*---------------Прерывание по таймеру-1------------------*/
SIGNAL(SIG_OVERFLOW1) /*Прерывание по переполнению таймера*/
{ switch (direction) /*Переключатель смены направления ШИМ*/
  { case UP:                            /*Увеличение яркости*/
      if (++pwm == 1023)       /*Если максимальное значение*/
        direction = DOWN;              /*Смена направления*/
        break;
    case DOWN:                          /*Уменьшение яркости*/
      if (--pwm == 0)           /*Если минимальное значение*/
        direction = UP;                /*Смена направления*/
        break;
  }
  OCR = pwm; /*Занесение очередного значения в регистр ШИМ*/
}
/*-----------------Начальная инициализация------------------*/
void ioinit (void)
{ TCCR1A = _BV (PWM10) | _BV (PWM11) | _BV (XCOM11);
  TCCR1B = _BV (CS10);                  /*Настройка таймера-1*/
  OCR = 0;                    /*Нулевое значение в регистре ШИМ*/
  DDROC = _BV (OC1); /*Настройка линии светодиода на выход*/
  timer_enable_int (_BV (TOIE1));   /*Разрешение таймера-1*/
  sei ();                             /*Разрешение прерываний*/
}
/*-----------------Основная программа----------------------*/
int main (void)
{ ioinit ();           /*Начальная инициализация регистров*/
  for (;;);              /*Бесконечный цикл с прерываниями*/
  return (0);
}                                     /*Окончание программы*/
```

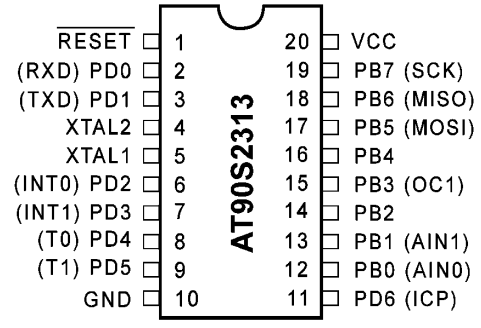

рис.3

uments/DOC0839.PDF, 1,6 Мб. Из этого файла берется ее условное обозначение (**рис.3**) и определяется, что название "OC1" относится к линии порта PB3 и выводу 15 МК.

В DATASHEET приведена типовая схема включения микросхемы и указаны номера выводов MISO, MOSI, SCK, $\overline{RES}$, необходимые для подключения адаптера программатора. Теперь нетрудно составить схему "быстрого старта" (**рис.4**). Тактирование МК осуществляют кварцевый резонатор ZQ1 и два конденсатора C2, C3. Керамический конденсатор C1 должен располагаться в непосредственной близости от выводов питания VCC, GND. Микросхему DD1 устанавливают в 20-выводную панель. Разброс частоты ZQ1 – 3...5 МГц.

Разъем XP1 может иметь два разных варианта включения, что указано в цоколевке контактов в скобках. Соответственно под адаптер стандарта AVR910 или STK200 (см. "Ступень 1"). Ответной частью в любом случае будет 10-контактная розетка IDC-10F. Если адаптер имеет буферную микросхему, то питание на нее поступает через линию VCC разъема XP1.

Практическая работа

Все составляющие проекта разложены по полочкам, настало время собрать их вместе в определенной последовательности.

1. Запустить на выполнение текстовый редактор PN: "Пуск – Программы – WinAVR – Programmers Notepad". Закрыть окно New, чтобы оно не мешало работе. Загрузить файл демонстрационной Си-программы: "File – Open – <выбрать путь C:\WinAVR\examples\demo\demo.c> – Открыть". Произвести из начального меню компиляцию программы: "Tools – make all", дождаться надписи в нижнем окне экрана "Process Exit Code: 0" (**рис.5**). После этого в папке с Си-программой появятся 10 новых файлов, среди которых hex-файл demo.hex с кодами прошивки МК.

2. Запустить на выполнение программу PonyProg версии 2.06c: "Пуск – Программы – PonyProg – PonyProg2000" (считается, что эта программа ранее была установлена на компьютер при выполнении заданий из "Ступени 1"). При первом

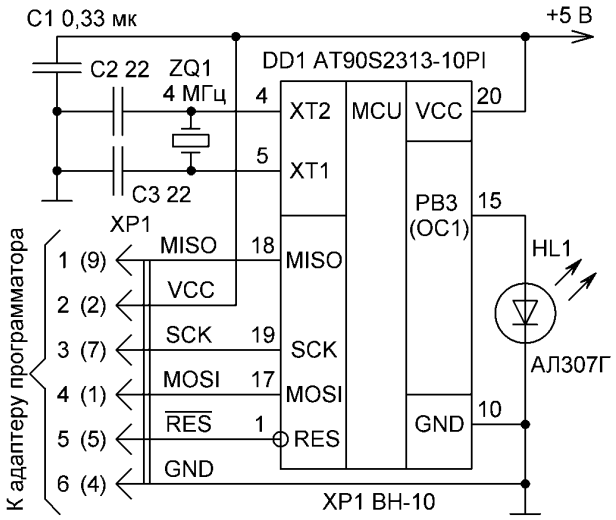

рис.4

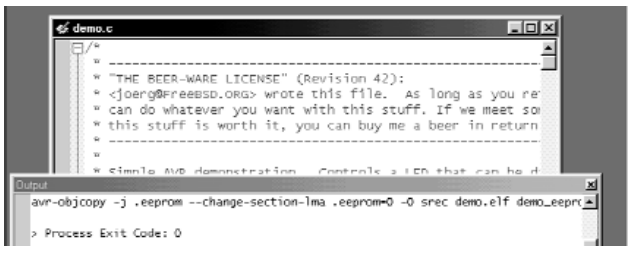

рис.5

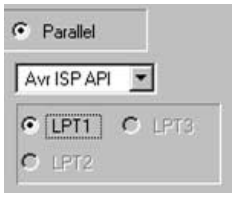

рис.6

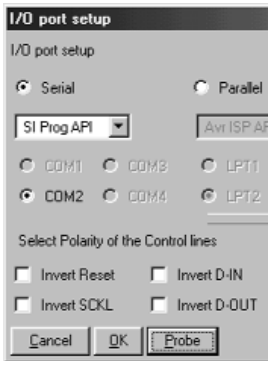

рис.7

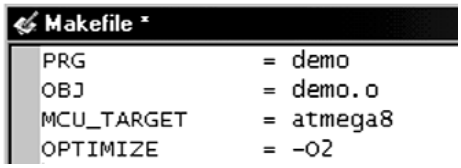

рис.9

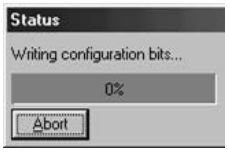

рис.10

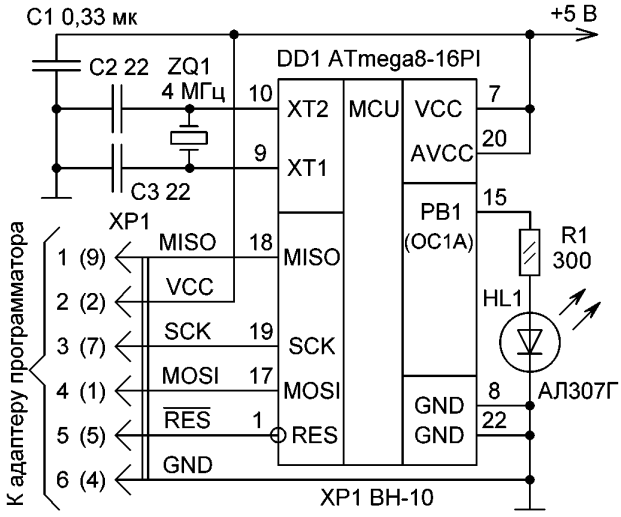

рис.8

запуске нажать три раза OK и произвести калибровку компьютера: "Setup – Calibration – Yes – OK". При повторных запусках нажать один раз OK и калибровку не проводить. Выбрать тип микросхемы: "Device – AVRmicro – AT90S2313". Указать последовательность действий: "Command – Program Options – <установить флажки только возле пунктов Reload Files, Erase, Write Program memory (FLASH)> – OK".

3. Подключить адаптер программатора с одной стороны к LPT- или COM-порту компьютера, а с другой стороны – к разъему XP1 макетной платы (рис.4). Подать на МК DD1 питание 5 В.

4. В начальном меню PonyProg выбрать опции: "Setup – Interface Setup", затем для LPT-адаптера поставить точки возле пунктов Parallel, LPT1 или LPT2, выбрать строку Avr ISP API (**рис.6**). Для COM-адаптера поставить точки возле пунктов Serial, COM1 или COM2, выбрать строку SI ProgAPI (**рис.7**). Для COM-адаптера с буферной микросхемой 74HC125 ("Ступень 1", рис.5) дополнительно поставить точку возле Invert Reset. Проверить исправность аппаратной части адаптера нажатием кнопки "Probe", в ответ должно появиться сообщение "Test OK" (исправно) или "Test Filed" (найти и устранить неисправность в адаптере). Для возврата в начальное меню нажать два раза OK.

5. Загрузить в PonyProg скомпилированный в пункте 1 hex-файл: "File – Open Program (FLASH) File – <выбрать строку с типом файла *.hex> – <выбрать путь к файлу C:\WinAVR\examples\demo\demo.hex> – Открыть". Далее нажать одновременно клавиши <Ctrl> и <P>. Через 10 с на экране монитора должно появиться сообщение: "Programming Sucsessfull" (успешное программирование).

Если макетная плата собрана без ошибок и МК исправен, то после программирования автоматически должны начаться циклы плавной засветки и гашения светодиода HL1 с периодом около 1 с. При этом адаптер программатора отсоединять от разъема XP1 не надо.

Модельные эксперименты

Теперь представьте, что необходимо изменить текст Си-программы и увеличить в 8 раз период мигания светодиода HL1. Естественно, на компьютере должны быть постоянно запущены программы PonyProg и PN, а пользователь должен уметь "мышью" оперативно переключаться между ними.

В верхнем окне PN в программе "demo.c" найти строку номер 90: "TCCR1B = _BV (CS10);" и откорректировать ее следующим образом: "TCCR1B = _BV (CS11);". Откомпилировать полученную Си-программу: "Tools – make all", дождаться в нижнем окне надписи "Process Exit Code: 0". Переключиться на программу PonyProg, нажать одновременно клавиши <Ctrl> и <P> и через 10 с увидеть по светодиоду HL1 на макетной плате результаты своей работы.

Не правда ли, как просто и быстро! Подобные изменения в AVR-программах в зависимости от типа МК можно производить от 1000 до 10000 раз.

Теперь более сложная задача: поменять тип МК с AT90S2313 на ATmega8. В начале следует идентифицировать вывод под названием OC1A согласно DATASHEET http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf, 2,4 Мб. Составляется схема включения МК (**рис.8**). Микросхема DD1 устанавливается в 28-выводную панель, остальные элементы используются из схемы рис.4. Резистор R1 ограничивает ток через светодиод HL1, поскольку выходной порт у ATmega8 в несколько раз мощнее, чем у AT90S2313.

Далее необходимо откорректировать make-файл, для чего в PN выполнить действия: "File – Open – <выбрать путь C:\WinAVR\examples\demo\makefile> – Открыть". В его третьей сверху строке заменить текст "MCU_TARGET = at90s2313" на "MCU_TARGET = atmega8" (**рис.9**). Откомпилировать полученную программу: "Tools – make all", дождаться надписи "Process Exit Code: 0".

Переключиться на PonyProg, сменить тип МК: "Device – AVRmicro – ATmega8". Задать конфигурацию фьюз-битов (чего не требовалось для AT90S2313): "Command – Security and Configuration Bits – Clear all – Write" (**рис.10**). Выйти в начальное меню PonyProg и, при поданном на устройство питании 5 В, одновременно нажать клавиши <Ctrl> и <P>. Через 30 с на макетной плате наблюдать плавное засвечивание и гашение светодиода HL1. Задача выполнена.

При первых экспериментах некоторые действия в процессе программирования придется принять на веру, не задавая вопросов. Например, как создать make-файл, какие фьюз-биты в PonyProg надо ставить для разных типов МК, по какому принципу менять содержимое операторов в Си-программе. Ответы на эти и другие вопросы будут рассмотрены в следующих статьях цикла.

Практическое задание. Опробовать методику ввода и компиляции программы "demo.c", а также методику программирования разных типов МК через PonyProg и адаптер.

Литература

1. Рюмик С.М. С Интернета по нитке...//Радиомир. Ваш компьютер. – 2002. – №8. – С.9–12.

Микроконтроллеры AVR. Ступень 3

С.М. Рюмик, г. Чернигов

Dictum factum
Сказано – сделано (лат.)

Первые опыты по программированию микроконтроллеров (МК) семейства AVR обычно проходят без особых затруднений. Теперь наступает черед самостоятельных действий по синтезу схем при условии осознанного выбора режимов и элементов.

МК семейства AVR относят к высокопроизводительным 8-разрядным устройствам с RISC-архитектурой. Быстродействие МК принято оценивать в миллионах операций в секунду, сокращенно MIPS (**M**illions of **I**nstructions **P**er **S**econd). В качестве эталона операции принимается самая короткая пересылка 8 битов данных из одного регистра в другой.

Все МК с ядром AVR, благодаря конвейерной RISC-архитектуре, способны выполнять одну пересылку за один такт работы процессора. Получается, что быстродействие напрямую зависит от тактовой частоты, которая, как правило, определяется кварцевым резонатором. Например, самой "высокочастотной" среди AVR общего применения является ATtiny13, допускающая работу при 24 МГц. Следовательно, ее максимальное быстродействие 24 MIPS. По сравнению с микросхемой AT89C2051 семейства MCS-51, это в 12 раз больше. Для ориентира, рекорд среди специализированных AVR принадлежит USB-контроллеру AT76C713 – 48 MIPS.

Базовый элемент для экспериментов

Методика быстрого изучения МК основывается на предположении, что достаточно освоить базовую микросхему, чтобы затем по шаблону составлять программы к другим ее разновидностям. Для AVR коварную шутку сыграла многовариантность моделей. Что ни МК, то новые нюансы. Названия регистров, флагов, фьюзов, хотя и похожи, но чуть-чуть отличаются. А для выяснения этого "чуть-чуть" приходится изучать документы в сотню и более страниц. К счастью, общего у разных типов AVR ("classic", "tiny", "mega") больше, чем различий.

Два-три года назад обучение AVR рекомендовали начинать с AT90S2313. Действительно, этот недорогой 20-выводный МК хорошо вписывается в несложные радиолюбительские конструкции. Более того, у него даже цоколевка и назначение выводов совпадает с AT89C2051 (за исключением инверсии сигнала RESET). Однако серия "classic" уже снята с производства по веским техническим причинам. И хотя в продаже эти микросхемы будут находиться еще длительное время, надо смотреть в будущее, в перспективу.

В **табл.1** приведены варианты замены для двух поколений AVR. Водораздел между ними проходит в технологических нормах, соответственно 0,5 и 0,35 мкм. Другой показатель – время разработки: до 2001-2002 гг. и после. Кроме того, в новых МК исправлены конструктивные ошибки, добавлены функциональные возможности, увеличена тактовая частота до 16...24 МГц и число перезаписей FLASH-ПЗУ с 1000 до 10000. Как следствие, повысилась долговременная устойчивость работы в условиях сильных помех.

Замена МК идет не один к одному. На сайте фирмы Atmel http://www.atmel.com в разделе технической документации имеются файлы с однотипным названием "Replacing by..." и "Migrating between...", в которых указаны пути преодоления различий.

Из МК второго поколения наиболее полным набором функций обладает микросхема ATmega128, на которой можно показать все доступные аппаратные и программные приемы. Однако ее 64-выводный TQFP-корпус и высокая цена не годятся для любительских экспериментов. Все остальные МК серий "mega", "tiny" можно рассматривать в первом приближении как усеченные версии ATmega128 (меньше памяти, функций, выводов). Кстати, последнее обстоятельство часто является определяющим при выборе МК.

Различают микросхемы в DIP-корпусе с малым (8), средним (20-28) и большим (40) числом выводов. Среди имеющихся в продаже "осьминогов" заслуживают внимание 8-выводные ATtiny13, ATtiny15L, но они хороши для компактных интеллектуальных датчиков, а не для учебы. Характерно, что слово "tiny" (произносится "тайни") переводится с английского как "крошечный".

Среди 40-выводных "сороконожек" наиболее дешевые и ходовые – это ATmega8515, 8535, 162. Их удобно применять в технически сложных приборах. Однако на макетной плате с двумя светодиодами они смотрятся громоздко.

Остаются МК среднего диапазона. В частности, перспективно применение ATtiny2313, который идет на замену AT90S2313, но его еще нет в продаже. Как компромисс, предлагается выбрать в качестве базовой микросхему ATmega8. Она компактна по габаритам (28 выводов), доступна по цене ($3-3,5), имеет все функции ATmega128 за исключением интерфейса JTAG.

Типовая схема включения AVR

Для нормального функционирования любого МК требуется выполнить ряд условий: подать питание, обеспечить генерацию тактовых импульсов, организовать начальный сброс, подключить периферию к входам-выходам. Исходная информация по всем перечисленным вопросам содержится в главном техническом документе, так называемом DATASHEET ("data" – данные, "sheet" – листок). Различают краткую (summary) и полную (complete) версии "дейташита", отличающиеся объемом информации. Например, для ATmega8 summary-файл http://www.atmel.com/dyn/resources/prod_documents/2486S.pdf имеет длину 199 Кб и содержит 21 страницу текста, а complete-файл http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf – длину 2,4 Мб и 305 страниц текста.

Если подходить к изучению AVR основательно, то рано или поздно придется распечатать на бумаге все страницы полного DATASHEET. Труд не пропадет даром, ведь построение документов на фирме Atmel унифицировано для всех AVR. Многие разделы написаны "один к одному", что пригодится при смене типа МК.

Назначение выводов ATmega8 (**рис.1**): VCC, AVCC – питание, GND – общий, PB0-PB7 – линии порта B, PC0-PC6 – линии порта C, PD0-PD7 – линии порта D, AREF – выход внутреннего источника напряжения 1,3 В. В скобках возле обозначения контактов указаны альтернативные функции, переключаемые программно. По количеству их может быть не одна, как в AT89C2051, а целых две, на что указывает наклонная разделительная черта. Название и назначение функций будет расшифровано по мере их изучения.

Особенности подачи питания

В каталогах встречаются два типа микросхем: ATmega8-16 и ATmega8L-8. Первая из них допускает питание 4,5...5,5 В при тактовой частоте 0...16 МГц, вторая – соответственно 2,7...5,5 В при 0...8 МГц. Это не означает, что ATmega8 выйдет из строя при подаче питания 3 В. Более того, в таком режиме она успешно работает с различными кварцевыми резонаторами. Однако нельзя гарантировать устойчивый запуск МК при крайних значениях температур, да и ток потребления будет выше, чем у ATmega8L. Сказываются технологические различия в изготовлении.

Вывод – если требуется максимальное быстродействие, то надо ставить ATmega8 и повышать тактовую частоту до 8...16 МГц при питании 5 В. Если главнее всего экономичность устройства, то лучше применить ATmega8L и понизить частоту, питание. В дальнейшем большинство приводимых схем будут рассчитаны на универсальный диапазон тактовых частот 4...8 МГц и питание 3...5 В.

Предусмотрительный пользователь приобретет для экспериментов обе разновидности микросхемы (по стоимости они примерно одинако-

Таблица 1

Снятые с производства AVR первого поколения	Замена AVR второго поколения
AT90S1200, AT90S2313	ATtiny2313
AT90S2323, AT90S2343	ATtiny25
AT90S4414, AT90S8515	ATmega8515
AT90S4433	ATmega8
AT90S8535	ATmega8535
ATmega103	ATmega128
ATmega161	ATmega162
ATmega163	ATmega16
ATmega323	ATmega32
ATtiny11, ATtiny12	ATtiny13

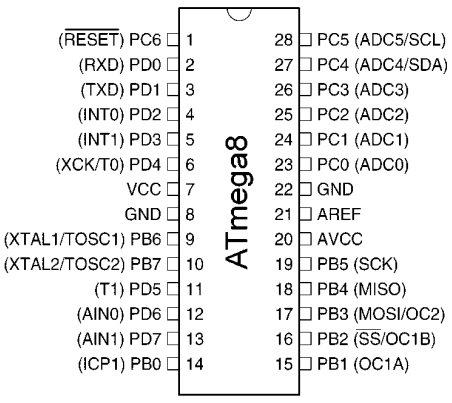

рис.1

вы) – для взаимной перепроверки и на крайние начальные условия. Если ориентироваться на DIP-корпус, то буквы в конце названия МК должны быть PI (температурный диапазон -40...+85°C), PU (то же, но в корпусе, не содержащем свинец), PC (0...+70°C).

ATmega8 имеет двойное питание: "цифровое" VCC, GND (вывод 8) и "аналоговое" AVCC, GND (вывод 22). В стандартном включении (**рис.2**), когда на входы и выходы подаются уровни лог."1" и "0", обе пары соединяют параллельно. Точнее, физически закорачивают только цепи VCC, AVCC, поскольку GND-выводы 8 и 22 уже замкнуты внутри микросхемы через сопротивление 0,7 Ом.

Конденсаторы C1, C2 обязательно керамические, например, K10-17. Располагаться они должны максимально близко от "своих" по схеме выводов. Такая конфигурация рекомендуется в [1] для микросхем с двойным питанием. Если в МК нет вывода AVCC, в частности, ATmega8515, то вместо двух ставят один конденсатор. Более того, на практике так поступают и с ATmega8.

Если в МК используется встроенный 6-канальный АЦП, то для уменьшения помех применяют последовательный LC-фильтр по выводу AVCC. Если АЦП не нужен, то все равно вывод AVCC соединяют с VCC коротким проводом.

Для снижения уровня излучаемых помех рекомендуется применять общий LC-фильтр по питанию (**рис.3**). С такой необходимостью могут столкнуться разработчики промышленной аппаратуры при выполнении норм электромагнитной совместимости.

Как показывает практика, МК семейства AVR "не любят" высокого питающего напряжения (выше 6 В). Кроме того, регламентируется максимальный ток через выводы GND, VCC, который не должен превышать 200 мА. В качестве защиты удобно применять маломощный стабилизатор 78L05 и параллельно включенный сапрессор напряжением 5,6 В.

Система начального сброса

Для установки внутренних регистров МК в исходное состояние необходимо произвести начальный сброс. ATmega8 располагает следующими возможностями (**рис.4**):

внутренний автоматический сброс по достижению напряжения питания 1,4...2,3 В;

сброс от внутреннего детектора просадок питающего напряжения Brown-Out;

внешний сброс уровнем лог."0" на выводе 1 /RESET;

сброс от внутреннего таймера Watchdog при остановке работы процессора.

Если напряжение питания стабильно во времени и подается скачком на МК, то внешние элементы для сброса вообще не нужны (см. "Ступень 2"). Обнуление происходит автоматически узлом Power-On. Подобная схема подходит для лабораторного макетирования и домашних самоделок, но в промышленной автоматике может давать сбои при импульсных помехах по питанию и при его слишком плавном нарастании (спаде).

Для AVR первого поколения фирма Atmel рекомендует на вывод RESET устанавливать стандартную цепочку "резистор-конденсатор-диод" (4,7 кОм – 0,01 мкФ – 1N4148) или отдельный супервизор питания [1, 2]. Альтернативный вариант – резервная аккумуляторная батарея напряжением 4,8 В, постоянно подзаряжаемая через элементы R1, VD2 (**рис.5**).

МК AVR второго поколения имеют улучшенную защиту от кратковременных (brown-out) и полных (black-out) просадок питания. Теперь в каждую микросхему встроен переработанный узел детектора пониженного напряжения BOD (**B**rown-**O**ut **D**etector). Детектор анализирует напряжение питания VCC и вырабатывает сигнал сброса при достижении одного из двух порогов: 2,7 или 4 В. Имеется гистерезис 0,1 В, что позволяет четко обслуживать приборы с "подсевшими" батареями питания.

Режим включения-выключения детектора, а также значения порогов задаются при программировании двух фьюзов: BODEN (**BOD EN**able) и BODLEVEL (**табл.2**). Первое знакомство с фьюзами состоялось в "Ступени 2". Для дальнейшей работы надо знать, что фьюзы, или биты конфигурации, – это некоторые ячейки в FLASH-ПЗУ МК, которые можно многократно прошивать в "1" или "0" с помощью программатора. В процессе штатной работы их значения изменить нельзя.

Почему фьюзы так хорошо защищены от вмешательства извне? Это необходимо для устранения конфликтных ситуаций и устойчивого запуска МК. Например, чтобы при сбое в программе случайно не установился порог срабатывания детектора (4 В) выше чем напряжение питания (3 В).

В ответственных случаях вводят кроме внутреннего детектора еще и внешний супервизор питания (**рис.6**). Логика рассуждений простая: копеечный супервизор "кашу маслом не испортит", зато спасет в непредвиденной ситуации. Резисторы R1, R2 показаны пунктиром. Первый из них нужен, если микросхема DA1 не имеет выхода с открытым коллектором. Второй – шунтирует высокоомный интегральный резистор Rн, повышая помехоустойчивость. Диод VDo внутренний. Он защищает вход RES от случайной подачи отрицательного напряжения.

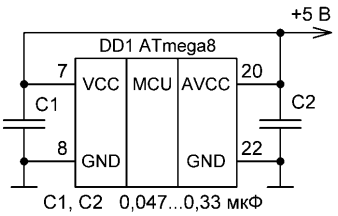

рис.2

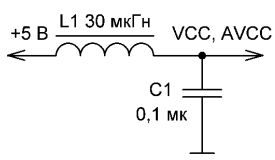

рис.3

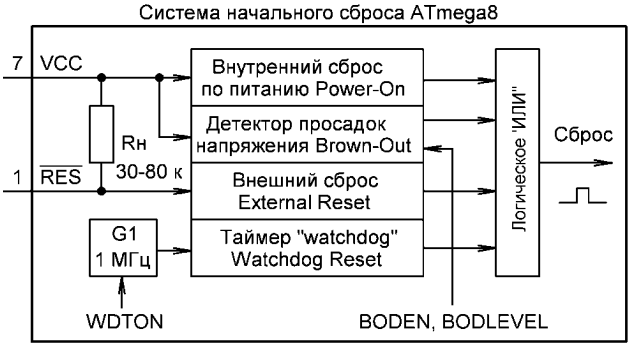

рис.4

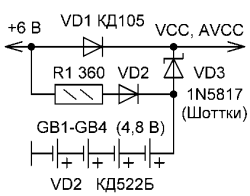

рис.5

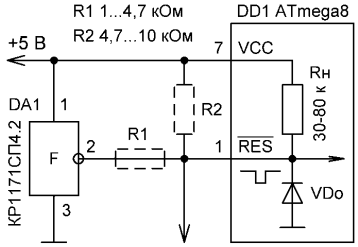

рис.6

Таблица 2

Функция	BODEN	BODLEVEL	Микросхема
Детектор BOD отключен	1	0 или 1	ATmega8, 8L
Порог срабатывания 2,7 В	0	1	ATmega8L
Порог срабатывания 4 В	0	0	ATmega8, 8L

Надобность в супервизоре может возникнуть при нестандартном номинале питания, например, 4,2 В, что очень близко к порогу срабатывания 4 В. Внутренний детектор при этом блокируют установкой в "1" фьюза BODEN.

С фьюзами связано и еще одна техническая новинка, отсутствующая в AT89C2051, – встроенный "watchdog". Если фьюз WDTON (**W**atch **D**og **T**imer **ON**) запрограммирован (т.е. WDTON=0, **табл.3**), то запускается специальный таймер, который анализирует состояние процессора. В случае его "зависания", таймер через регулируемое время 17 мс...2,2 с выдает сигнал начального сброса.

Почему бы "watchdog" не сделать постоянным элементом МК, тогда и фьюз включать не надо? Дело в том, что внутренний генератор G1 (рис.4), от которого работает "watchdog", потребляет энергию. При отключенном фьюзе (WDTON=1) генератор обесточивается, что важно, например, для батарейного питания. Сравните, 50 мкА в спящем режиме при наличии и 3 мкА при отсутствии Watchdog.

Осталось рассмотреть внешний сброс, который активизируется, если на входе RESET в течение более 1,5 мкс удерживается лог."0". Кнопку сброса SB1 обычно подключают вместе со стандартной RC-цепочкой (**рис.7**). Поскольку на вывод 1 МК приходят также сигналы от разъема ISP, то контакты кнопки SB1 при программировании должны всегда находиться в разомкнутом состоянии.

Интересное наблюдение. Во время сброса ток потребления ATmega8 не только не уменьшается, но и почти в два раза увеличивается (с 5...15 мА до 20...25 мА). Этот момент надо учитывать при ограниченных энергоресурсах.

Во многих схемах, приводимых в Интернете, элементы SB1, C1 отсутствуют. Тем не менее, и в этом случае можно осуществить кнопочный сброс, но не снаружи, а изнутри МК программным путем. Кнопка сброса подключается к одной из линий портов и при ее нажатии устанавливается определенный флаг, приводящий к выполнению программы заново.

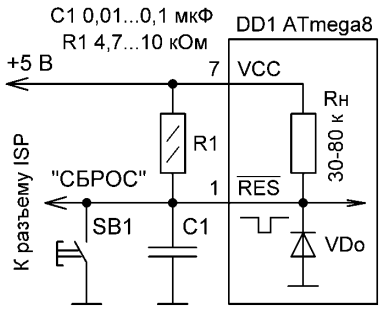

рис.7

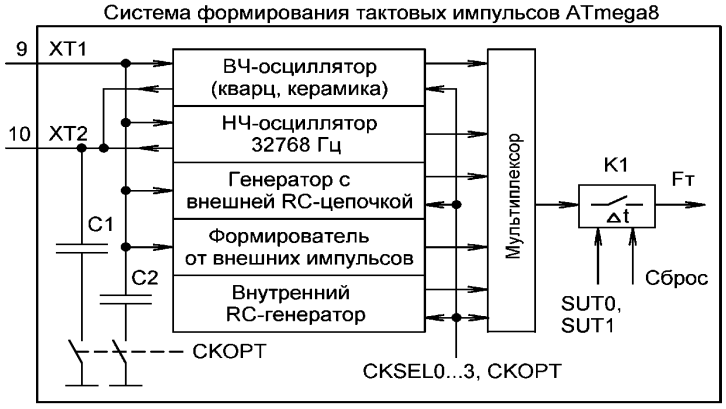

рис.8

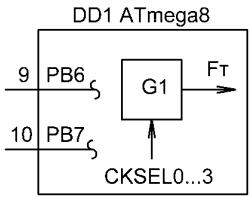

рис.9

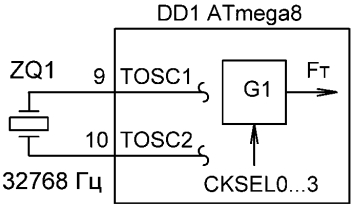

рис.10

Таблица 3

Функция	WDTON	Примечание
"Watchdog" отключен	1	Фьюз не запрограммирован
"Watchdog" включен	0	Фьюз запрограммирован

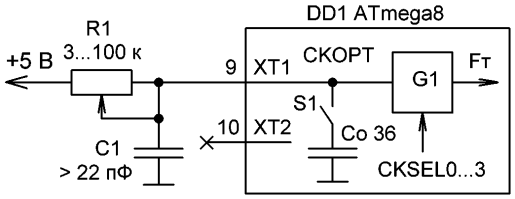

рис.11

Система формирования тактовой частоты

Синхронизация тактовых сигналов в МК семейства AVR происходит более сложно и многовариантно, чем в MCS-51. С одной стороны это хорошо, поскольку появляется богатый выбор возможностей, с другой – легко ошибиться и привести годный контроллер в "безмолвное" состояние.

Структурная схема системы формирования тактовой частоты в ATmega8 показана на **рис.8**. Внутри МК по входам XTAL1, XTAL2 (в дальнейшем сокращенно XT1, XT2) условно показаны 5 отдельных генераторных узлов, мультиплексор и ключ K1, срабатывающий с задержкой во времени. Коммутацию режимов работы задают фьюзы: CKSEL (**C**loc**K** **SEL**ect), CKOPT (**C**loc**K** **OPT**ion) и SUT (**S**tart-**U**p **T**ime). Они выступают здесь как перемычки (англ. fuse – "предохранитель"), которые МК не может самостоятельно изменить.

Информация о фьюзах в DATASHEET размещена довольно хаотично, что вызывает определенные затруднения при поиске. Если из всех возможных комбинаций, ответственных за формирование тактовой частоты, оставить лишь те, которые обеспечивают задержку выхода на режим 65 мс (SUT0, SUT1), то получится компактная **табл.4**. Для сведения, эта задержка в DATASHEET носит название "time-out" и прибавляется к сигналу сброса, позволяя высокодобротным кварцевым резонаторам быстрее стабилизировать частоту генерации. Ключ задержки K1 на рис.8 показан условно, для лучшего понимания процессов. Опытные пользователи могут фьюзами уменьшить "time-out" до 4 мс, но они должны четко представлять, какую выгоду из этого получат. А вот проблем с нестабильностью запуска и помехоустойчивостью может прибавиться.

ATmega8 допускает следующие режимы тактирования:

от высокочастотного кварцевого резонатора 0,9...16 МГц;

от керамического резонатора 0,4...0,9 МГц со встроенными конденсаторами;

от низкочастотного кварцевого резонатора 32768 Гц;

от внутреннего RC-генератора 1; 2; 4; 8 МГц;

от внешней RC-цепочки 0,4...12 МГц;

от внешнего импульсного генератора 0...16 МГц.

Обилие режимов сродни восточному базару – выбирай, что приглянется.

Общие рекомендации. При невысоких требованиях к стабильности временных интервалов проще всего использовать внутренний, программно перестраиваемый RC-генератор (**рис.9**). В **табл.5** приведены его параметры, из чего вытекает, что лучшая стабильность получается при частоте 1 МГц. Случайно или нет, но именно на эту частоту настраивают ATmega8 при выпуске с завода-изготовителя. Выводы XT1, XT2 превращаются в обычные линии порта ввода-вывода PB6, PB7, по ним можно передавать или принимать логические сигналы. Кроме того, имеется возможность установкой программного флага сделать еще одно превращение – использовать выводы XT1, XT2 для синхронизации внутреннего таймера от резонатора ZQ1 (вторая альтернативная функция TOSC1, TOSC2, **рис.10**).

Если необходима плавная подстройка частоты, то применяют генератор с частотозадающей цепью в виде внешних RC-элементов (**рис.11**). Корпус переменного резистора R1 должен быть соединен с общим проводом, чтобы уменьшить влияние емкости тела. Ключ S1 управляется фьюзом CKOPT. Если он будет запрограммирован (CKOPT=0), то конденсатор C1 вообще можно удалить из схемы, поскольку его функцию будет выполнять внутренняя интегральная емкость Co=36 пФ. Формула для расчета частоты генерации Fт[МГц]=1000 / (3*R[кОм]*C[пФ]). Например, Fт=0,46 МГц при R1=33 кОм, C1=22 пФ. В реальности, значение будет отличаться от расчетного из-за емкости монтажа и входной емкости микросхемы.

Среднюю и высокую стабильность обеспечивают соответственно керамический (**рис.12**) и кварцевый резонаторы (**рис.13**). Первый из них быстрее выходит на режим и не требует внешних конденсаторов, а второй обладает наивысшей стабильностью. Интересно назначение фьюза CKOPT, который выполняет функцию своеобразного регулятора амплитуды. Если он запрограммирован (равен "0"), то сигнал на выводе XT2 будет меняться "rail-to-rail", т.е. от 0 до VCC, обеспечивая ответвление сигнала Fт через инвертор DD1. Если CKOPT=1, то амплитуда сигнала резко уменьшается. Одновременно снижаются и паразитные ВЧ-излучения, что важно для обеспечения требований по электромагнитной совместимости.

При работе от аккумуляторной батареи и невысоких требованиях к быстродействию, выгодно применять часовой кварцевый резонатор 32768 Гц (**рис.14**). На такой низкой частоте собственный ток потребления ATmega8 составляет всего лишь 80 мкА! Программирование фьюза CKOPT приводит к добавлению двух внутренних конденсаторов Co. Их наличие позволяет подключать резонатор ZQ1 напрямую к выводам XT1, XT2. Допускается использовать резонаторы и на другие низкие частоты 30...300 кГц, но входные цепи оптимизированы именно под частоту 32768 Гц, как наиболее распространенную.

Если в схеме устройства уже имеется отдельный кварцевый генератор, то его логическими сигналами можно тактировать МК (**рис.15**), сэкономив при этом финансы на резонаторе. Резистор R1 нужен, если DD1 является ТТЛ-микросхемой. Он подтягивает уровень лог."1" к шине питания, поскольку по входу XT1 (в отличие от других линий портов) требуется обеспечить повышенное напряжение 0,8*VCC.

В условиях сильных промышленных помех, когда прибор установлен в непосредственной близости от "искрящих" цепей двигателей, входы XT1, XT2 и подключенные к ним элементы могут служить транзитным путем для наводок. Чтобы устранить сбои, рекомендуется конденсаторы C1, C2 (рис.13) устанавливать вблизи выводов XT1, XT2, а их земляные обкладки подключать прямо к общему выводу 8 МК отдельными короткими проводниками. Кроме того, корпус кварцевого резонатора ZQ1 припаивают коротким проводом к цепи GND, а на печатной плате вокруг него и конденсаторов проводят экранирующий контур.

Устранить сбои иногда помогает замена резонатора отдельным покупным кварцевым генератором в металлическом DIP-корпусе, напри-

мер, из серии JCO-8 фирмы Jauch. Это эффективное, но дорогое решение. Ввиду того, что на выходе генератора присутствуют логические уровни, то фьюзами устанавливают режим работы МК от внешних входных импульсов.

Подключение входов-выходов

С точки зрения электронщика, знание внутренней структуры портов входа-выхода является обязательным условием при освоении любого МК. Порты AVR принципиально отличаются от портов MCS-51. Чтобы не запутаться, даже названия им придуманы разные: P0, P1, P2, P3 у MCS-51 и PA, PB, PC, PD у AVR. Для справки, многовыводные ATmega128 имеют еще порты PE, PF, PG, а у маловыводных ATtiny в наличии только несколько линий порта PB.

Изучать структуру портов легче в сравнении MCS-51 - AVR. На **рис.16, 17** приведены упрощенные схемы организации входов и выходов МК AVR для линии PB1. Схемы остальных линий портов будут аналогичны за исключением PC6 (верхний диод VDo отсутствует). Управление входами-выходами в ATmega8 производится через программно доступные регистры: PORTn, DDRn, PINn, где n=B, C, D (**табл.6**).

Первое отличие – ключ S1, которым можно подключать или отключать нагрузочный резистор Rн по каждому из входов. Выполнен он на основе интегрального МОП-транзистора, поэтому прозвонить его омметром нельзя. В MCS-51 резистор Rн на одних входах имеется, а на других отсутствует.

Второе отличие – входной триггер Шмитта U1 и линия задержки U2. Они предназначены для повышения помехоустойчивости и отсечения коротких импульсных помех.

Таблица 4

Режим генерации	CKSEL3	CKSEL2	CKSEL1	CKSEL0	CKOPT	SUT1	SUT0	Диапазон частот
ВЧ кварцевый резонатор	1	1	0	1	1	0 (BOD), 1 (65 мс)	1	0,9-3 МГц
	1	1	1	1	1		1	3-8 МГц
	1	1	1	1	0		1	1-16 МГц
Керамический резонатор	1	0	1	1	1	0 (65 мс)	0	0,4-0,9 МГц
	1	0	1	0	1	1 (BOD)	0	
НЧ кварцевый резонатор	1	0	0	1	0 (36 пФ)	1	0	32768 Гц
	1	0	0	1	1 (0 пФ)	1	0	
Внешняя RC-цепочка	0	1	0	1	0 (36 пФ), 1 (0 пФ)	0 (BOD), 1 (65 мс)	0	0,1-0,9 МГц
	0	1	1	0			0	0,9-3 МГц
	0	1	1	1			0	3-8 МГц
	1	0	0	0			0	8-12 МГц
Внутренний RC-генератор	0	0	0	1	1	0 (BOD), 1 (65 мс)	0	1 МГц
	0	0	1	0	1		0	2 МГц
	0	0	1	1	1		0	4 МГц
	0	1	0	0	1		0	8 МГц
Внешние входные импульсы	0	0	0	0	0 (36пФ), 1 (0 пФ)	0 (BOD), 1 (65 мс)	0	0-16 МГц

Условные обозначения: BOD – включен детектор пониженного напряжения, 65 мс – включена задержка "time-out", 36 пФ – включена емкость Co по входу XT1 (XT2), 0 пФ – выключена емкость Co

Таблица 5

Частота RC-генератора, МГц	Диапазон перестройки, МГц	Стабильность по питанию, кГц / 1 В	Стабильность по температуре, Гц / 1°C
1	0,55…1,8	20	350
2	1,1…3,5	45	800
4	2,2…7,7	80	1700
8	4,5…14	230	4000

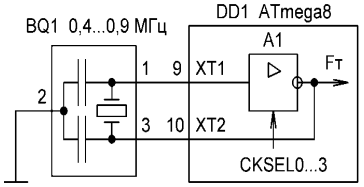

рис.12

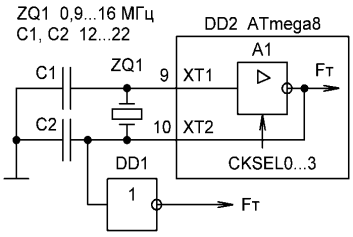

рис.13

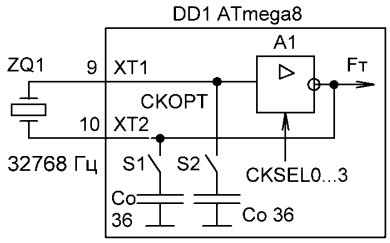

рис.14

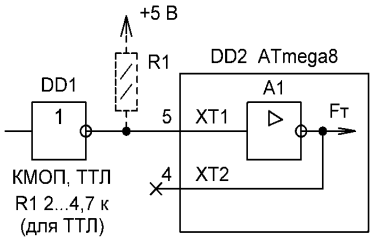

рис.15

Таблица 6

DDB0…7, DDC0…6, DDD0…7	PORTB0…7, PORTC0…6, PORTD0…7	Вход-выход	Примечание
0	0	Вход	Z-состояние
0	1	Вход	Rн=20…50к
1	0	Выход	Лог."0"
1	1	Выход	Лог."1"

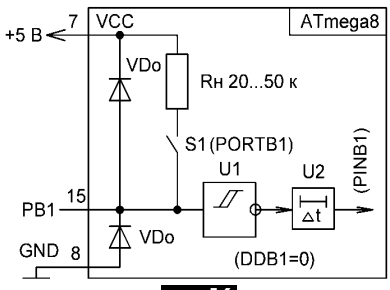

рис.16

Третье отличие – возможность чтения информации прямо с выхода линии порта через сигнал PINB1. В MCS-51 можно было вывести сигнал в линию порта, но узнать, "дошел" ли он по назначению, нельзя. В AVR в линию порта посылается сигнал PORTB1, а истинное значение на выводе PB1 считывается сигналом PINB1 с задержкой в полтора такта частоты процессора (U2). Эта приятная "мелочь" во многих случаях позволяет упрощать схемы.

Четвертое отличие – при нажатии кнопки сброса или начальной подаче питания все линии портов AVR переходят в "оборванное" высокоимпедансное Z-состояние (у MCS-51 – в лог."1"). Этот момент надо учитывать при построении схем нагрузок. Например, если сигнал с выхода МК поступает на базу транзистора, то между ней и общим проводом надо устанавливать резистор 100 кОм, чтобы в момент сброса база не "висела в воздухе".

Пятое отличие – микросхема, у которой все линии портов настроены как входы с внутренними нагрузочными резисторами (PORTn=1, DDRn=0), потребляет примерно на четверть меньше тока, чем при отключенных резисторах (PORTn=DDRn=0). Парадокса здесь нет, просто "парящие в воздухе" входы находятся в неустойчивом состоянии (КМОП-технология) и любая наводка приводит к "дребезгу сигналов" с соответствующим увеличением потребления тока. Вывод – все неиспользуемые входы надо программно нагружать на внутренние резисторы или же ставить внешние.

Из электрических характеристик следует отметить значительно возросший у AVR выходной ток в высоком состоянии, что позволяет подключать светодиоды не только к цепи VCC (SA1 в нижнем положении), но и к GND (SA1 в верхнем положении). Микросхемы семейства AVR первого поколения имеют несимметричную нагрузочную способность, при лог."0" на выходе гарантируют ток 10...20 мА в обоих направлениях, при лог."1" – 3...4 мА. Микросхемы AVR второго поколения почти все обеспечивают одинаковую токовую нагрузку 20 мА. Ток короткого замыкания через одну линию значительно выше до 150...200 мА, но долго микросхема в таком режиме не поработает и с большой долей вероятности выйдет из строя. Аналогичное произойдет при подаче напряжения за пределами от -0,5 В до VCC +0,5 В, когда пробьются диоды VDo.

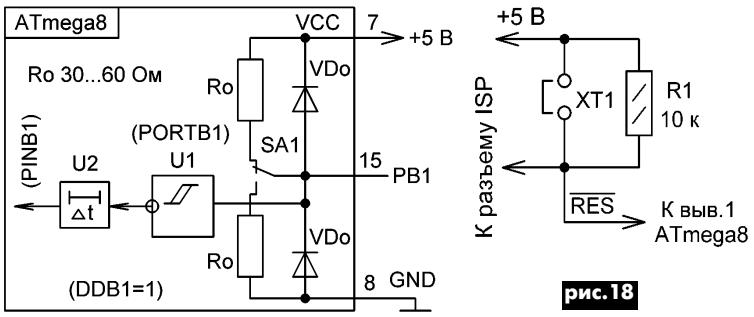

рис.17 **рис.18**

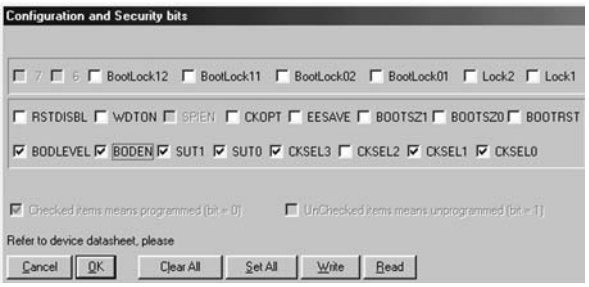

рис.20

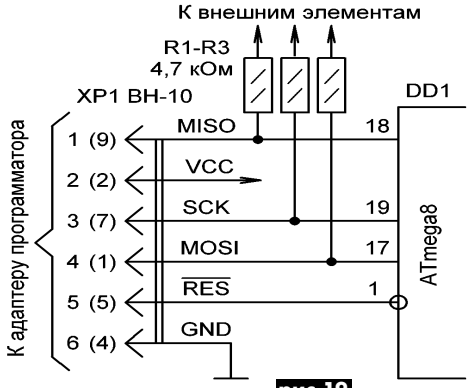

рис.19

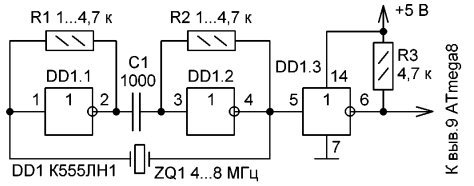

рис.21

Подключение адаптера программатора

Речь пойдет о четырех сигналах MISO, MOSI, SCK, необходимых для внутрисистемного программирования ISP. В "Ступени 2" был показан самый простой способ непосредственного их включения без внешних элементов. Он годится для лабораторных условий. Другое дело – промышленные объекты, где много источников импульсных помех и наводок. Не исключено, что проводники, идущие от разъема ISP к выводам МК, станут своеобразными антеннами и случайная комбинация лог."1" и "0", наводимая на них, будет интерпретироваться процессором как запись данных в FLASH-ПЗУ. В итоге нормально работающий и исправный МК может в непредсказуемые моменты времени портить свою программу. Судя по Интернет-форумам, такие случаи были на практике. Для восстановления ситуации надо заново перепрограммировать МК.

Чтобы подобные казусы не происходили, выводы MISO, MOSI, SCK подключают к цепи GND через резисторы 1...10 кОм и максимально уменьшают длину проводников от ISP-разъема. Иногда обходятся одним резистором 10 кОм между SCK и GND. К выводу $\overline{RES}$ присоединяют RC-цепочку (рис.7) или вообще соединяют с цепью VCC через перемычку XT1 (**рис.18**). При ее наличие работает только внутренний сброс МК, а в ее отсутствие можно осуществить программирование. Следует соблюдать осторожность, иначе забытый на плате джампер станет причиной выхода адаптера программатора из строя. Радикальное решение – вообще отказаться от ISP-разъема и припаивать его на проводах только при регулировании.

Судя по рис.1, выводы MISO, MOSI, SCK могут использоваться и как обычные линии портов PB3, PB4, PB5. Фирма Atmel предлагает развязывать их от сигналов ISP с помощью ограничительных резисторов R1-R3 (**рис.19**). Однако в условиях сильных помех через эти резисторы и подключенные к ним элементы, например удаленные датчики, могут наводиться ложные импульсы. Следовательно, для безопасности надо подключать выводы MISO, MOSI, SCK только к ISP-программатору. В дальнейших схемах выводы программирования, включая $\overline{RES}$, будут оставаться свободными.

Программирование фьюзов

Рассказ о фьюзах будет неполным без описания процедуры их программирования. Для работы потребуется программа PonyProg, тестовое устройство, собранное на макетной плате (см. "Ступень 2"), а также подключенный к разъему ISP адаптер программатора. Считается, что МК в тестовом устройстве уже содержит программу, заставляющую светодиод на макетной плате светиться с переменной яркостью.

Перед программированием фьюзов надо убедиться, что в PonyProg выбрана микросхема именно ATmega8 ("Device – AVRmicro – ATmega8"), иначе конфигурация фьюзов будет неверная. Далее в начальном меню выполнить действия: "Command – Security and Configuration Bits", после чего откроется конфигурационное окно с тремя строками.

Верхняя строка содержит 6 битов защиты информации от копирования и просмотра: BootLock01, 02, 11, 12; Lock1, 2. Для домашних экспериментов все они должны быть очищены (отсутствуют "галочки"), а для промышленных устройств – все установлены (присутствуют "галочки").

Средняя строка содержит 7 старших фьюз-битов, из них RSTDISBL, EESAVE, BOOTSZ0, 1, BOOTRST не должны иметь "галочки" (их программируют только в специально оговоренных случаях), значение фьюзов WDTON и CKOPT определяются по табл.3, 4.

Нижняя строка содержит 8 младших фьюз-битов. Все они выбираются по табл.2, 4.

Важная деталь, которая иногда смущает начинающих. Понятие запрограммированный фьюз (значение "0") соответствует установке в окошке PonyProg "галочки" и, наоборот, без "галочки" у незапрограммированного фьюза (значение "1"). Это связано с особенностями процедуры очистки FLASH-памяти, когда процесс полного стирания сопровождается записью во все ячейки лог."1".

Забывчивость или невнимательность при заполнении фьюзов опасны. Четверть беды, если при неверно установленных фьюзах процессор вдруг станет работать на порядок медленнее (хотелось установить внутренний генератор на 8 МГц, а получилось на 1 МГц). Полбеды, если окажется, что на вход устройства надо подавать внешние импульсы, а в схеме стоит кварцевый резонатор. Хуже всего, если окажется запрограммированным фьюз RSTDISBL, переводящий вход $\overline{RES}$ в дополнительную линию порта PC6. После этого повторное программирование через ISP-адаптер будет невозможным. Придется вызывать "скорую помощь" в виде параллельного программатора (см. "Ступень 1"), которым можно восстановить значение этого фьюза.

"Опасные" фьюзы, наподобие RSTDISBL, присутствуют не во всех типах AVR. Например, в ATmega8515 их нет. В каждом конкретном случае надо смотреть DATASHEET, обращая внимание на раздел Fuse Bits.

На **рис.20** показан внешний вид панели PonyProg с установленными режимами для ATmega8:

отсутствие защиты от просмотра и копирования;
тактирование от внутреннего генератора частотой 8 МГц;
включен детектор Brown-Out с напряжением порога 4 В;
выключен таймер Watchdog.

Сокращенная формула программирования фьюзов: CKSEL0=CKSEL1=CKSEL3=SUT0=SUT1=BODLEVEL=BODEN=0, остальные фьюзы и биты защиты равны "1" (не запрограммированы, отсутствуют "галочки").

При смене условий тактирования следует помнить, что повторное программирование возможно только в той схеме, для которой установлены фьюзы. В частности, нельзя запрограммировать МК, настроенный на режим внешних импульсов, если подключить к его входам кварцевый резонатор и т.д. В качестве "палочки-выручалочки" надо всегда держать под рукой дежурный генератор (**рис.21**). Его выходной сигнал подается на вход XT1 контроллера, что помогает запрограммировать ATmega8 (и не только) в самых причудливых случаях.

Практическое задание. Ознакомиться с DATASHEET на Atmega8 и по возможности распечатать полную версию документа. Провести эксперименты с различными вариантами установки фьюзов на тестовом устройстве.

Литература
1. AVR042: AVR Hardware Design Considerations – http://www.atmel.com/dyn/resources/prod_documents/doc2521.pdf.
2. AVR180: Внешняя защита от провалов напряжения – http://sinbad.narod.ru/avr/BrownOut.htm (рус.).

Микроконтроллеры AVR. Ступень 4

С.М. Рюмик, г. Чернигов

В изучении микроконтроллеров (МК) семейства AVR не надо спешить. Лишь поднявшись на первые три "ступени" (программатор, компилятор, типовые схемы включения), можно приступить к разработке своих собственных конструкций. Начинать, как всегда, легче от простого к сложному.

Идея вдохновляет,
опыт исполняет,
метод – царствует
Н. Векшин

Жизнь не стоит на месте. В новом 2005 году уже успели произойти два важных с точки зрения семейства AVR события.

Во-первых, фирма Atmel расширила классификацию AVR-совместимых МК, выпустив новые серии микросхем Lighting AVR, Smart Battery AVR (**табл.1**). Наметилась тенденция к четкому разделению на универсальные и специализированные контроллеры. Кроме того, появление высоковольтных (4...25 В) МК означает прорыв в технологии. Не исключено, что в будущем и другие контроллеры фирмы Atmel смогут выдерживать 25 В. Учитывая, что нижний предел во вновь разрабатываемых AVR составляет 1,8 В, устройства станут некритичными к напряжению питания (сколько вольт не подаешь, а оно все равно работает!).

Во-вторых, обновился пакет программ WinAVR. Новый релиз имеет номер 20050214, что означает дату рождения 14 февраля 2005 г. Скачать его можно на сайте http://sourceforge.net/projects/winavr/. Объем исполняемого файла "WinAVR-20050214-install.exe" увеличился до 13,7 Мб, внесены улучшения, исправлены неточности, расширены функции.

Тем, кто уже успел скачать предыдущую версию WinAVR 20040720, не следует спешить с ее удалением с винчестера. Золотое правило безопасности учит, что на компьютере надо держать архивы как минимум двух последних версий обновляемых программ – для перепроверки и для сравнения возможностей. Разумеется, основной рабочей версией должна быть последняя по времени программа, предыдущую версию предварительно деинсталлируют.

Интерфейс WinAVR-20050214 остался без изменений, компиляция программ, как и прежде, производится через редактор Programmers Notepad (PN). Важными являются новшества в Си-компиляторе, входящем в пакет WinAVR. Он стал чуть быстрее, чуть правильнее, пополнился список поддерживаемых МК.

Небольшой нюанс. Иногда WinAVR в Интернете называют компилятором. На самом деле это пакет программ, в состав которого входит ядро настоящего свободно распространяемого Си-компилятора AVR-GCC. Он является неотъемлемой частью коллекции компиляторов GCC (**G**NU **C**ompiler **C**ollection), поддерживающих Windows, Linux, языки Си, Си++, Си–, Ada, Fortran, Java. В WinAVR включена версия компилятора, специально "заточенная" под архитектуру AVR, платформу Windows и процедуры Си, Си++ стандартов ANSI, ISO. Компиляторы GCC имеют свой сайт http://gcc.gnu.org/ и свою эмблему (**рис.1**).

рис.1

При обнаружении неточностей в компиляции следует обращаться в базу ошибок GCC http://gcc.gnu.org/bugs.html. О том, что к сообщениям пользователей там прислушиваются, свидетельствуют благодарности от имени авторов к программистам с явно славянскими фамилиями.

Практическую апробацию компилятора AVR-GCC проводят не только одиночки-любители, но и солидные учебные заведения. Например, в Государственном университете Гранд Вэлли (г. Аллендейл, Мичиган, США) на лабораторных работах по моделированию динамических систем используют пакет WinAVR и МК ATmega32 (http://claymore.engineer.gvsu.edu/ ~jackh/eod/egr345.html). Студенты приобщаются к программированию, а заодно учатся работать со свободно распространяемым компилятором.

Язык Си – повторение материала

Радиолюбителей, читающих настоящий цикл статей, можно условно разделить на три категории. Первые из них хорошо знают Ассемблер, умеют составлять на нем программы для AVR, но хотели бы перейти на язык более высокого уровня. Вторые – освоили программирование на языке Си для МК семейства MCS-51 и хотели бы адаптировать свои знания применительно к AVR. Третьи – начинают "с нуля". Во всех перечисленных случаях не лишним будет изучить (или вспомнить) базовые понятия и конструкции языка Си. В **табл.2–4** даны соответствующие ссылки на журналы РА за 2004 год. Краткая справка по синтаксису языка Си приведена в РА 6/2004.

В дальнейшем при составлении листингов программ предлагается использовать ограниченное число операторов и максимально простые логические построения. Аналогично и с изучением возможностей МК ATmega8, принятого за базовый элемент. Сложные для понимания или редко встречающиеся в нем функции будут вынесены за скобки. Основная "сверхзадача" заключается в том, чтобы приобщиться к методологии, а не в том, чтобы потратить годы на шлифование знаний об одном единственном контроллере. Опыт подтверждает мудрую мысль Козьмы Пруткова: "Никто не обнимет необъятного".

"Tips and Tricks"

Перевод текста с английского интуитивно понятен – "подсказки и уловки". Речь пойдет о полезных мелочах, облегчающих работу с пакетом WinAVR.

• Если при инсталляции WinAVR удалить "галочку" возле пункта "Add Shortcuts to Desktop" в разделе "Устанавливаемые компоненты", то на рабочий стол не будут выводиться ярлыки инструментов.

• Чтобы при запуске PN не открывалось новое окно, нужно снять галочку возле "Tools-Options-Start with a new blank document".

• Чтобы текст программы в PN был красиво отформатирован по столбцам, следует сменить тип шрифта Lucida Console на Courier New Cyr по методике "Tools-Options-Style Schemes-<выбрать поочередно все пункты, каждый раз изменяя название шрифта>-OK".

• Чтобы строки в PN были пронумерованы, следует включить опцию "View-Line Number". Кроме того, надо проследить, чтобы в самом конце листинга оставалась одна лишняя пустая строка, иначе появится замечание компилятора.

Итак, новая версия WinAVR установлена в папку C:\WinAVR. На очереди самостоятельная разработка программы для пробного устройства.

Первая конструкция

У профессиональных программистов с незапамятных времен существует обычай. В первой пробной программе они выводят на экран монитора сообщение: "Hello, world!" Тем самым миру сообщается об успешном освоении еще одного алгоритмического языка или новой компьютерной платформы.

Таблица 1

Семейство	Обозначение микросхем	Особенности
"CAN AVR"	AT90CAN128	Интерфейс CAN
"LCD AVR"	ATmega169, 329, 3290, 649, 6490	Подключение ЖКИ
"Lighting AVR"	AT90PWM2, AT90PWM3	Управление лампами и двигателями
"megaAVR"	ATmega48, 8, 88, 8515, 8535, 16, 162, 165, 168, 32, 325, 3250, 64, 640, 645, 6450, 128, 1280, 1281, 2560, 2561	Универсальная серия с многофункциональными возможностями
"Smart Battery AVR"	ATmega406	Питание 4...25 В
"tinyAVR"	ATtiny13, 15L, 2313, 25, 26, 28L, 45, 85	Малогабаритный корпус

Таблица 2

Си-функция	Номер журнала
while	PA5-12
if	PA6-12
else	PA6-12
for	PA7-12
return	PA7-12
switch	PA8-11
break	PA8-11
do	PA10

Таблица 3

Ключевые слова	Номер журнала
#include	PA5-12
#define	PA6-11
void	PA5-12
unsigned	PA6-12
volatile	PA10, 12
extern	PA11, 12
static	PA12

Таблица 4

Структурная схема	Номер журнала	Особенности
Тип 1	PA5	Без внешних и внутренних функций
Тип 2	PA7	С внутренними, но без внешних функций
Тип 3	PA11	С внутренними и внешними функциями

Электронщики, не долго думая, стали вводить в пробные программы для МК точно такую же фразу. Хорошо, если МК сопрягается с компьютером или имеет выход на многоразрядный жидкокристаллический экран. А если нет? В таком случае логичнее вместо "Hello, world!" засвечивать обычные индикаторы, управляемые от кнопок. Эффект – одинаковый.

На **рис.2** показана схема пробного устройства, которое применялось при изучении МК семейства MCS-51 (см. РА 5/2004), а на **рис.3** – его AVR-аналог. Алгоритм работы: индикатор HL1 должен светиться при нажатой и не светиться при отпущенной кнопке SB1, индикатор HL2, наоборот, должен светиться при отпущенной и не светиться при нажатой кнопке SB2.

Различия в схемах. Конденсатор C4 в MCS-51 обеспечивает единичный импульс сброса. Для AVR сброс формируется внутри МК DD1, а резисторы R1, R2 повышают помехоустойчивость.

Элементы C2, C3, ZQ1 в MCS-51 стабилизируют тактовую частоту. Для AVR они не нужны, поскольку используется внутренний RC-генератор с частотой 1 МГц.

Нагрузкой кнопок SB1, SB2 в обеих схемах служат внутренние резисторы между цепью +5 В (VCC) и входами МК соответственно P3.4, P3.5 и PD4, PD5. Их номиналы – десятки килоом.

Назначение резисторов R3, R4 на рис.3 не совсем очевидно. Действительно, при нормальной работе они не нужны, токи через кнопки SB1, SB2 протекают слабые, защищаться нет от чего. Но вдруг произойдет сбой процессора, и линии PD4, PD5 программно переключатся из режима вход в режим выход с высоким логическим уровнем? В этой ситуации при нажатых кнопках SB1, SB2 и отсутствии резисторов R3, R4 через каждый из выводов 6, 11 микросхемы DD1 будет протекать ток короткого замыкания до 100 мА. Если не принять спешных мер, то через некоторое время корпус DD1 перегреется и МК может выйти из строя. Такая ситуация теоретически возможна. Главную опасность представляет не само замыкание (МК его кратковременно выдерживает), а длительность воздействия, число одновременно закороченных выводов и тепловой нагрев корпуса.

Другая крайность. Представим радиолюбителя-оптимиста, который считает, что вероятность программного сбоя именно в его конструкции чуть выше, чем прямое попадание метеорита в быстро движущийся автомобиль... Однако в первой прошивке программы он по недосмотру допускает ошибку и случайно программирует линии PD4, PD5 как выходы. Пока удастся разобраться в причинах "почему не работает", МК при нажатых кнопках и отсутствии резисторов может перегреться.

Очевидно, компромиссное решение лежит где-то посередине. При отладке устройства на макетной плате ограничительные резисторы по входам, которые могут быть напрямую соединены с общим проводом GND или с питанием VCC, лучше поставить, а вот в отработанной конструкции без них можно обойтись. В дальнейшем такие резисторы рисоваться не будут, но их наличие каждый должен держать в уме.

Пример преобразования программы MCS-51 в AVR

Среди тех, кто на уровне банальной эрудиции поверхностно знаком с основами языка Си, бытует мнение о легкости, с которой можно перенести программу с одной микроконтроллерной платформы на другую. Действительно, для языка Си это сделать намного легче, чем для Ассемблера. Однако как только от слов переходят к делу, бравурные речи приобретают минорный тон. Оказывается, адаптация программ требует больше времени, чем планировалось ранее, да и узкопрофильный специалист с такой задачей не справится. Необходимо знать отличия в архитектуре МК, особенности организации портов, системы прерываний, грамматики компиляторов.

Считается, что проще конвертировать программу, чем создавать ее заново. Заманчиво было бы взять за основу алгоритм работы уже существующей программы и заменить в ней названия линий портов, подкорректировать функции и регистры прерываний. Работа требует внимательности, но будет вознаграждена сторицей.

Например, электрические схемы, приведенные на рис.2, 3, функционально идентичны, следовательно, и Си-программы будут у них похожи. В **листинге 1** показан текст для MCS-51, а в **листинге 2** – конвертированный текст для AVR.

Пояснения к листингам

Беглое сравнение листингов 1 и 2 явно не в пользу последнего. Только сейчас можно по достоинству оценить простоту управления портами в MCS-51, где не надо заботиться о фьюзах и не надо создавать make-файл. За расширенные возможности МК AVR приходится "платить" усложнением программного обеспечения.

Поскольку листинг 1 уже рассматривался ранее, то основное внимание будет уделено листингу 2.

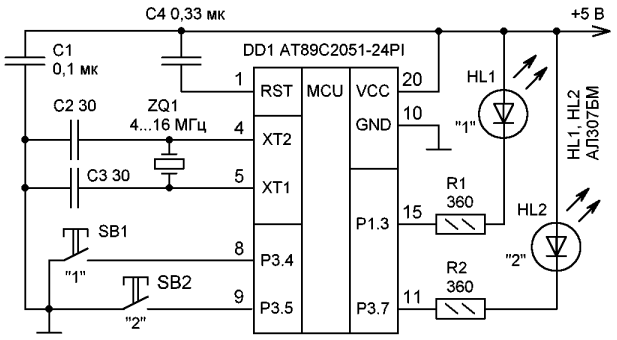

рис.2

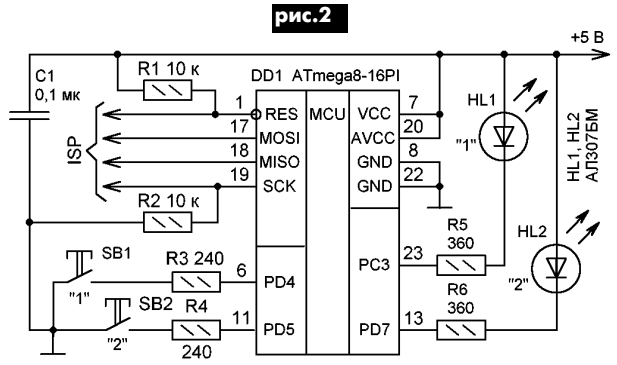

рис.3

Листинг 1

```
/*Пробный тест, МК шаг 3, журнал РА, №5, 2004= 1*/
#include <io51.h>       /*Системная библиотека= 2*/
       /*Пустая строка, разделитель заголовка= 3*/
void main (void)     /*Начало "тела" программы= 4*/
{                 /*Начало главной функции main= 5*/
  while (1)                 /*Бесконечный цикл= 6*/
  {          /*Начало функции while в строке 6= 7*/
    P1.3 = P3.4;         /*Повторение SB1 (HL1)= 8*/
    P3.7 = P3.5 ^ 1;      /*Инверсия SB2 (HL2)= 9*/
  }       /*Окончание функции while в строке 6=10*/
}         /*Окончание функции main и программы=11*/
```

Листинг 2

```
//Пробный тест. =AVR, ступень 4=. Журнал РА, №4-2005  =1
//Make: Name=avr41, MCU=atmega8, Level=2, Debug=VMLab =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)  =3
#include <avr/io.h>          //Библиотека ввода-вывода =4
          /*Пустая строка-разделитель*/              // =5
int main(void)              //Начало основной программы =6
{ PORTB = PORTC = PORTD = 0xFF; //Входы с резисторами =7
  DDRC |= _BV(PC3);                 //PC3 выход с лог.1 =8
  DDRD |= _BV(PD7);                 //PD7 выход с лог.1 =9
  while (1)  //Бесконечный цикл между строками 11-17 =10
  {  //Проверка лог.1 на линии PD4 и передача ее PC3 =11
    if (bit_is_set(PIND,PD4)) PORTC |= _BV(PC3);  // =12
    else PORTC &= ~_BV(PC3);          //Иначе PC3 лог.0 =13
 //Проверка лог.0 на линии PD5 и установка лог.1 PD7 =14
    if (bit_is_clear(PIND,PD5)) PORTD |= _BV(PD7); //=15
    else PORTD &= ~_BV(PD7);          //Иначе PD7 лог.0 =16
  }            //Окончание функции "while" в строке 10 =17
}       //WinAVR-20050214, длина программы 142 байта =18
```

Строка 1 начинается с двух наклонных линий, иначе называемых "прямой слэш". Все, что справа от них – это комментарии к программе. Компилятор допускает еще одну разновидность текстовых пояснений, как показано в строке 5. В отличие от слэш-варианта, между символами "/*" и "*/" может находиться не одна, а несколько строк подряд. С точки зрения компилятора оба варианта равноценны.

Строки 2, 3 специфичны для AVR и отсутствуют в MCS-51. Это подсказка пользователю, который будет повторять конструкцию, о содержимом make-файла в программе MFile и установке фьюзов в программе PonyProg. Человек, который в первый раз видит листинг, будет знать, что название всех файлов проекта начинается с "avr41", в устройство надо ставить микросхему ATmega8, которая работает от внутреннего генератора 1 МГц, а компиляция программы будет производиться по второму уровню оптимизации с отладочной информацией в формате VMLab.

Строка 6. Согласно стандартам языка Си основная программа должна иметь тип "int". В листинге 1 это требование не обязательно.

Строка 7. После подачи питания все линии портов МК автоматически настраиваются в режим входа без нагрузочного резистора (PORTB=PORTC=PORTD=DDRB=DDRC=DDRD=0). Чтобы входы "не висели в воздухе", во все разряды портов записываются единицы (шестнадцатиричный байт 0xFF), при этом внутри микросхемы к линиям портов подключаются резисторы сопротивлением 20...50 кОм. Вольтметром можно убедиться, что напряжения на линиях близкие к уровню питания.

Строка 8 – перевод линии PC3 порта C из состояния ВХОД в состояние ВЫХОД. К этой линии в схеме, показанной на рис.3, подключаются резистор R5 и светодиод HL1. Исходный уровень будет лог."1", поскольку ранее в строке 7 он был записан в регистр PORTC.

Расшифровка оператора: "Установить в "1" третий разряд регистра DDRC". Эта конструкция будет в дальнейшем часто встречаться в программах, поэтому чуть подробнее. С левой стороны расположено название 8-разрядного регистра DDRC, подлежащего изменению. Знаки "|=" означают, что будет произведена операция ЛОГИЧЕСКОЕ ИЛИ между левой и правой частью формулы, а результат помещен в левую часть, т.е. в регистр DDRC. Символы "_BV" – это системное макроопределение компилятора AVR-GCC, которое устанавливает в "1" один из битов (**B**it **V**alue). Какой именно – указывается в круглых скобках далее. В нашем случае это третий бит (PC3). Итого, в регистре DDRC устанавливается в "1" бит номер 3, остальные биты не изменяются.

В Си-программах встречаются и другие записи этого оператора, например, DDRC |= (1 << PC3), что эквивалентно.

Строка 9 аналогична строке 8. Она устанавливает в "1" седьмой разряд регистра DDRD, переводя линию PD7 с входа на выход.

Строки 12, 13. Выражение "bit_is_set" является системным макроопределением компилятора AVR-GCC. Оно проверяет содержимое разряда PD4 в регистре PIND и, если результат равен "1", то выдает сообщение "истина", в противном случае – "ложь". К линии PD4 на схеме присоединена кнопка SB1, поэтому проверяется, нажата она или нет. Полная расшифровка функции "if-else": "Если на линии PD4 обнаружена лог."1", то занести лог."1" в линию PC3 и перейти к строке 14. Если на линии PD4 присутствует лог."0", то перейти к строке 13 и установить лог."0" в линии PC3".

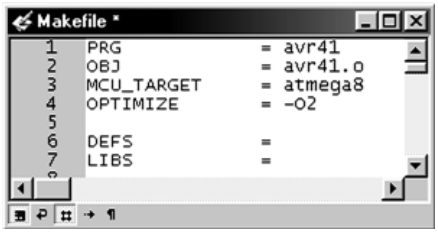

рис.4

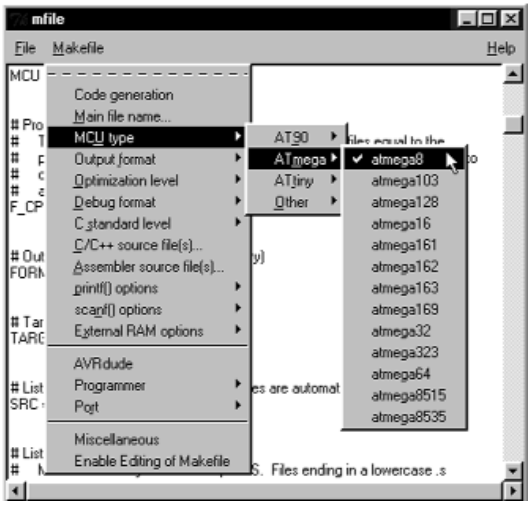

рис.5

Таблица 5

Исходное состояние линии PC3			Последующее состояние линии PC3			
DDRC	PORTC	Вх-Вых	Вход (z)	Вход (R)	Выход (0)	Выход (1)
0	0	Вход (z)	—	PORTC |= _BV(PC3);	DDRC |= _BV(PC3);	PORTC |= _BV(PC3); DDRC |= _BV(PC3);
0	1	Вход (R)	PORTC &= ~_BV(PC3);	—	PORTC &= ~_BV(PC3); DDRC |= _BV(PC3);	DDRC |= _BV(PC3);
1	0	Выход (0)	DDRC &= ~_BV(PC3);	PORTC |= _BV(PC3); DDRC &= ~_BV(PC3);	—	PORTC |= _BV(PC3);
1	1	Выход (1)	PORTC &= ~_BV(PC3); DDRC &= ~_BV(PC3);	DDRC &= ~_BV(PC3);	PORTC &= ~_BV(PC3);	—

Условные обозначения: ВХОД (z) - без резистора; ВХОД (R) - с резистором; ВЫХОД (0) - лог."0"; ВЫХОД (1) - лог."1"

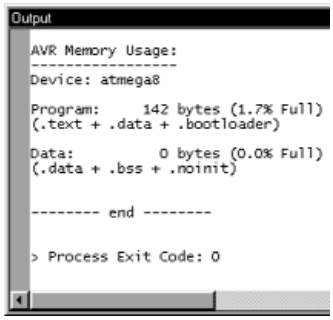

рис.6

Знак "волна" или "тильда" впереди "_BV" в строке 13 означает инверсию выражения справа, т.е. вместо установки "1" будет установлен "0" в третьем бите. Если учесть, что знак ЛОГИЧЕСКОЕ ИЛИ заменен знаком ЛОГИЧЕСКОЕ И (&), то получится, что в регистре PORTC установится в "0" бит номер 3, остальные биты не меняются.

В **табл.5** приведены варианты Си-операторов, которые надо вводить в листинги, для перевода линий в различные состояния. Все примеры рассчитаны на линию PC3, в остальных случаях меняется последняя буква в названии регистра (PORTB, PORTD, DDRB, DDRD) и порядковый номер линии (PB0-PB7, PC0-PC7, PD0-PD7), например, PORTB |= _BV(PB0).

Строки 15, 16 аналогичны строкам 12, 13, но с макроопределением "bit_is_clear" (дословный перевод – "бит очищен"), которое выдает сообщение "истина" при наличие лог."0" в разряде PD5 регистра PIND и "ложь" при лог."1". Если сравнить с листингом 1, то эквивалент этой сложной конструкции содержится в одном операторе строки 9.

Строка 18 содержит упоминание о текущей версии пакета WinAVR, при которой производилась компиляция программы, и соответствующий ей размер кодов прошивки МК.

Создание MAKE-файла

Как известно, каждая Си-программа в WinAVR должна иметь свой собственный make-файл. Самый простой способ его создания – это копирование уже существующего файла "makefile" из папки C:\WinAVR\examples\demo\ и замены в нем первых трех строк по образцу, показанному на **рис.4**. Разумеется, исходный текст программы "avr41.c" (листинг 2) и скопированный make-файл должны находиться вместе в отдельно созданной папке рабочего проекта.

Чтобы каждый раз не копировать make-файл и не изменять в нем текст, в пакете WinAVR имеется средство для его автоматизированного создания – инструмент MFile (автор Joerg Wunsch, г. Дрезден, Германия). Порядок действий. Запустить на выполнение программу MFile: "Пуск – Программы – WinAVR – MFile". Выбрать пункт меню Makefile и последовательно заполнить "анкету":

• В пункте "Makefile-Main file name..."–Main file" ввести имя проекта avr41 и нажать OK.

• В пункте "Makefile-MCU type-ATmega" выбрать тип микросхемы atmega8 (**рис.5**).

• В пункте "Makefile-Optimization level" задать уровень оптимизации 2 как наиболее часто встречающийся в программах WinAVR. Для справки, цифра "0" – без оптимизации, буква "s" – минимальная длина кодов, цифры "1-3" – три разных метода оптимизации, причем цифра "3" не означает лучший вариант, все зависит от конкретной Си-программы.

в пункте "Makefile-Debug format" установить формат отладочной информации "AVR-ext-COFF (AVR Studio 4.07+, VMLab 3.10+)".

Остальные пункты менять не надо, пусть остаются принятыми по умолчанию.

Далее следует сохранить полученный файл: "File–Save as...–<выбрать папку, где находится файл "avr41.c">–<указать имя "makefile">–Сохранить". Закрыть программу MFile ("File – Exit") и просмотреть содержимое вновь созданного файла в любом текстовом редакторе. Традиционно makefile не имеет расширения, так принято в среде UNIX.

Следующим этапом надо открыть программу PN, загрузить в нее Си-программу "avr41.c" и откомпилировать через пункт "[WinAVR]

Make All". По сравнению с make-файлом, созданным по рис.4, добавилось число выводимых сообщений. В частности, появилась полезная информация о проценте занимаемого прошивкой места в FLASH-ПЗУ МК (**рис.6**).

После компиляции открыть программу PonyProg, ввести в нее вновь созданный файл "avr41.hex", подключить кабель ISP-адаптера к пробному устройству, подать питание 5 В. Далее запрограммировать фьюзы SUT0, CKSEL3, CKSEL2, CKSEL1 согласно строке 3 листинга 2, после чего нажать одновременно клавиши <Ctrl> и <P>. Через 30 с дождаться сообщения "Programming Successful" и проверить кнопками SB1, SB2 пробного устройства правильность свечения индикаторов HL1, HL2.

"Маячок-мигалка"

Второй эксперимент будет заключаться в проверке динамических возможностей МК. Стандартная задача – заставить попеременно мигать два светодиода, превратив их в своеобразные "маячки". Для удобства предлагается использовать ту же самую схему, что на рис.3, но без кнопок SB1, SB2 и резисторов R3, R4.

Управляющая программа приведена в **листинге 3**. Ее построение сходно с листингом 2.

Строки 13, 14 – стандартный прием установки лог."0" и лог."1" на выходах линий портов. Это надо запомнить.

Строка 15 – организация задержки времени. При запуске программы на выполнение значение переменной "pause" устанавливается в 0 (строка 7). При каждом проходе строки 15 к переменной "pause" добавляется единица (два знака "+"). Через 15000 проходов (итераций) число, накопленное в "pause", сравняется с 15000, и управление будет передано на строку 16. Каждая итерация занимает какое-то время процессора, в результате чего получается определенная задержка во времени. Изменить ее можно в любую сторону, например, увеличив число 15000 до 65535 или уменьшив до 1.

Строка 18 – второй вариант задержки времени. Свое начальное значение (15000) переменная "pause" получает при выходе из строки 15. Каждый проход строки 18 уменьшает ее величину на единицу (два знака "–"). Через 15000 итераций значение "pause" обнулится, и управление будет передано на строку 19. Получается своеобразная лестница: в строке 15 идем вверх, в строке 18 – вниз.

Виртуальный осциллограф

После компиляции программы "avr42.c" и получения файла "avr42.hex" производят программирование МК и опробование в работе. Казалось бы, мигание светодиодов должно быть симметричным (одинаковое число итераций в строках 15, 18 листинга 3), однако на глаз заметно, что HL2 светится дольше, чем HL1.

Для разгадки феномена будет привлечен "компьютерный судья".

Существует проблема, общая для всех без исключения Си-компиляторов. Речь идет о формировании точных отрезков времени. Как, например, узнать, сколько миллисекунд уйдет на выполнение операторов в строках 15, 18 листинга 3? Иными словами, какой период свечения индикаторов HL1, HL2 и какова скважность импульсов на линиях PC3, PD7 микросхемы DD1 в "маячке-мигалке"?

Визуальный или секундомерный контроль времени здесь не проходит, так как светодиоды мигают примерно 3-4 раза в секунду. Проблема решается при наличии цифрового запоминающего осциллографа, в котором измерения длительностей проводят с высокой точностью.

С другой стороны, а чем компьютер хуже? Он всегда под рукой, да и математические уравнения решает лучше осциллографа. Если "объяснить" компьютеру, как устроен МК ATmega8, заложить в него точные значения времени выполнения контроллерных инструкций, то можно смоделировать работу процессорной системы и рассчитать абсолютно все отрезки времени без погрешностей.

Такие программы существуют и одна из них – это Visual Micro Lab (VMLab) фирмы AMTools. На сайте http://www.amctools.com имеется информация о двух версиях программы – бесплатной демонстрационной (http://www.amtools.net/vmlab310.zip, 3,8 Мб) и платной полной (75 евро). Ограничения в демо-версии следующие: размер исполняемого кода не более 4 Кб, объем используемой в МК памяти не более 50%, число одновременно открываемых файлов проекта не более 4, моделирование не более 50 тысяч инструкций.

На счастье, простые любительские конструкции, использующих ATmega8, проходят ниже планки ограничений. Для сравнения, 4 Кб – это в два раза больше, чем у AT89C2051. Следовательно, демо-версия VMLab может служить хорошим подспорьем разработчику, заменяя во многих случаях паяльник.

Важная деталь. Связка WinAVR-VMLab обладает уникальными возможностями по сравнению с другими Си-компиляторами. Только для WinAVR в VMLab можно на ходу перекомпилировать Си-программу и сразу же увидеть результат коррекции на экране виртуального осциллографа. Еще одна приятная мелочь – возможность "шагать" прямо по тексту Си-программы и делать остановки, наблюдая за диаграммами.

Листинг 3

```
//Маячок-мигалка =AVR, ступень 4=. Журнал РА, №4-2005 =1
//Make: Name=avr42, MCU=atmega8, Level=2, Debug=VMLab =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)  =3
#include <avr/io.h>          //Библиотека ввода-вывода =4
//==============ОСНОВНАЯ ПРОГРАММА================== =5
int main(void)             //Начало основной программы =6
{ unsigned int pause=0;         //Переменная для паузы =7
                                                    // =8
  PORTB = PORTC = PORTD = 0xFF; //Входы с резисторами =9
  DDRC |= _BV(PC3);              //PC3 выход с лог.1 =10
  DDRD |= _BV(PD7);              //PD7 выход с лог.1 =11
  while (1)  //Бесконечный цикл между строками 13-19 =12
  { PORTC |= _BV(PC3);           //PC3 выход с лог.1 =13
    PORTD &= ~_BV(PD7);          //PD7 выход с лог.0 =14
    while (++pause < 15000);  //Пауза 15000 итераций =15
    PORTC &= ~_BV(PC3);          //PC3 выход с лог.0 =16
    PORTD |= _BV(PD7);           //PD7 выход с лог.1 =17
    while (--pause > 0);      //Пауза 15000 итераций =18
  }          //Окончание функции "while" в строке 12 =19
}      //WinAVR-20050214, длина программы 148 байтов =20
```

Эмулятор, симулятор, имитатор

Как правильно называть VMLab – эмулятор, имитатор или симулятор МК? Различия между этими понятиями следующие.

Слово "эмуляция" (emulation) относится к однородным системам, составленным из одного и того же материала [1]. Например, компьютер, состоящий из металла и кремния, не может эмулировать человека, состоящего из белковых органических соединений, и наоборот. При эмуляции одна система выполняет функцию другой, стараясь делать это в реальном времени. Эмуляции на IBM PC подвергаются: игровые автоматы, игровые приставки, домашние компьютеры, китайские "Тетрисы" и даже программируемые карманные калькуляторы.

Процесс имитации (imitation) неразрывно связан с подражанием [1]. Человек может петь как соловей, а компьютер – "разговаривать" как человек. Программы-имитаторы – это настольные игры (имитация логического мышления), программы синтеза голосовой информации (имитация голоса), машинные собеседники (имитация искусственного интеллекта).

Технический термин "симуляция" (simulation) относится к случаю, когда одна система моделирует поведение другой, имея на входах одни и те же данные, а на выходах – идентичные или подобные [1]. Это эквивалентно "черному ящику" с неизвестной внутренней структурой, причем его начинка может быть разнородной, а не одинаковой, как при эмуляции. Известны компьютерные симуляторы самолетов, вертолетов, велосипедов, парусников, автомобилей и т.д.

Как видно, к VMLab из трех определений больше подходит название "симулятор", что подтверждает фирменный термин "software simulation with analog simulation" из его файла помощи. Программные симуляторы, в отличие от внутрисхемных эмуляторов, работают гораздо медленнее. Скорость расчетов у VMLab низкая, требуется как минимум IBM PC с процессором 800...1000 МГц, чтобы долго не ждать до прорисовки очередной линии.

Технология работы с VMLab

Краеугольным камнем в философии VMLab служит понятие "проект". Это текстовый файл с расширением .prj, в котором на специальном языке описывается словами электрическая схема подключения внешних цепей к МК. Примеры проектов расположены в папках C:\Vmlab\AVR_demo\ и C:\Vmlab\WinAVRdemo\, которые появляются в процессе инсталляции VMLab на диск С. Судя по ним, подключать к МК можно: резисторы, конденсаторы, транзисторы, светодиоды, ЖК-индикаторы, кнопки, синусоидальные генераторы и даже терминал RS-232 с изменяемой скоростью передачи данных.

Этапы работ для получения файла-проекта "маячка-мигалки".

1. Запустить на выполнение программу VMLab: "Пуск – Программы – Vmlab". Создать в ней новый проект "Project-New project", заполнить в пошаговом режиме (Step1-4) графы, как показано на **рис.7**. В качестве пути хранения проекта указать ту папку, где находятся файлы "avr42.c", "makefile", "avr42.hex" и т.д. Нажать OK, после чего на экране монитора появятся три окна. В окне "Messages" наблюдать сообщение "PRJ file is OK!".

2. Развернуть окно с проектом avr42.prj и произвести коррекцию текста согласно **листингу 4** (англоязычные комментарии сокращены и русифицированы). В самом конце текста надо не забыть доба-

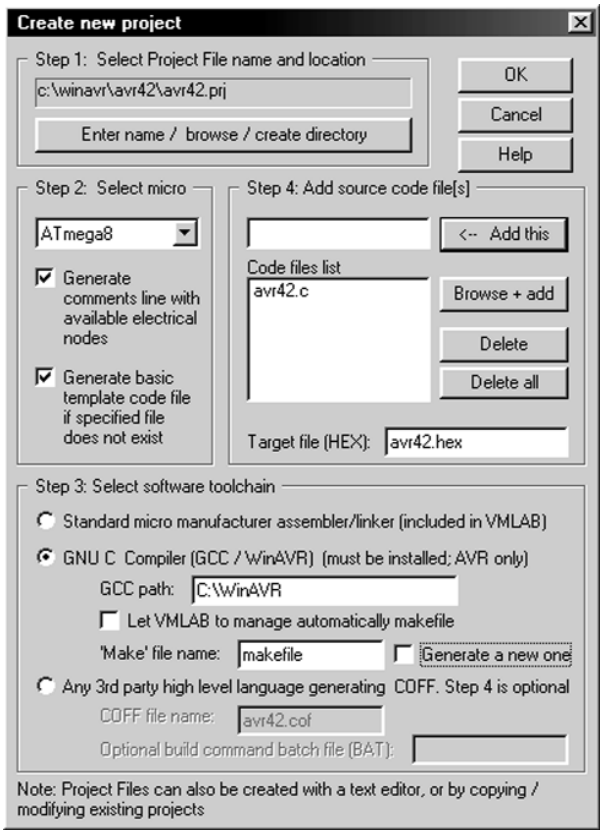

рис.7

вить одну лишнюю пустую строку. Проект составлен в предположении, что на рис.3 интерес представляют только функционально значимые элементы: R5, R6, HL1, HL2, DD1. Резисторы R1, R2 и конденсатор C1 в логической работе не участвуют, а элементы SB1, SB2, R3, R4 для "маячка" не нужны.

3. Построить проект, для чего нажать клавишу F9 или иконку "Build". В окне "Messages" наблюдать сообщение "Success! All ready to run".

4. Запустить проект на выполнение, для чего нажать клавишу F5 или иконку "Go/Continue". В окне "Messages" наблюдать сообщение "Starting hardware-software co-simulation..."

5. Открыть экран виртуального осциллографа "View-Scope". Установить в его настройках развертку по горизонтали 20 мс/дел, развертку по вертикали 2 В/дел. Подождать, пока будет построен график изменения напряжений на линиях PC3, PD7 (**рис.8**). Как и ожидалось, единичному уровню PC3 соответствует нулевой уровень PD7 и наоборот. Особенность VMLab состоит в том, что рисуются не только логические уровни на линиях МК, но и вычисляются их абсолютные аналоговые величины.

6. Для временной остановки изображения нажать клавиши Shift+F5 или иконку "Pause program". Далее в левой стороне экрана поставить точку в окошке "Cursor 1", затем "мышью" на осциллограмме провести вертикальную линию первого замера. Поставить точку в окошке "Cursor 2" и провести линию второго замера. Цифровые данные будут высвечиваться на поле осциллограммы. В частности, на рис.8 отсчеты равны 1322,4 и 1427,2 мс, следовательно, время между переходами напряжений "0"–"1" (свечение индикатора HL2) составляет t2=104,8 мс. Сместив линию курсора к левому фронту импульса, аналогичным образом определяют время свечения индикатора HL1 t1=60 мс.

Обсуждение результатов

Почему t2 почти в половину больше t1? В листинге 3 за время t2 отвечает строка 15, за время t1 – строка 18. И в том, и в другом случае задержка производится подсчетом 15000 инструкций. Однако в строке 15 к ним каждый раз добавляется сравнение с числом 15000 функции "while". Эта операция по времени длиннее, чем сравнение с нулем в строке 18 (такая особенность у всех МК), следовательно, и общая задержка будет больше.

Модельные эксперименты

Виртуальный осциллограф помог определить точные значения времени, не прибегая к услугам измерительных приборов. Для проведения модельных экспериментов можно попробовать изменить период следования импульсов свечения. Для этого, не выходя из VMLab, развернуть окно "Code. Target file: avr42.hex", выбрать внизу закладку "avr42.c" и заменить в строке 15 число 15000 на 1000 (на вопрос "Restart simulation?" нажать "Да"). Затем клавишами F9, F5 запустить построение графика, открыть окно осциллографа, уменьшить развертку по горизонтали и наблюдать рисунок с измененными параметрами. Иногда для правильного отображения времени может потребоваться полное перестроение проекта: "Project-Re-build all".

Там же, в закладке "avr42.c", можно установить точки останова, нажав одну или несколько кнопок с левой стороны листинга. По мере прохождения импульсов, задействованные строки в Си-программе окрашиваются желтым цветом. При останове строка становится серой. В пошаговой отладке задействуются три иконки с названием "Step" в верхней части экрана. После завершения работы над проектом, его следует закрыть "Project-Close project", изменения автоматически будут сохранены в Си-программе и HEX-файле.

Листинг 4

```
; ************************************************************
; PROJECT: Маячок-мигалка. =Микроконтроллеры AVR. Ступень 4=
; AUTHOR: . Журнал РА №4-2005, Рюмик С.М.
; ************************************************************
.MICRO "ATmega8"       ; Микроконтроллер (МК) ATmega8
.TOOLCHAIN "GCC"       ; Си-компилятор AVR-GCC
.GCCPATH   "C:\WinAVR" ; Путь к папке с пакетом WinAVR
.GCCMAKE   "makefile"  ; Использовать "родной" make-файл
.TARGET    "avr42.hex" ; Указатель на HEX-файл
.SOURCE    "avr42.c"   ; Указатель на Си-программу
; ------------------------------------------------------------
.TRACE                 ; Начало блока трассировки связей МК
; ------------------------------------------------------------
.POWER VDD=5 VSS=0     ; Цепи питания VDD=5В, VSS=общий провод
.CLOCK 1meg            ; 1 МГц, тактовая частота МК
.STORE 250m            ; 250 мс, длина развертки по горизонтали
; ------------------------------------------------------------
; Описание линий связи на электрической схеме
D1 VDD D1_NODE         ; HL1 подключается между +5В и R5
R5 D1_NODE PC3 0.36K   ; R5=360 Ом подключается между HL1 и PC3
D2 VDD D2_NODE         ; HL2 подключается между +5В и R6
R6 D2_NODE PD7 0.36K   ; R6=360 Ом подключается между HL2 и PD7
.PLOT v(PC3) v(PD7)    ; Вывод на экран напряжений PC3, PD7
```

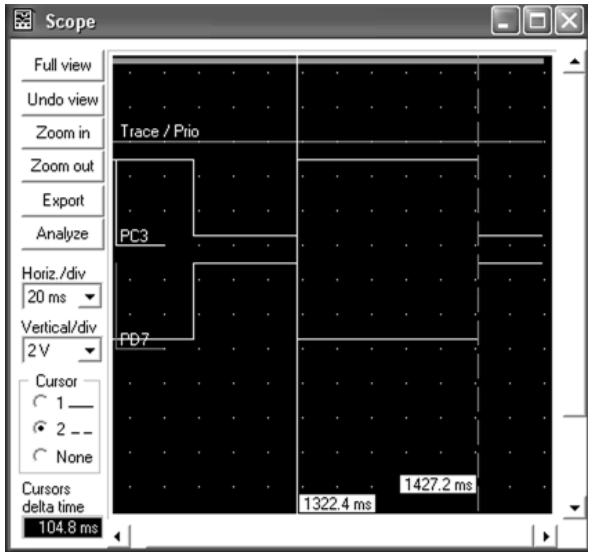

рис.8

Литература

1. The Blackwell Guide to Philosophy of Computing and Information. Edited by Luciano Floridi, August 2003, 392 pages, http://www.blackwellpublishing.com/pci/downloads/Glossary.pdf.

Микроконтроллеры AVR. Ступень 5

Как измерить Человека?
А. Сент-Экзюпери

С.М. Рюмик, г. Чернигов

Чтобы устройство, собранное на микроконтроллере (МК), было интересным, оно должно содержать "изюминку". Разумеется, трудно ожидать от простых учебных конструкций феноменальных возможностей, уникальных параметров и досконального сервиса, но все же...

Разработка приборов, содержащих МК, состоит из нескольких этапов.

• Логический уровень: анализ поставленной задачи, уяснение алгоритма работы, проведение математических расчетов.

• Физический уровень: определение числа входных и выходных сигналов, выбор МК, интерфейса сопряжения, составление электрической схемы устройства.

• Программный уровень: разработка алгоритма функционирования МК, составление листинга программы, компиляция, компьютерное моделирование процессов.

• Практический уровень: монтаж и сборка устройства, программирование МК, отладка, опробование в реальной работе.

Не все составляющие в перечисленных этапах обязательны для выполнения. Допускается их коррекция и повторное проведение. В зависимости от сложности поставленной задачи состав уровней может меняться (сокращаться, уточняться, дополняться). Разработка – процесс творческий. И вдвойне приятно, если удается найти нестандартные схемные и программные решения или неожиданные сферы применения.

В частности, предлагается разработать несколько простых микроконтроллерных приборов для экспериментов в такой "малотехнической" области науки, как психология. Заодно можно будет попрактиковаться в программировании линий портов МК, в формировании задержек сигналов, в устранении "дребезга" контактов кнопок, в организации светодиодных шкал.

"Иллюзия октавы"

По определению словаря, октава – это музыкальный интервал, отношение частот между нотами которого равно 2. Древние римляне удивились бы такому противоречию, ведь в переводе с латыни слово "октава" означает "восьмая". Разгадка банальная: в диатоническом музыкальном ряду, который был изобретен позже, октава является восьмой по счету ступенью.

Эффект под названием "иллюзия октавы" относится скорее к области практической психологии, чем к музыке. Впервые он был исследован в 1974 г. Дианой Дойч (Diana Deutsch) – профессором психологии Университета Калифорнии, США (http://philomel.com/pdf/Nature-1974_251_307-309.pdf, 67 Кб). Суть эксперимента. Испытуемому надевают стереонаушники и предлагают прослушать звуковой сигнал, в котором в противофазе смешаны ноты основной частоты 400 Гц и на октаву выше 800 Гц (**рис.1,а**). Ноты звучат с одинаковой длительностью и громкостью, каждое ухо получает 50% высоких и 50% низких тонов.

Казалось бы, слышимость в обоих ушах должна быть примерно одинаковой. Однако суммарную звуковую картину редко кто слышит правильно, а вместо этого возникает множество иллюзий. Например, одним людям кажется, что появляются биения сигналов, другим – что звуки периодически перемещаются из одного уха в другое, третьим – что высокий тон слышится с паузами из одного уха, а низкий – из другого и т.д. Любопытно, что картина практически не меняется, если переставить наушники местами или изменить частоту основного тона.

По статистике большинство людей слышат периодический высокий звук из правого уха (R) и низкий – из левого (L). На **рис.1,б** показана нотная запись этого случая. На ней специально выделены ноты с прямоугольником в центре, которые соответствуют диаграмме R на рис.1,а. Напрашивается догадка, что "иллюзия октавы" связана с особенностями работы мозговых полушарий, которые различаются у левшей и правшей.

Определить "господствующее ухо" поможет несложный прибор, представляющий собой двухканальный генератор звуков.

Исходные данные для разработки: частота основного тона 400 Гц, удвоенного – 800 Гц, длительность звучания одной ноты 250 мс (одна четвертая при темпе 240 ударов в минуту), нотные диаграммы в левом и правом каналах – согласно рис.1,а.

Электрическая схема прибора "Иллюзия октавы" (**рис.2**) содержит управляющий МК DD1, который работает от внутреннего RC-генератора частотой 1 МГц. Кварцевая стабилизация не требуется, поскольку слуховой аппарат человека чувствителен к изменению интервалов, а не к абсолютной высоте звука. На линиях портов PB0, PB1 формируются сигналы для правого R и левого L наушников, подключенных к разъему X1.

Конденсаторы C2–C5 совместно с резисторами R1*, R2* образуют фильтры, которые сглаживают форму прямоугольных сигналов, устраняют металлические хрипы и призвуки. Для соблюдения симметрии номиналы элементов в каждом канале должны быть идентичными. Кроме того, резисторы R1*, R2* снижают громкость звука до комфортных 40...50 дБ (уровень разговора в комнате). Уменьшать сопротивления R1, R2 не рекомендуется, чтобы наушники не превратились для слушателя в источник рева авиационного двигателя...

Кнопкой SB1 дискретно регулируют длительность звучания одной ноты от 175 до 475 мс (5,7...2,1 Гц) с пятью градациями. В исходных данных такого требования не было, да и в экспериментах Дианы Дойч – тоже. Но практика показала, что для лучшего осмысления результатов желательно прослушивать ноты при разных темпах.

Си-программа прибора "Иллюзия октавы" приведена в **листинге 1**.

Строки 2, 3 содержат сведения, необходимые для создания makefайла (MFile) и программирования фьюзов (PonyProg).

Строки 5, 6. Чтобы использовать в программе стандартные системные функции задержек во времени, надо подключить библиотеку "avr/delay.h" и указать тактовую частоту МК в герцах. Буквы "UL" в константе F_CPU расшифровываются как "**U**nsigned **L**ong", т.е. "длинное целое число" в пределах 0...4,2 млрд.

Строка 13. Типичный прием перевода в режим выхода сразу двух линий PB0 и PB1.

Строки 15–25. Формирование временной диаграммы нечетного такта (длительность одной ноты) согласно **рис.3**. Библиотечная функция "_delay_loop_1(X)", где "X"=TIK=208, производит задержку времени "T1" по формуле T1[мкс]=3*F_CPU[МГц]*X=3*1*208=624 мкс, что близко к требуемому для 800 Гц полупериоду 625 мкс. Меняя константу TIK в строке 7 в пределах 1...256, можно увеличить или уменьшить частоту звучания ноты.

Строки 26–28. Если нажата кнопка SB1, то происходит циклическое изменение переменной "temp" согласно ряду: 70, 100, 130, 160, 190. От нее зависит длительность одного такта (см. строки 15, 30). Каждое срабатывание кнопки сопровождается перерывом в звучании на 1 с, тем самым слушателю понятно, что происходит переход к следующему темпу. Библиотечная функция "_delay_loop_2(Y)" производит задержку времени T_2 по формуле T2[мс]= (F_CPU[МГц]*Y)/250= =(1*62500)/250=250 мс. Меняя значение "Y" в пределах 1...65536 можно увеличить или уменьшить паузу после нажатия кнопки.

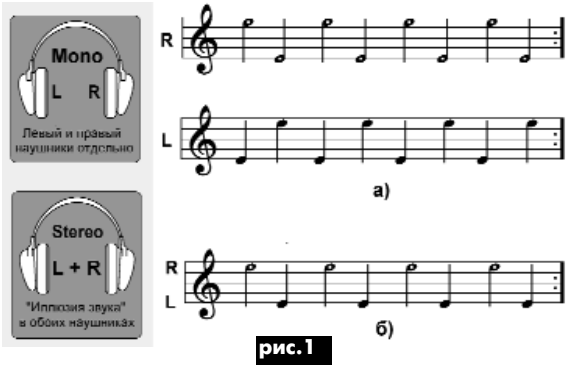

рис.1

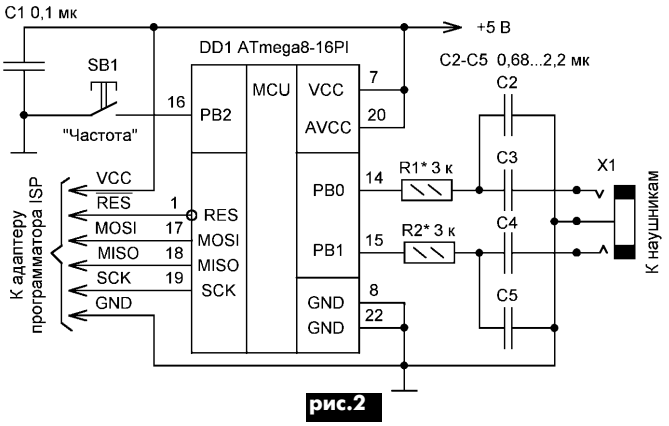

рис.2

Строки 30–40 аналогичны строкам 15–25, но с переменой частот в двух каналах.

Прошивка МК и опробование в работе

Компиляция программы производится средствами пакета WinAVR: MFile и Programmers Notepad (PN). Зашивать МК можно как и прежде, используя программу PonyProg. Кстати, на сайте ее автора Claudio Lanconelli появилась новая версия 2.06f, релиз от 5 марта 2005 г. (http://www.lancos.com/e2p/V2_06/ponyprogV206f.zip, 606 Кб, **рис.4**). Интерфейс программы остался прежним, добавлены новые типы МК, исправлены неточности.

В пакете WinAVR тоже имеется свой программатор в виде инструмента AVRDUDE (**AVR D**ownloder **U**ploader, автор Brian S. Dean), который позволяет зашивать МК не хуже, чем PonyProg, причем быстрее и проще. Единственное, что останавливает от его широкого применения – это отсутствие поддержки COM-адаптеров (см. рис.5, 6 в "Ступени 1"), а также относительная сложность процедуры зашивки фьюзов. Тем не менее, ознакомиться с возможностями запасного программатора полезно.

Чтобы активизировать AVRDUDE, необходимо при создании make-файла указать параметры его интерфейса. Порядок действий. Запустить на выполнение программу MFile: "Пуск – Программы – WinAVR – MFile". Последовательно заполнить "анкету":

в пункте "Makefile-Main file name..."–Main file" ввести имя проекта "avr51";

в пункте "Makefile-MCU type-ATmega" выбрать тип микросхемы "atmega8";

в пункте "Makefile-Optimization level" задать уровень оптимизации 2;

в пункте "Makefile-Debug format" установить формат "AVR-ext-COFF (AVR Studio 4.07+, VMLab 3.10+)";

в пункте "Makefile-Programmer" указать тип адаптера "pony-stk200";

в пункте "Makefile–Port" задать номер параллельного порта "lpt1" или "lpt2", "lpt3".

Далее следует сохранить полученный файл: "File–Save as...–<выбрать папку, где находится файл "avr51.c">–<указать имя "make-file">–Сохранить". Закрыть программу "File – Exit".

Перед программированием МК надо зашить его фьюзы согласно строке 3 листинга 1. Лучше всего это сделать с помощью PonyProg, где они наглядно размещены на панели. Графическая оболочка AVR-DUDE C:\WinAVR \bin\avrdude-gui.exe еще слишком "свежая", да и вводить в ней числами фьюзы неудобно и даже опасно. Например, ошибешься при переводе фьюза в HEX-код, получишь "0" в разряде RST-DISBL (см. "Ступень 3") и придется обращаться за помощью к параллельному программатору.

После зашивки фьюзов можно программировать МК, для чего открыть программу PN, загрузить в нее листинг 1 и откомпилировать через пункт "Toools – [WinAVR] Make All". Небольшой нюанс. При компиляции появится предупреждение "Warning: "F_CPU" redefined". Это следствие того, что в строке 6 листинга 1 указано значение тактовой частоты 1 МГц, а в make-файле по умолчанию оно установлено 8 МГц. Поскольку главным является все же листинг Си-программы, то число F_CPU в make-файле будет проигнорировано. Компиляция пройдет успешно. Чтобы подобное сообщение не появлялось, надо открыть Makefile в PN и заменить строку 51 "F_CPU 8000000" строкой "F_CPU 1000000". Если забудете это сделать – не беда, ошибки в работе контроллера не произойдет.

Подключить к МК LPT-адаптер (см. рис.3, 4 в "Ступени 1"). Подать питание на разработанное устройство. Не выходя из PN, выбрать пункт "Tools – [WinAVR] Program", и через пару секунд МК будет запрограммирован. Меньшее по сравнению с PonyProg время программирования связано с тем, что AVRDUDE при верификации проверяет не полный объем FLASH-ПЗУ, а только ту часть, которая была записана.

Эксперименты с "иллюзией октавы"

Для проведения экспериментов потребуются добровольные помощники. Каждому из них предлагается в течение 20...30 с прослушать звуковой сигнал "Иллюзия октавы" в стереонаушниках. Желательно, чтобы наушники имели большие амбушюры для хорошей изоляции звукового потока из одного уха в другое. Темп выдачи нот можно изменять, нажимая и удерживая до появления секундной паузы кнопку SB1.

После прослушивания меняют наушники местами и повторяют эксперимент.

Если испытуемый утверждает, что звук высокого тона 800 Гц слышен преимущественно в правом ухе, то его "ведущее ухо" – правое и он с большой долей вероятности правша. У левшей обычно нет четкой локализации эффекта, они слышат смешанные звуки, часто с биениями. Особый интерес представляют случаи, когда "правша по жизни" имеет "левосторонний" слух и наоборот. Это может быть признаком необычных способностей (таланта) у человека, надо только внимательнее к нему приглядеться.

Генератор биоритмов

В современной физиологии существует метод исследования функ-

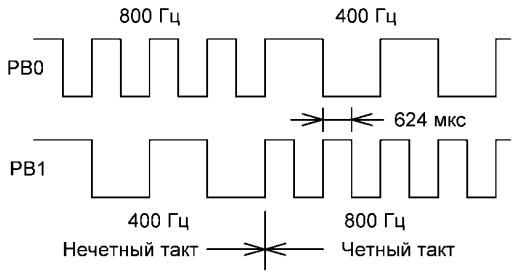

рис.3

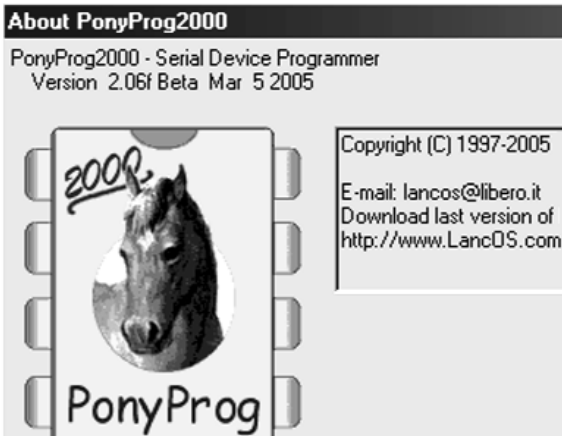

рис.4

Листинг 1

```
//"Иллюзия октавы", =AVR, ступень 5= Журнал РА-5/2005 =1
//Make: Name=avr51, MCU=atmega8, Level=2, Debug=VMLab =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)  =3
#include <avr/io.h>          //Библиотека ввода-вывода =4
#include <avr/delay.h>         //Библиотека задержек =5
#define F_CPU 1000000UL      //Тактовая частота 1 МГц =6
#define TIK 208      //Константа для задержки 624 мкс =7
//==============ОСНОВНАЯ ПРОГРАММА================== =8
int main(void)             //Начало основной программы =9
{ unsigned char a, temp=100;   //Счетчики тактов нот =10
           //Частота 400 (800) Гц, темп 2,1...5,7 Гц =11
  PORTB = PORTC = PORTD = 0xFF; //Входы с резисторами=12
  DDRB |= _BV(PB0) | _BV(PB1); //PB0, PB1 выходы "1" =13
  while (1) //Бесконечный цикл из чет.-нечет. тактов =14
  { for (a=temp; a > 0; a--)     //L=800 Гц, R=400 Гц =15
    { _delay_loop_1(TIK);       //Задержка на 624 мкс =16
      PORTB &= ~_BV(PB0);     //Лог."0" (PB0), 800 Гц =17
      _delay_loop_1(TIK);       //Задержка на 624 мкс =18
      PORTB |= _BV(PB0);      //Лог."1" (PB0), 800 Гц =19
      PORTB &= ~_BV(PB1);     //Лог."0" (PB1), 400 Гц =20
      _delay_loop_1(TIK);       //Задержка на 624 мкс =21
      PORTB &= ~_BV(PB0);     //Лог."0" (PB0), 800 Гц =22
      _delay_loop_1(TIK);       //Задержка на 624 мкс =23
      PORTB |= _BV(PB0) | _BV(PB1);    //Две лог."1" =24
    }       //Окончание длительности нечетного такта =25
    if (bit_is_clear(PINB, PB2))   //Если нажата SB1 =26
    { if ((temp += 30) > 200) temp=70; //Смена темпа =27
      for (a=0; a<4; a++) _delay_loop_2(62500); //1 с=28
    }  //Окончание действий после нажатия кнопки SB1 =29
    for (a=temp; a > 0; a--)     //L=400 Гц, R=800 Гц =30
    { _delay_loop_1(TIK);       //Задержка на 624 мкс =31
      PORTB &= ~_BV(PB1);     //Лог."0" (PB1), 800 Гц =32
      _delay_loop_1(TIK);       //Задержка на 624 мкс =33
      PORTB |= _BV(PB1);      //Лог."1" (PB1), 800 Гц =34
      PORTB &= ~_BV(PB0);     //Лог."0" (PB0), 400 Гц =35
      _delay_loop_1(TIK);       //Задержка на 624 мкс =36
      PORTB &= ~_BV(PB1);     //Лог."0" (PB1), 800 Гц =37
      _delay_loop_1(TIK);       //Задержка на 624 мкс =38
      PORTB |= _BV(PB0) | _BV(PB1);    //Две лог."1" =39
    }         //Окончание длительности четного такта =40
  }                        //Переход к нечетному такту =41
}      //WinAVR-20050214, длина программы 246 байтов =42
```

ционирования головного мозга с помощью снятия электроэнцефалограммы (ЭЭГ). На голову пациента надевают специальные датчики, которые улавливают биопотенциалы. Ученые выяснили, что регистрируемые сигналы содержат сложную смесь периодических колебаний. Если применить для их анализа разложение в ряд Фурье, то можно оценить частотный спектр сигналов и амплитуду каждой составляющей.

Мозг человека в разных состояниях генерирует разные сигналы ЭЭГ с преобладание тех или иных частот, или ритмов (**табл.1**). Наблюдательный читатель может спросить, почему названия ритмов идут не по порядку букв греческого алфавита? Дело в том, что изобретатель метода ЭЭГ, австрийский физиолог Ганс Бергер (Hans Berger), проводя опыты в 1924–29 гг., сначала обнаружил колебания на частоте 10 Гц и назвал их "альфа". В последствие были открыты "бета"-, "дельта"-, "тета"-ритмы и еще ряд промежуточных поддиапазонов в них.

Если головной мозг представить в виде электронного устройства, то кроме "передатчика", генерирующего ритмы, в нем должен быть и "приемник". Напрашивается идея "засинхронизировать" внутренние биоритмы внешним генератором воздействия. Таким способом можно мягко, ненавязчиво подсказать мозгу время, когда пора спать, а когда, наоборот, надо заставить организм работать на полную мощность.

Воздействие может производиться через слуховой, зрительный и тактильный каналы. Самым щадящим и безопасным следует признать звуковой раздражитель. Для этого необходим прибор, который бы генерировал сигналы в диапазоне биоритмов 0,5...30 Гц. Однако эти частоты напрямую человек не ощущает, их необходимо перенести на звуковую поднесущую.

В частности, известен эффект бинауральных биений, когда на левое ухо подается, например, частота 400 Гц, а на правое ухо – 410 Гц. При прослушивании такой смеси вместо двух отдельных тонов отчетливо слышен усредненный звук частотой 405 Гц, модулированный по амплитуде разностной частотой 10 Гц. В **табл.2** приведены субъективные ощущения человека при разных частотах модуляции. Именно этот способ и будет использоваться в экспериментах.

Исходные данные для разработки. Двухканальный генератор с изменяемой частотой прямоугольных импульсов в диапазоне 200...400 Гц. Сдвиг частот между каналами должен соответствовать ритмам из табл.1. Обеспечить кнопочный выбор поддиапазонов, световую индикацию и плавное регулирование громкости.

Электрическая схема генератора биоритмов показана на **рис.5**. Кнопками SB1–SB4 выбирают один из четырех ритмов, индикацию которых обеспечивают светодиоды HL1–HL4. Выходные сигналы через элементы RP1, R5, R6, C2, C3 поступают на стереонаушники. Переменный резистор RP1 регулирует громкость звучания одновременно в двух каналах. Он может быть заменен двумя одиночными резисторами.

Кнопкой SB5 циклически меняют частоту основного тона, согласно шести градациям, в диапазоне примерно 150...450 Гц. Почему "примерно"? Потому что тактовый генератор МК DD1, используемый для перестройки частоты, берется внутренний, а пределы его девиации в DATASHEET на ATmega8 оговорены лишь приблизительно, с погрешностью ±25...50%.

Переключатель SA1 позволяет изменять тембр звука, значительно оживляя его за счет перекрестной модуляции. Через конденсатор C4* сигналы левого и правого каналов сдвигаются по фазе и частично смешиваются. В итоге получается красивый объемный эффект, напоминающий звучание приставки "лесли" от электрогитары.

Си-программа генератора биоритмов приведена в **листинге 2**.

Строка 3. Фьюзы установлены для режима внутреннего RC-генератора частотой 8 МГц. Соответственно в строке 6 указано это значение в Герцах.

Строка 16. Новинка – внутренний регистр контроллера под названием OSCCAL (**OSC**illator **CAL**ibration). Этот регистр незримо участвует во всех операциях, когда МК работает от внутреннего RC-генератора. При каждом включении питания в него автоматически заносится число, которое с точностью до 3% устанавливает тактовую частоту, максимально близкую к номиналу 1 МГц (2, 4 или 8 МГц). Просмотреть это число можно через PonyProg, выбрав пункты меню: "Command – Read Osc. Calibration Byte". В частности, для ATmega8 OSCCAL=167 или в HEX-виде 0xA7.

Тактовую частоту генератора можно в любой момент времени подстроить программно, причем довольно в широких пределах. На **рис.6** показана типовая зависимость частоты генерации от содержимого регистра OSCCAL. Как видно, обнуление регистра (строка 16) приводит к уменьшению частоты с 8 до 4,3 МГц, т.е. почти в 2 раза! Приняв к сведению, что график приблизительный, точных расчетов по нему делать нельзя.

Таблица 1

Ритм	Диапазон частот, Гц	Уровень, мкВ	Физиологические особенности
Дельта	0,5...4	10...250	Фаза глубокого сна, бессознательное состояние
Тета	4...8	10...200	Фаза быстрого сна, полудрема, работа подсознания, медитация, гипноз, характерно у детей
Альфа	8...13	30...100	Расслабленность, дневная мечтательность, абстрактное мышление, запоминание, ясное, светлое состояние, хорошая адаптация к изменениям, творческое озарение
Бета	13...30	5...30	Активное, бодрствующее сознание, логическое мышление, концентрация внимания, решение проблем, стресс

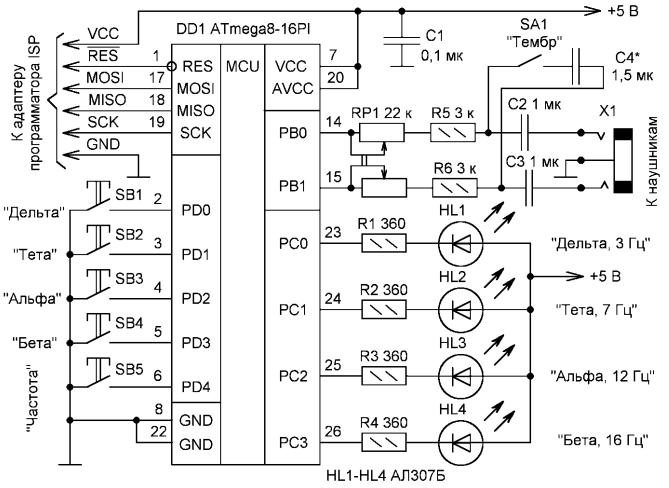

рис.5

Таблица 2

Диапазон частот, Гц	Субъективные ощущения	Ритм
1...4	Мягкий розлив	Дельта
4...6	Переход от биений к вибрато	Тета
6...8	Зона вибрато у певцов	Тета, альфа
8...10	Вибрато, переходящее в тембр	Альфа
10...15	Острый розлив	Альфа, бета
15...20	Бурление	Бета
20...60	Диссонанс	Бета

Строки 19, 20. Стандартный прием получения на линии PB0 прямоугольных импульсов, по форме близких к меандру, с частотой, определяемой числом 64 в строке 19.

Строки 22–33. Проверка нажатия кнопок SB1–SB4 и изменение переменной "sdvig", которая определяет разность частот в левом и правом каналах. Поскольку абсолютная величина этой переменной не так велика, как хотелось бы (не хватает быстродействия МК), то частоты биоритмов получаются фиксированными 3, 7, 12, 16 Гц. Строго говоря, более плавную перестройку частоты можно было бы получить, используя прерывания, но цель рассматриваемой Си-программы – показать способы прямого формирования сетки частот через задержки во времени.

Строки 34–37. Формирование шести градаций частоты тона путем изменения содержимого регистра OSCCAL согласно ряду: 0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF.

Строка 38. Чтобы пауза после нажатия кнопок была примерно одинаковой для всех тактовых частот вне зависимости от OSCCAL, переменная "b" каждый раз вычисляется через переменную "c".

Строка 43. Формирование на линии PB1 сигнала, близкого к меандру, с частотой, зависящей от переменной "sdvig". Итого, сигнал PB1 меняется по частоте, а сигнал PB0 (строка 20) – нет.

Эксперименты с генератором биоритмов

Считается, что бинауральные явления вызывают эффект синхронизации работы левого и правого полушарий головного мозга, характерные для медитации. Не случайно, что некоторые фирмы разрабатывают методики улучшения работоспособности, памяти, уверенности в себе именно за счет воздействий на организм с частотами альфа и бета-ритмов.

Известны и компьютерные программы, например, BrainWave Generator (http://www.bwgen.com/bwgen31.exe, 1,2 Мб), "Мозгоправ" (http://andrei512.narod.ru/programs/Mozgoprav_03.2005.zip, 42 Кб), которые генерируют звуки, позволяющие расслабиться или мобилизоваться. Микроконтроллерный генератор биоритмов фактически является их упрощенным компактным вариантом.

Судя по откликам в Интернете, звуковое воздействие генератора биоритмов иногда может приводить к самым неожиданным последствиям, например к отказу от курения (http://www.bwgen.com/comments.htm)! Разумеется, основополагающим фактором здесь является не бинауральный эффект, а настроенность организма на подсознательное восприятие информации. Как здесь не вспомнить крылатую фразу известного врача и писателя Владимира Леви: "Самовнушение – это все, что с нами происходит".

Начинать работу с прибором надо в том диапазоне частот, который соответствует желаемому состоянию организма. Например, для достижения предсонного состояния следует нажать и удерживать кнопку SB2 до появления секундной паузы, затем кнопкой SB5 и переключателем SA1 установить экспериментально подобранную частоту (для каждого человека свою), при которой достигается наибольший эффект. Длительность воздействия: кому-то остаточно 10 минут, кому-то полчаса. В завершении сеанса громкость звука желательно постепенно уменьшить резистором RP1. Разумеется, нельзя в это время заниматься активной мозговой деятельностью (чтение книги), иначе и сна не будет, и книгу придется перечитывать заново.

Генератор биоритмов нельзя отождествлять с "волшебной палочкой". Это всего лишь помощник. Многое зависит от психоэмоционального состояния человека и его физиологических особенностей. Кроме того, головной мозг человека инстинктивно сопротивляется любому воздействию извне. Чтобы помочь ему расслабиться, надо, например, для погружения в сон принять наиболее удобную позу и свести к минимуму мыслительный процесс.

Если человек не восприимчив к генератору биоритмов или наблюдаются явления, противоположные ожидаемым (вместо сна бодрствование или наоборот), то эксперименты надо прекратить. В целом работа с генератором биоритмов рассчитана на потенциально здоровых людей, или, как шутят медики, на "не полностью еще обследованных". Категорически не рекомендуется прослушивать монотонные ритмы людям, склонным к эпилепсии и применяющим кардиостимуляторы.

Тестер инерционности слуха

Природа предусмотрительно ограничила человеку диапазон слышимых звуков от 10...20 Гц до 18...22 кГц. Естественный отбор закрепляет в последующих поколениях лишь те наследственные признаки, которые необходимы для выживания. К примеру, у дельфинов диапазон сдвинут в сторону ультразвука 100 Гц...200 кГц, что позволяет им пользоваться эхолокацией.

Кроме частотных свойств слух обладает инерционностью. Если какой-либо источник звука выключить на время, меньшее 30 мс, а затем снова включить (без щелчка), то человек не обнаружит паузы. Практическое следствие – эффект эха начинает ощущаться, если отраженный сигнал приходит с задержкой более 40...50 мс.

Степень инерционности слуха отличается у разных людей. Один из способов его оценки заключается в прослушивании "вращающегося звука" (Rotor Sound). Испытуемый помещается в центре комнаты, а вокруг него создается механическим или электрическим способом перемещающееся по кругу звуковое поле. В домашних условиях проще это сделать электрическим путем, расположив по сторонам комнаты как минимум 8 динамиков (**рис.7**). Если динамики будут излучать звук по очереди, слева направо или справа налево, то создается виртуальный источник "вращающегося звука". Скорость вращения легко изменить с помощью уменьшения или увеличения длительности звучания каждого динамика.

При определенной частоте вращения наступает эффект "объемности" звука, когда нельзя точно локализовать направление его прихода. Эта точка и будет характеризовать степень инерционности слуха человека.

Исходные данные для разработки. Число звуковых каналов – 8. Каналы должны работать по очереди одинаковое время, в согласованном кольце. Частота основного тона 1...2 кГц. Предусмотреть кнопочное изменение периода "вращения" в пределах 100...900 мс (ча-

Листинг 2

```
//Генератор биоритмов, =AVR, ступень 5=, РА, №5, 2005 =1
//Make: Name=avr52, MCU=atmega8, Level=2, Debug=VMLab =2
//Фьюзы: SUT0=CKSEL3=CKSEL1=CKSEL0="галочки" (8 МГц)  =3
#include <avr/io.h>          //Библиотека ввода-вывода =4
#include <avr/delay.h>          //Библиотека задержек =5
#define F_CPU 8000000UL      //Тактовая частота 8 МГц =6
//==============ОСНОВНАЯ ПРОГРАММА================== =7
int main (void)            //Начало основной программы =8
{ unsigned int a=0;           //Общий счетчик времени =9
  unsigned char b, c=0;   //Вспомогательные счетчики =10
  unsigned char sdvig=63;  //Делитель сдвига частоты =11
  //Два канала 200-400 Гц со сдвигом частоты 3-16 Гц =12
  PORTB = PORTD = 0xFF; //B, D - входы с резисторами =13
  DDRB |= _BV(PB0) | _BV(PB1); //PB0, PB1 выходы "1" =14
  DDRC = 0x0F; PORTC = 0xFE; //PC0-PC3 выходы, HL1=0 =15
  OSCCAL=0x00; //Начальная низкая частота генератора =16
  while (1) //Бесконечный цикл из чет.-нечет. тактов =17
  { a++;   //Увеличение счетчика времени (0...65535) =18
    if ((a%64)==0)      //Делитель на 64 для канала 1 =19
    { PORTB ^= _BV(PB0);  //Изменение PB0 0-1-0-1... =20
      if (PIND != 0xFF)     //Проверка нажатия кнопок =21
      { if (bit_is_clear(PIND,PD0)) //Если нажата SB1=22
        { sdvig = 63; PORTC = 0xFE;     //Дельта, HL1 =23
        } //Установлен дельта-ритм 3 Гц, светится HL1=24
        if (bit_is_clear(PIND,PD1)) //Если нажата SB2=25
        { sdvig = 62; PORTC = 0xFD;       //Тета, HL2 =26
        }  //Установлен тета-ритм 7 Гц, светится HL2 =27
        if (bit_is_clear(PIND,PD2)) //Если нажата SB3=28
        { sdvig = 61; PORTC = 0xFB;      //Альфа, HL3 =29
        } //Установлен альфа-ритм 12 Гц, светится HL3=30
        if (bit_is_clear(PIND,PD3)) //Если нажата SB4=31
        { sdvig = 60; PORTC = 0xF7;       //Бета, HL4 =32
        } //Установлен бета-ритм 16 Гц, светится HL4 =33
        if (bit_is_clear(PIND,PD4)) //Если нажата SB5=34
        { if (++c > 5) c = 0;     //Цикл смены частот =35
          OSCCAL = c * 0x33; //Новое значение частоты=36
        }      //Установлена новая частота генератора =37
        for (b = 0; b < (8 + c * 7); b++)      //Цикл =38
        { _delay_loop_2(62500);   //Задержка времени =39
        }      //Выполнена пауза после нажатия кнопки =40
      }       //Окончание действий при нажатии кнопок =41
    }                 //Окончание работы с каналом 1 =42
    if ((a%sdvig)==0) PORTB ^= _BV(PB1);   //Канал 2 =43
  }    //Переход к началу бесконечного цикла "while" =44
}      //WinAVR-20050214, длина программы 328 байтов =45
```

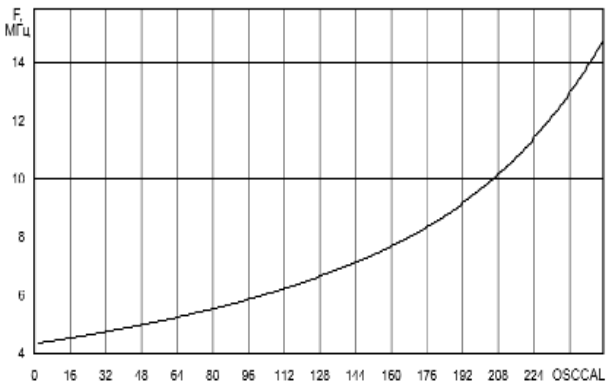

рис.6

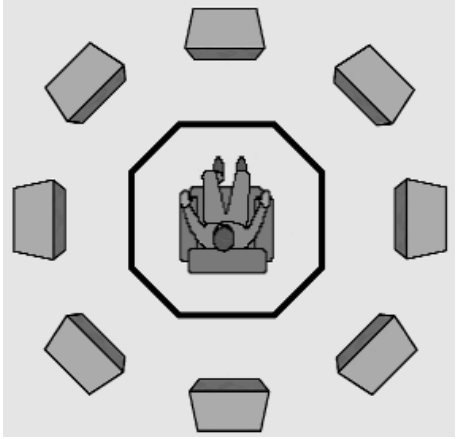

рис.7

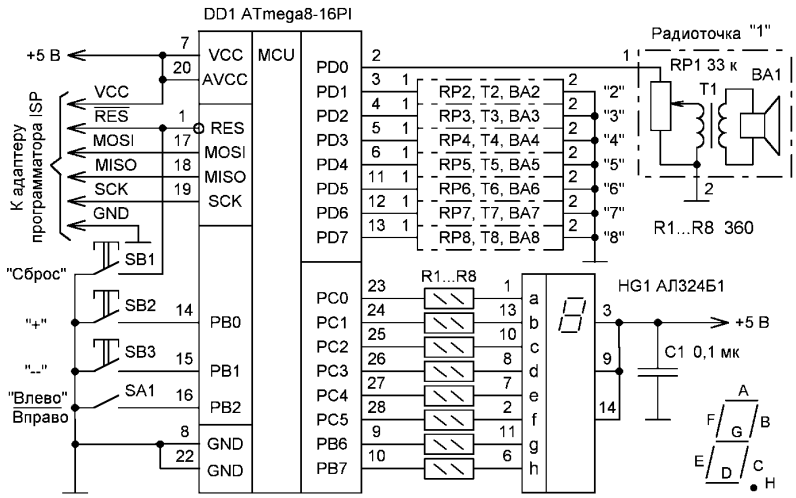

рис.8

стота 10...1,1 Гц). Индикацию периода вывести на семисегментный индикатор.

Электрическая схема прибора для тестирования инерционности слуха показана на **рис.8**. Динамики BA1–BA8 вместе с трансформаторами T1–T8 и регуляторами RP1–RP8 используются от радиотрансляционных точек. Их прямое подключение к линиям порта D является допустимым, поскольку система рассчитана на импульсный характер работы в каналах.

Индикация выводится на светодиодный индикатор HG1, ток через сегменты которого ограничивают резисторы R1–R8 200...360 Ом. Кнопками SB2, SB3 задают скорость "вращения звука" соответственно больше и меньше. Переключатель SA1 служит для смены направления вращения "слева направо" или "справа налево". Индикатором этого момента является светящаяся или погашенная десятичная точка на табло HG1.

Если между цепью +5 В и выводами 2–6, 11–13 МК подключить цепочки, состоящие из последовательно включенных резисторов 360 Ом и светодиодов АЛ307Б (анодом к +5 В), то получится оригинальная линейная шкала с пробегающей по циклу световой точкой.

Кнопка SB1 сбрасывает МК, устанавливая начальный тестовый режим с низким периодом вращения 5 с, чтобы испытуемый мог адаптировать свой слуховой аппарат и психологически настроиться на измерения.

Си-программа тестера "вращения звука" приведена в **листинге 3**.

Строки 7-9. Полезный для практики прием, когда входным или выходным линиям портов присваиваются логически понятные имена. Текст Си-программы становится легче читаемым, но увеличивается число строк.

Строки 16–20. Проверка отсутствия неконтролируемого "дребезга" контактов кнопок. Длится он обычно 20...40 мс, поэтому проводятся две проверки состояния с разницей в 50 мс (строка 17). Если повторная проверка "не прошла", то считается, что было ложное срабатывание и кнопка не нажата.

Строки 27–36. Пример прямого управления портом C, при этом его старшие неиспользуемые в схеме разряды всегда установлены в "1", чтобы, например, вход PC6 не "висел в воздухе" при коммутации.

Строка 54. Переменная "a" определяет число циклов генерации звука. Расчет длительности. В строке 13 начальное значение temp=50, следовательно, a=temp*50=50*50=2500. Длительность задержки в строке 56 составляет t=250 мкс, число каналов N=8, суммарное время T[c]=a*N*t[c]=2500*8*0,00025=5 с. Именно с таким периодом будет проходить один цикл вращения в тестовом режиме.

Строка 64. Проверка переменной "temp" с тем, чтобы после первого нажатия кнопки SB2 произошел переход сразу в режим 900 мс. Для кнопки SB3 такая проверка не нужна, поскольку она производится автоматически в строке 59.

Эксперименты с "вращающимся звуком"

Испытуемого усаживают в центре комнаты, вокруг него расставляют 8 динамиков по рис.7. Расстояние и высоту подбирают экспериментально. Включают прибор, переключателем SA1 устанавливают требуемое направление вращения звука. В начале (или после нажатия кнопки сброса SB1) проверяют слышимость сигналов в тестовом режиме. Резисторами RP1–RP8 подстраивают громкость так, чтобы звуки с разных направлений были примерно равными по уровню.

Затем кнопкой "+" постепенно увеличивают частоту вращения до тех пор, пока испытуемый не скажет, что эффект вращения стал размытым или вовсе прекратился. Показание индикатора HG1, умноженное на 100 мс, и будет означать критический для данного человека показатель. Теперь переключателем SA1 меняют направление вращения, нажимают сброс и вновь повторяют измерения. Как правило, показания будут отличаться. Связано это с разными функциями левого и правого полушарий головного мозга. Если будут зафиксированы значительные отклонения от среднестатистических показаний, то это может свидетельствовать о начинающихся нарушениях не только слухового аппарата, но и мозговой деятельности человека (предвестник болезни).

Практическое задание. Промоделировать работу электрических схем, показанных на рис.2, 5, 8, с помощью программы VMLab. Проверить результаты моделирования на реально собранных макетах устройств. Выслать разработчикам WinAVR по электронной почте свои предложению по улучшению интерфейса и расширению возможностей инструментов MFile, PN, AVRDUDE.

Листинг 3

```
//"Вращающийся звук"  =AVR, ступень 5=,  PA, №5, 2005  =1
//Make: Name=avr53, MCU=atmega8, Level=2, Debug=VMLab  =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)   =3
#include <avr/io.h>            //Библиотека ввода-вывода =4
#include <avr/delay.h>           //Библиотека задержек =5
#define F_CPU 1000000UL        //Тактовая частота 1 МГц =6
#define LR  PB0 //Условное имя для кнопки ВЛЕВО-ВПРАВО =7
#define PLUS  PB1          //Условное имя для кнопки "+" =8
#define MINUS PB2          //Условное имя для кнопки "-" =9
#define NOTA 83   //Константа для частоты звука 2 кГц =10
unsigned int a;       //Счетчик для длительности звука =11
unsigned char b, k;            //Номер канала звучания =12
unsigned char temp=50; //Начальная длительность звука =13
//---------Функция проверки нажатия кнопки----------- =14
unsigned char key (unsigned char s)  //"s"-линия порта=15
{ if (bit_is_clear(PINB,s))  //Если кнопка нажата, то =16
  { _delay_loop_2(12500);  //Пауза 50 мс, антидребезг =17
    if (bit_is_clear(PINB,s))     //Повторная проверка =18
    { return (0);  //Возврат с "0" при нажатии кнопки =19
    }                  //Окончание повторной проверки =20
  }    //Окончание процедуры обработки нажатия кнопки =21
  return (1);  //Возврат с "1" при отсутствии нажатия =22
}                           //Окончание функции "key" =23
//-------Функция свечения индикатора HG1------------- =24
void ind (unsigned char number)  //number=код символа =25
{ switch (number)       //Индикация цифр 1-9 и буквы F =26
  { case 0: PORTC=0xCE; PORTB &=~_BV(PB6); break; //F =27
    case 1: PORTC=0xF9; PORTB |= _BV(PB6); break; //1 =28
    case 2: PORTC=0xE4; PORTB &=~_BV(PB6); break; //2 =29
    case 3: PORTC=0xF0; PORTB &=~_BV(PB6); break; //3 =30
    case 4: PORTC=0xD9; PORTB &=~_BV(PB6); break; //4 =31
    case 5: PORTC=0xD2; PORTB &=~_BV(PB6); break; //5 =32
    case 6: PORTC=0xC2; PORTB &=~_BV(PB6); break; //6 =33
    case 7: PORTC=0xF8; PORTB |= _BV(PB6); break; //7 =34
    case 8: PORTC=0xC0; PORTB &=~_BV(PB6); break; //8 =35
    case 9: PORTC=0xD0; PORTB &=~_BV(PB6);        //9 =36
  }            //На HG1 индицируется выбранный символ =37
  return;                 //Возврат обратно в программу =38
}                             //Окончание функции "ind" =39
//==============ОСНОВНАЯ ПРОГРАММА=====================40
int main(void)             //Начало основной программы =41
{ PORTB=0xFF; DDRB=0xD0; //MOSI,PB6,PB7- выходы с "1" =42
  PORTC=0xFF; DDRC=0x3F;     //PC0-PC5 - выходы с "1" =43
  PORTD=DDRD=0xFF; //Все линии порта D - выходы с "1" =44
  ind (0);       //Индикация буквы F - тестовый режим =45
  while (1)                        //Бесконечный цикл =46
  { for (b=0; b < 8; b++)  //Смена 8 каналов звучания =47
    { if (bit_is_clear(PINB,LR))   //Если включен SA1 =48
      { k=b; PORTB &= ~_BV(PB7); //Горит точка на HG1 =49
      } //Установлено направление звука справа-налево =50
      else //Если отключен SA1, то направление вправо =51
      { k=7-b; PORTB |= _BV(PB7); //Точка HG1 погашена=52
      } //Установлено направление звука слева-направо =53
      for (a=temp * 50; a > 0; a--) //Длительн. звука =54
      { PORTD ^= _BV(k);       //Генерация 0-1-0-1... =55
        _delay_loop_1(NOTA);  //Пауза (частота звука) =56
      }    //Окончание генерации звука в одном канале =57
      if ((key(MINUS)) == 0) //Если нажата кнопка "-" =58
      { if (++temp > 9) temp=9; //Проверка на максимум=59
        ind (temp); //Индикация нового значения на HG1=60
        _delay_loop_2(62500);       //Пауза на 250 мс =61
      }  //Окончание действий при нажатии кнопки "-" =62
      if ((key(PLUS)) == 0)  //Если нажата кнопка "+" =63
      { if (temp == 50) temp = 10; //Начальн. нажатие =64
        if (--temp < 1) temp=1; //Проверка на минимум =65
        ind (temp); //Индикация нового значения на HG1=66
        _delay_loop_2(62500);       //Пауза на 250 мс =67
      }  //Окончание действий при нажатии кнопки "+" =68
    }          //Переход к следующему каналу звучания =69
  }     //Переход к началу бесконечного цикла "while" =70
}       //WinAVR-20050214, длина программы 496 байтов =71
```

Микроконтроллеры AVR. Ступень 6

С.М. Рюмик, г. Чернигов

Мы редко понимаем, чего в действительности хотим
Франсуа де Ларошфуко

Начиная изучение микроконтроллеров (МК), человек ставит перед собой определенные цели и задачи. Вполне может быть, что для кого-то мечтой является разработка прибора с применением многострочного жидкокристаллического индикатора (ЖКИ). О том, как управлять им с помощью МК семейства AVR, пойдет речь дальше. Приводимые в статье сведения будут полезны для сопряжения ЖКИ и с другими разновидностями МК.

рис.1

Иногда плавное течение событий надо на время изменить. Чтобы заглянуть в будущее, чтобы увидеть перспективу, чтобы уже в начале пути почувствовать свои силы.

Применительно к радиолюбительским конструкциям, их товарный вид во многом зависит от применяемого типа индикатора. Одно дело светодиоды или семисегментные матрицы и совсем другое – многострочные символьные ЖКИ. По своим функциональным возможностям они настолько поражают воображение, что их автоматически причисляют в разряд сложных изделий и изучают на завершающем этапе учебы.

Так ли это оправдано и необходимо? Предлагается провести альтернативный эксперимент и уже сейчас собрать пару конструкций на двухстрочном ЖКИ.

Выбор ЖКИ

Следует различать обычные многопозиционные ЖКИ серии ИЖКЦ, модули на их основе с микросхемой HT1611 и алфавитно-символьные ЖКИ с встроенным контроллером (**рис.1**). Именно последние и относят к наиболее перспективным изделиям. В подтверждение тому множество фирм в мире, специализирующихся на выпуске подобной продукции. Крупнейшие из них находятся в Тайване, Китае, Японии, США (**табл.1**).

Среди параметров, отличающих одни ЖКИ от других, выделяется марка внутреннего контроллера. В дальнейшем будут рассматриваться только модели, совместимые с контроллером HD44780 (фирма Hitachi) и его аналогами, например, KS0066 (фирма Samsung), SED1278 (фирма Epson), ST7066 (фирма Sitronix). Таких изделий на отечественном рынке подавляющее большинство, да и в любительских конструкциях они стали стандартом "де-факто".

При покупке ЖКИ надо поинтересоваться следующими моментами:

поддерживается ли система команд HD44780 ("Да");

имеется ли русификация знакогенератора с выводом больших и малых букв ("Да");

однополярное или двухполярное требуется питание ("Однополярное +5 В");

имеется ли подсветка (если "Да", то нужна светодиодная, а не электролюминесцентная);

сколько символов и строк отображается на экране ЖКИ ("8x1", "8x2", "16x1", "16x2", "20x2", "20x4" и т.д.).

На последний вопрос однозначного ответа нет. Все зависит от эргономических требований к разрабатываемому устройству. Если ограничиться экспериментальными работами, то приемлемым выбором по критерию "цена + возможности" является двухрядный ЖКИ с числом символом 16 в каждой строке. Это одна из наиболее ходовых моделей (а значит, и более дешевая). Кроме того, разрядность, меньшая чем 16x2, не позволит в будущем "развернуться" конструкторскому воображению, а большая разрядность накладна по финансам и габаритам.

Остальные параметры ЖКИ, такие, как угол обзора сегментов, цвет фона, контрастность, яркость подсветки, химический состав "жидкого кристалла", напрямую зависят от технологии. В общем случае, чем выше цена изделия, тем параметры лучше. Но для первых экспериментов это не главное, подойдут непритязательные ЖКИ без подсветки, например, SC1602 (фирма Sunlike), WD-C1602 (фирма Wintek), BC1602 (фирма Bolymin), MTC-16204 (фирма Microtips), PC1602 (фирма Powertip), MT-16S2 (фирма МЭЛТ) и т.д.

Внутреннее устройство ЖКИ

По определению, в состав ЖКИ входит индикатор на жидких кристаллах. Впервые жидкие кристаллы были обнаружены в 1888 г. австрийским ботаником Фридрихом Райнитцером (Friedrich Reinitzer) при изучении физических свойств некоторых органических тканей. Однако до практического применения жидких кристаллов потребовалось еще столетие. В частности, первый в мире калькулятор с ЖК-экраном выпустила фирма Sharp в 1964 г., затем появились электронные часы (1966 г., RCA) и телевизор с размером экрана по диагонали 14 см (1976 г., Sharp).

Физический принцип работы ЖКИ легко уяснить, представив элементарный конденсатор, обкладки которого заполнены специальным бесцветным веществом. При подаче на конденсатор напряжения, вещество меняет свою кристаллическую структуру и перестает пропускать свет. Визуально появляется темная окраска единичного сегмента индикации. Поскольку конденсатор имеет малые токи утечки, то и весь индикатор получается низкопотребляющим.

Достоинством символьных многострочных ЖКИ является то, что заботу о подаче требуемых напряжений на массив "ЖК-конден-

Таблица 1

Фирма	Страна	Интернет-адрес
Ampire Co. Ltd	Тайвань	http://www.ampire.com.tw
Anshan Yes	Китай	http://www.yes-lcd.com
AV-Display	Китай	http://www.av-display.com.cn
Batron (Data Modul)	Германия	http://www.data-modul.de
Bolymin	Тайвань	http://www.bolymin.com.tw
Data International Co.	Тайвань	http://www.datavision.com.tw
Hantronix	США	http://www.hantronix.com
Intech LCD Group	Китай	http://www.intechlcd.com
JE-AN	Корея	http://www.jeanlcd.co.kr
Microtips Technology	Тайвань	http://www.microtips.com.tw
Optrex	Япония	http://www.optrex.co.jp
Powertip	Тайвань	http://www.powertip.com.tw
Sunlike Display Tech. Corp.	Тайвань	http://www.lcd-modules.com.tw
Tianma Microelectronics	Китай	http://www.tianma.com
Winstar Display Co.	Тайвань	http://www.winstar.com.tw
Wintek	Тайвань	http://www.wintek.com.tw
МЭЛТ	Россия	http://www.melt.aha.ru

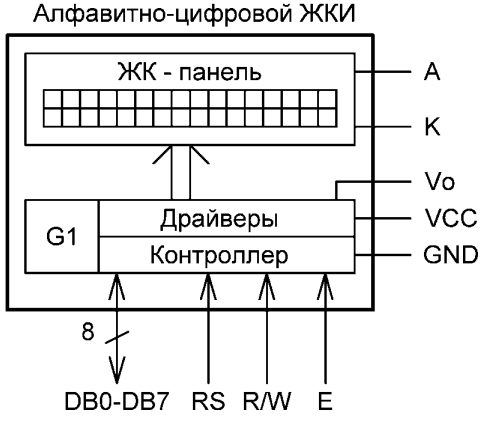

рис.2

саторов" берет на себя встроенный управляющий контроллер. На **рис.2** показана структурная схема типового ЖКИ с организацией 16x2, которая идентична для всех моделей независимо от фирмы-изготовителя. Основу составляет специализированный контроллер, обычно выполненный в виде одной или двух микросхем-"капелек", реже – в виде фирменной SMD-микросхемы. По назначению выводов и системе команд он совпадает с родоначальником серии – HD44780. Общепринятое название таких микросхем "Dot Matrix Liquid Crystal Display Controller/Driver", из чего следует их двойная функция – контроллер управляет интерфейсом, а драйвер "зажигает" сегменты.

Контроллер синхронизируется внутренним RC-генератором G1, имеющим частоту 250 ±50 кГц. Напряжение подсветки подается через выводы A и K на светодиоды, которые освещают ЖК-панель с торца или обратной стороны корпуса. Светодиоды включены матрицей и соединены параллельно-последовательно. В связи с этим напряжение подсветки довольно высокое 4,0...4,2 В.

Назначение и нумерация всех внешних выводов ЖКИ унифицированы (**табл.2**). Это не зависит от количества строк и символов, будь то "8x1" или "16x2". Даже контакты светодиодной подсветки 15, 16 имеются на всех ЖКИ, хотя при ее физическом отсутствии они будут просто "висеть в воздухе".

Небольшой нюанс. На печатной плате ЖКИ порядок нумерации контактных площадок отличается от модели к модели. Например, встречаются следующие варианты: слева направо 1-16, справа налево 16-1, вперемежку 15, 16, 1-14. Подсказку следует искать визуально по отмаркированным цифрам на печатной плате. Контакты 15, 16 обычно дублируются еще одной парой контактов с маркировкой A и K соответственно. Электрически они соединены параллельно.

Конструктивно выводы могут располагаться сверху, снизу или на боковой стороне платы ЖКИ. Это не суть важно, ведь соединяться с изделием они будут жгутом проводов длиной до 10 см. Крепление ЖКИ производится винтами через 4 угловых отверстия.

Электрический интерфейс состоит из трех шин:

DB0-DB7 шина данных;

RS, R/W, E шина управления;

VCC, GND, Vo, A, K шина питания.

Типовая схема подключения ЖКИ к МК показана на **рис.3**. Именно она и будет использоваться для первой тестовой проверки ЖКИ с выведением на экран знаменитой фразы "Hello, world!" ("Здравствуй, мир!"). Кнопка SB1 осуществляет начальный сброс. Переменным резистором R2 регулируют контрастность изображения. Его сопротивление непринципиально и может меняться от 5 до 20 кОм.

Кстати, резистор R2 является первым элементом, который надо обязательно покрутить в разные стороны при начальном включении питания. Если ЖКИ исправен, то в крайних положениях движка будут наблюдаться полное гашение и полная засветка экрана. Отрегулировать R2 следует на перегибе характеристики, как правило, с потенциалом ближе к общему проводу, когда слабо видны все точки знакомест на ЖКИ. Неправильная установка контрастности может привести к ложному выводу о дефекте индикатора, хотя все, что надо сделать, это покрутить движок резистора.

Управляющая программа хранится в МК DD1. Чтобы облегчить ее составление, здесь и в дальнейшем приняты некоторые упрощения. Во-первых, ЖКИ будет работать только на прием информации

Таблица 2

Вывод ЖКИ	Обозначение	Выполняемая функция
1	GND	Общий провод
2	VCC	Питание +5 В
3	Vo	Напряжение фокусировки
4	RS	"0" запись команд, "1" запись данных
5	R/W	"0" запись в ЖКИ, "1" чтение из ЖКИ
6	E	Перепад уровня из "1" в "0" – ввод данных
7 - 14	DB0 – DB7	Двунаправленная шина данных
15	A	Анод матрицы светодиодов подсветки
16	K	Катод матрицы светодиодов подсветки

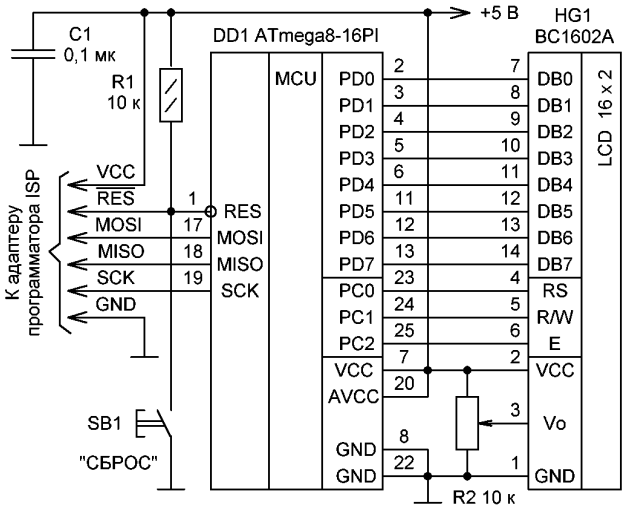

рис.3

Таблица 3

Команда ЖКИ	HEX-код	Выполняемые действия	Время вып., мкс
Очистка дисплея	0x01	Пустой экран, очистка памяти, курсор в левой верхней позиции	1640
Возврат курсора в начало	0x02	Курсор в левой верхней позиции, память не очищается	1640
Сдвиг курсора влево	0x04	После вывода очередного символа курсор автоматически сдвигается на одно знакоместо влево	40
Сдвиг курсора вправо	0x06	После вывода очередного символа курсор автоматически сдвигается на одно знакоместо вправо	40
Выключение дисплея	0x08	Полное отсутствие изображения на экране ЖКИ	40
Выключение курсора	0x0C	Разрешается вывод изображения, но курсор не виден	40
Прямоугольная форма курсора	0x0D	Разрешается вывод изображения, курсор в виде темного мигающего прямоугольника	40
Линейная форма курсора	0x0E	Разрешается вывод изображения, курсор в виде нижней подстрочной немигающей линии	40
Комплексная форма курсора	0x0F	Разрешается вывод изображения, курсор в виде мигающего прямоугольника с подчеркиванием	40
Интерфейс 4 бита, 1 строка	0x20	Связь с однострочным ЖКИ через 4 линии шины данных	40
Интерфейс 4 бита, 2 строки	0x28	Связь с двухстрочным ЖКИ через 4 линии шины данных	40
Интерфейс 8 бит, 1 строка	0x30	Связь с однострочным ЖКИ через 8 линии шины данных	40
Интерфейс 8 бит, 2 строки	0x38	Связь с двухстрочным ЖКИ через 8 линии шины данных	40
Доступ к ОЗУ знакогенератора	0x40-0x7F	Запись данных по этим адресам позволяет создать 16 своих символов	40
Установка позиции курсора	0x80--0xCF	Курсор устанавливается в позицию согласно рис.4	40

Верхняя строка ЖКИ

0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
0xC0	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7	0xC8	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF

Нижняя строка ЖКИ

рис.4

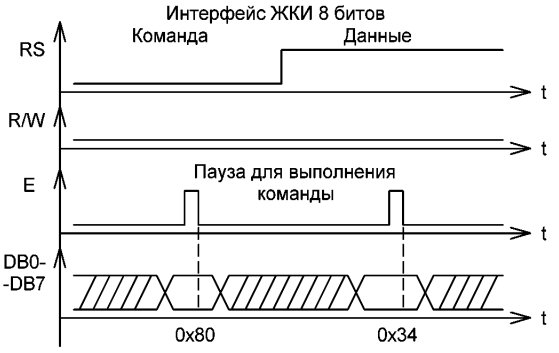

рис.5

по всем 11 соединительным линиям шины данных и управления. Во-вторых, экран ЖКИ считается жестко привязанным к начальной позиции с фиксированными адресами знакомест. В-третьих, при программировании будет использоваться ограниченный набор команд (желающие смогут в последствие расширить свои познания, изучив DATASHEET на HD44780 http://www.gaw.ru/pdf/lcd/Chips/Hitachi/hd44780u.pdf, 316 Кб).

Программное управление ЖКИ

Поскольку внутри ЖКИ находится свой собственный контроллер со своей разветвленной системой команд, то задача упрощается. Две такие мощные и интеллектуальные микросхемы, как HD44780 и ATmega8, смогут быстро между собой "договориться" на машинном языке. Труд программиста заключается в том, чтобы "объяснить" контроллерам правила общения и установить протокол соединения.

В **табл.3** показана расшифровка наиболее употребляемых команд, посылаемых от МК в ЖКИ, а на **рис.4** – распределение адресов на верхней и нижней строках экрана. Время выполнения команд указано приблизительно. Оно определяется частотой внутреннего RC-генератора ЖКИ, которая, в свою очередь, зависит от технологического разброса и температуры нагрева корпуса.

Различают команды прямого и косвенного действия. Первые из них занимают адреса 0x01-0x3F и не требуют передачи данных. За вторыми (диапазон выше 0x3F) обязательно следует передача одного или нескольких байтов информации. Для примера на **рис.5** показаны временные диаграммы выполнения команды 0x80 "Установка курсора в первое знакоместо верхней строки экрана" и индикация в нем цифры "4" пересылкой кода данных 0x34.

Формировать диаграммы, показанные на рис.5, должен МК с учетом задержек из табл.3, необходимых контроллеру ЖКИ на выполнение команд. Для повышения устойчивости работы экономить на задержках не надо. По крайней мере, при отладке программы они должны быть достаточно большими.

Каждое знакоместо на экране ЖКИ имеет свой логический адрес. Представить его можно в виде регистра, куда заносится один байт информации. В зависимости от содержимого байта на экране появляется тот или иной символ. Распределение символов соответствует таблице знакогенератора, похожей на применяемые в шрифтах компьютера.

В **листинге 1** показана Си-программа для тестовой проверки ЖКИ по схеме, собранной на рис.3.

Пояснения к листингу 1

Строки 5, 6 назначают условные имена для сигналов шины управления RS и E. Это стандартный прием на случай быстрого внесения изменений. Например, если вышел из строя вывод 25 микросхемы DD1, к которому был подключен сигнал RS, то можно перейти на другой исправный вывод (28) и в строке 5 указать "#define RS PC5". Все остальные замены по тексту листинга компилятор сделает автоматически.

Строка 7. Константа TIME определяет длительность положительного импульса сигнала E на диаграмме рис.5. Формирование других задержек организуется умножением TIME на определенные коэффициенты. Получается своеобразный регулятор быстродействия. Он понадобится, в частности, для нестандартного ЖКИ, при этом константу TIME можно увеличить в несколько раз.

Строки 9–12. Функция задержки времени сделана универсальной. При малых значениях входной переменной "a" – микросекунды, при больших (до 65535) – миллисекунды.

Листинг 1

```
//Проверка ЖКИ, =AVR. Ступень 6=  Журнал РА, №6, 2005 =1
//Make: Name=avr61, MCU=atmega8, Level=2, Debug=VMLab =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)  =3
#include <avr/io.h>          //Библиотека ввода-вывода =4
#define RS PC0     //Условное имя для сигнала RS (ЖКИ) =5
#define EN PC2     //Условное имя для сигнала E (ЖКИ) =6
#define TIME 2   //Базовая задержка при частоте 1 МГц =7
//-------------Функция задержки времени-------------- =8
void pause(unsigned int a) //"a" - длительность паузы =9
{ unsigned int cn;          //"cn" - счетчик времени =10
  for (cn=a; cn > 0; cn--);  //Цикл задержки времени =11
}                          //Окончание функции "pause" =12
//---------Функция записи команды в ЖКИ------------- =13
void lcd_com(unsigned char p)   //"p" - байт команды =14
{ PORTC &= ~_BV(RS);                    //Сигнал RS=0 =15
  PORTC |= _BV(EN);                     //Сигнал EN=1 =16
  PORTD = p;     //Вывод на шину DB0-DB7 ЖКИ команды =17
  pause(TIME);              //Длительность сигнала EN =18
  PORTC &= ~_BV(EN);     //Фронт записи команды в ЖКИ =19
  pause(5 * TIME);    //Пауза для выполнения команды =20
}                      //Окончание функции "lcd_com" =21
//---------Функция записи данных в ЖКИ-------------- =22
void lcd_dat(unsigned char p)    //"p" - байт данных =23
{ PORTC |= _BV(RS) | _BV(EN);   //Сигналы RS=1, EN=1 =24
  PORTD = p;  //Вывод на шину DB0-7 ЖКИ байта данных =25
  pause(TIME);              //Длительность сигнала EN =26
  PORTC &= ~_BV(EN);     //Фронт записи данных в ЖКИ =27
  pause(5 * TIME);    //Пауза для выполнения команды =28
}                      //Окончание функции "lcd_dat" =29
//---------Функция инициализации ЖКИ---------------- =30
void lcd_init(void)   //Режим 8 бит, мигающий курсор =31
{ lcd_com(0x08);          //Полное выключение дисплея =32
  lcd_com(0x38); pause(1000*TIME); //8 бит, 2 строки =33
  lcd_com(0x38); pause(20*TIME);   //8 бит, 2 строки =34
  lcd_com(0x38); lcd_com(0x38);    //8 бит, 2 строки =35
  lcd_com(0x01); pause(1000*TIME); //Очистка дисплея =36
  lcd_com(0x06);              //Сдвиг курсора вправо =37
  lcd_com(0x0D); //Включение дисплея, мигающий курсор=38
}                     //Окончание функции "lcd_init" =39
//==============ОСНОВНАЯ ПРОГРАММА================== =40
int main(void)            //Начало основной программы =41
{ PORTB = DDRD = 0xFF; //B=входы с резист., D=выходы =42
  PORTC = 0xF8; DDRC = 0x07; //PC0..3 выходы с лог.0 =43
  lcd_init();      //Инициализация ЖКИ (8 бит, 16x2) =44
  lcd_dat('H'); lcd_dat('e'); lcd_dat('l');  //Текст =45
  lcd_dat('l'); lcd_dat('o'); lcd_dat(',');  //Текст =46
  lcd_dat(' '); lcd_dat('w'); lcd_dat('o');  //Текст =47
  lcd_dat('r'); lcd_dat('l'); lcd_dat('d');  //Текст =48
  lcd_dat(' '); lcd_dat('!');     //Окончание текста =49
  return (0); //Успешное завершение программы "main" =50
}       //WinAVR-20050214, длина BIN-кода 292 байтов =51
```

Строки 14–21, 23–29 формируют соответственно левую и правую половину временных диаграмм, показанных на рис.5.

Строки 31–39. Без процедуры инициализации ни один ЖКИ работать не будет. Это самая важная часть листинга. Именно на процессе инициализации часто "спотыкаются" начинающие программисты. Дело в том, что в разных источниках приводятся разные варианты последовательностей команд инициализации и не все из них гарантированно будут работать с конкретным ЖКИ.

Наиболее общая процедура инициализации приведена в DATASHEET на HD44780. Функция "lcd_init" в целом повторяет ее с тем отличием, что команда полного выключения дисплея 0x08 поставлена первой, чтобы при включении питания на экране не появлялся "мусор". Здесь нет ограничений против экспериментов, главный критерий – практика.

Строки 44–49. После выполнения инициализации курсор устанавливается в крайнее слева положение в верхней строчке экрана. Следовательно, первая буква "H" будет выведена именно в это знакоместо. Далее курсор автоматически переходит на одну позицию вправо (см. строку 37) и следующая команда выведет сюда букву "e" и т.д.

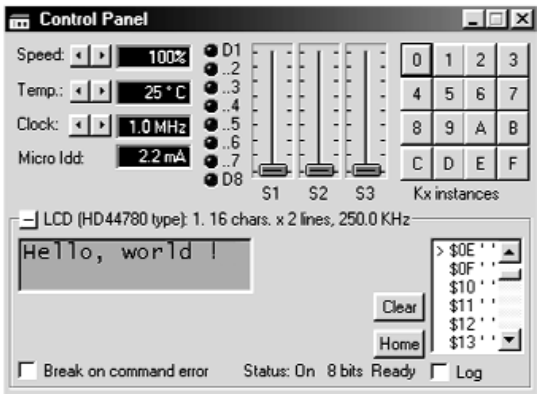

рис.6

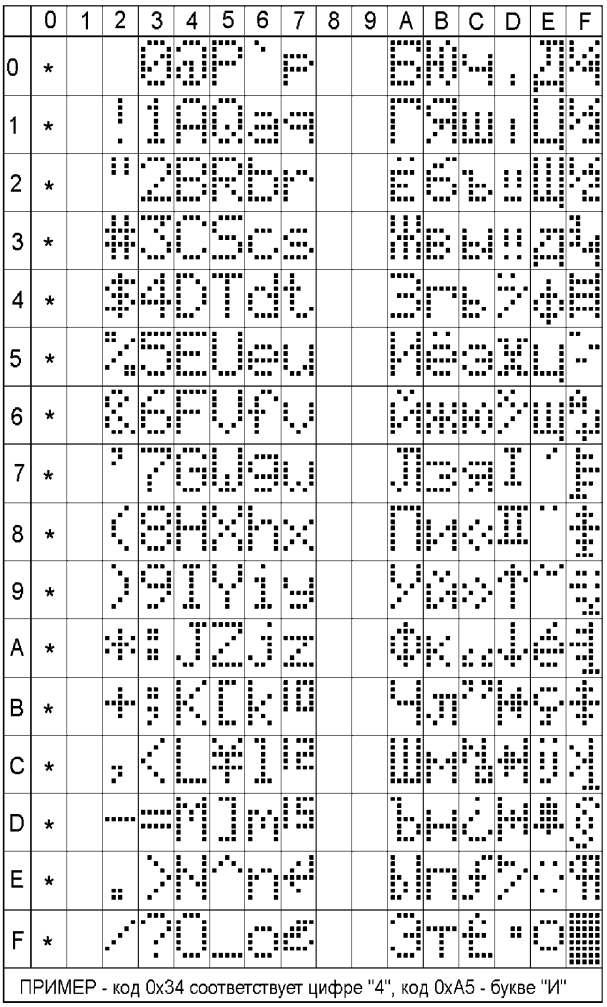

рис.7

Строка 50. Редкий случай, когда основная программа не содержит бесконечно повторяющегося цикла. Число "0" в функции "return" по традиции означает успешное завершение основной программы. Если бы потребовался аварийный выход из программы, то запись выглядела бы "return(-1)". Из-за гипотетического числа (-1) функция "main" в строке 41 имеет тип "int", а не "void".

Моделирование процессов в симуляторе VMLab

Тем, кто еще не успел приобрести ЖКИ, отчаиваться не надо. Увидеть надпись "Hello, world!", формируемую в листинге 1, можно через симулятор VMLab.

Порядок действий. Запустить на выполнение программу VMLab. Создать новый проект под названием "avr61.prj" и сохранить его в той папке, где находится программа "avr61.c" (методика описана в "Ступени 4").

Листинг 2

```
;    Симв. Стр. F   RS   R/W   E     8-битовый интерфейс ЖКИ-МК
;     ---  --  ---  ---  ---  ---  -------------------------------
X1 LCD(16  2  250K) PC0  PC1  PC2  PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0
.PLOT v(PC0) v(PC2)
```

Листинг 3

```
//===============ОСНОВНАЯ ПРОГРАММА================== =40
int main(void)      //Проверка знакогенератора 0-255 =41
{ unsigned char cifra=0, b;                 //Счетчикм =42
  PORTB = DDRD = 0xFF; //B=входы с резист., D=выходы =42
  PORTC = 0xF8; DDRC = 0x07;  //PC0-3 выходы с лог.0 =43
  lcd_init();      //Инициализация ЖКИ (8 бит, 16x2) =44
  while (1)                        //Бесконечный цикл =45
  { lcd_com(0x0C);      //Удаление видимости курсора =46
    lcd_com(0x86);    //Курсор на шестое слева место =47
    lcd_dat(cifra/100 + 0x30);                //Сотни =48
    lcd_dat((cifra/10)%10 + 0x30);          //Десятки =49
    lcd_dat(cifra%10 +0x30);                //Единицы =50
    lcd_dat('=');                    //Знак равенста =51
    lcd_dat(cifra);     //Содержимое знакогенератора =52
    for (b=0; b < 10; b++) pause(25000); //Пауза 1 с =53
    lcd_com(0x86);    //Курсор на шестое слева место =54
    for (b=0; b < 5; b++) lcd_dat(' ');    //Очистка =55
    cifra++;      //Следующий символ знакогенератора =56
  }                     //Окончание функции "while" =57
}       //WinAVR-20050214, длина BIN-кода 340 байтов =58
```

Дописать в конце проекта текст согласно **листингу 2**. Первые две строки относятся к комментариям, объясняющим формат модели ЖКИ с позиционным обозначение X1. Сокращение "LCD" – это англоязычный аналог аббревиатуры "ЖКИ". Число символов (16), число строк (2) и частота F внутреннего генератора (250 кГц) относятся к характеристикам индикатора. Далее следует перечисление линий портов МК, к которым подключаются соответственно сигналы RS, R/W, E, DB7–DB0. Замыкает проект строка виртуального осциллографа, согласно которой будут рисоваться (PLOT) графики сигналов RS и E.

Запустить симуляцию проекта. Открыть окно виртуального осциллографа "View – Scope" и панель управления "View – Control Panel". Через несколько секунд на экране осциллографа появятся изображения импульсов RS, E, а в окошке "LCD (HD44780 type)" – искомая надпись (**рис.6**).

Важная деталь. Модель ЖКИ, имеющаяся в VMLab, несколько отличается от свойств реальных изделий (или наоборот?), причем в сторону увеличения технологического запаса. Например, если повысить быстродействие ЖКИ уменьшением в 2 раза чисел "1000" в строках 33, 36 листинга 1, то экран в VMLab будет пустой, хотя реальный индикатор МТС-16204X фирмы Microtips устойчиво отображает текст.

Русификация ЖКИ

Научившись выводить на экран ЖКИ англоязычные сообщения, пора заняться переводом их на русский язык. Как известно, каждый ЖКИ имеет встроенный знакогенератор, представляющий собой область ПЗУ объемом более 8 Кб, которая прошивается на заводе-изготовителе.

Традиционно первая половина ПЗУ с адресами 00-7Fh содержит начертания цифр, знаков препинания, а также заглавных и строчных букв латинского алфавита. Все как в IBM PC. Вторая половина "отдана на откуп" национальным алфавитам. В связи с этим HD44780 имеет модификации исполнения с тремя основными вариантами зашивки знакогенератора:

латиница и европейские языки (European standard font или Euro);

латиница и японские иероглифы (Japanese standard font или Japan;)

латиница и кириллица (Custom font или Russian, **рис.7**).

Не все из ячеек знакогенератора заполнены. При обращении к "пустым" ячейкам на экране будет выведена произвольная информация, чаще всего состоящая из засвеченных точек. Первые 16 символов с адресами 0x00-0x0F отмечены "звездочкой". При желании они могут быть самостоятельно запрограммированы пользователем.

Какой знакогенератор имеется в конкретном ЖКИ, должно быть указано в его условном обозначении или в технических параметрах, хотя на практике приходится верить честному слову

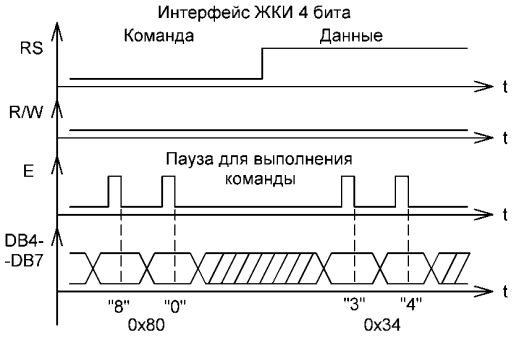

рис.8

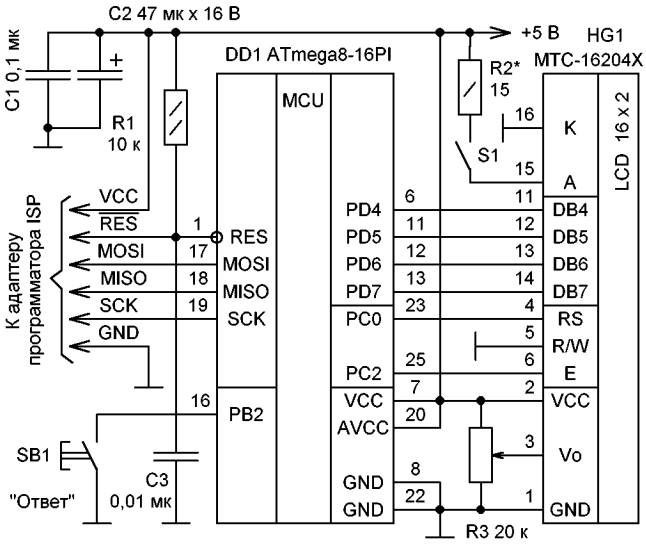

рис.9

продавца. Другой подход – вспомнить про отечественный менталитет и самому воочию увидеть на экране ЖКИ все возможные начертания символов. Для этого надо заменить строки 40–51 листинга 1 строками 40–58 **листинга 3**. Далее провести компиляцию и зашивку МК, после чего в верхней строке экрана ЖКИ будут с секундными паузами выводиться цифры десятичного адреса знакоместа 0–255 и графические образы содержащихся в них символов. Если графика и очередность появления символов соответствует рис.7, значит, ЖКИ в порядке.

Внимательный читатель, наверное, уже догадался сравнить коды знакогенератора из рис.7 со стандартными таблицами КОИ-8, а также с основной и альтернативной кодировками ГОСТа. Увы, ни одна из них не подходит под ЖКИ-кириллицу, видно про успехи нашей стандартизации в Японии и Тайване знают не очень много. В связи с этим текст, написанный в Си-программах по-русски, выглядит на русифицированном ЖКИ "по-китайски". Необходима программная перекодировка шрифта.

Если решать задачу ручным переводом, то порой возникают такие сложности, что даже опытные разработчики стараются все фразы составлять только по-английски или по-русски, но английскими буквами. К счастью, проблема довольно легко решается применением специальных утилит-перекодировщиков.

Теоретическую подкованность пора закрепить практическими действиями.

"Кибер-отгадчик"

Подобно многим научным предметам, находящимся на стыке двух дисциплин, математические фокусы не пользуются особым вниманием ни у математиков, ни у фокусников. Первые склонны рассматривать их как забаву, для вторых они кажутся не слишком эффектными. И все-таки математические фокусы имеют свой шарм, при этом стройная логика соединяется с занимательностью, а сухие цифры превращаются в зримые образы.

Особое место отводится головоломкам с абстрактными числами. Среди них, предсказание результатов математических действий, угадывание чисел, техника быстрого счета. В последнем случае, например, оказывается, что для моментального извлечения кубического корня или корня в пятой степени из 6-значных чисел вовсе не надо быть экстрасенсом или обладать феноменальной памятью [1]. Все дело в точном расчете и знании особенностей теории чисел.

Один из самых старинных фокусов с предсказанием результата заключается в том, что кого-нибудь просят задумать число, проделать над ним ряд операций и затем объявляют ответ. Если он совпадает – все выглядит как маленькое чудо или чтение мыслей на расстоянии.

С появлением электроники оформление фокусов можно разнообразить.

Исходные данные для разработки. Воспроизвести через ЖКИ диалог фокусника со зрителем по угадыванию чисел. Обеспечить кнопочное подтверждение ответа. Ширина шины данных интерфейса связи с ЖКИ – 4 бита.

Ключевым требованием при составлении электрической схемы устройства является ширина шины данных. В схеме, показанной на рис.3, она была равна 8 битам, поскольку МК и ЖКИ соединялись между собой по линиям DB0–DB7. Сейчас требуется в 2 раза уменьшить число линий связи. Сделать это можно без применения мультиплексоров, воспользовавшись специальным режимом работы контроллера HD44780.

Для активизации четырехбитового режима надо программно сформировать сигналы управления согласно временным диаграммам на **рис.8**. По структуре они совпадают с рис.5 за исключением удвоенного числа импульсов "E". Линии связи проходят через старшие разряды шины данных DB4–DB7, младшие DB0–DB3 остаются не задействованными.

Достоинство режима – малое число проводников, упрощение топологии печатной платы, экономия линий портов МК. Недостаток – пониженная скорость передачи данных в ЖКИ, так как приходится информацию передавать двумя порциями (нибблами или тетрадами) по 4 бита в каждой. Однако, учитывая обязательные задержки времени в программе и физическую инерционность "жидких кристаллов", снижение скорости почти не чувствуется.

Электрическая схема "кибер-отгадчика" показана на **рис.9**. Интерфейс МК-ЖКИ до предела упрощен. Даже сигнал R/W в МК не заводится: нет смысла, ведь все равно на нем постоянно должен присутствовать лог."0" (это следствие принятых ранее ограничений, которые гласят: "ЖКИ является только приемником данных").

Кнопка SB1 представляет собой пульт управления игрока. Нажимая ее, он подтверждает, что выполнил указания "кибера". Элементы R1, C1–C3 повышают помехоустойчивость. Переключателем S1 включается подсветка (если она имеется в ЖКИ). Яркость регулируется подбором резистора R2, чтобы протекающий через него ток не превышал 120...150 мА. "Крейсерское" значение тока 70...100 мА.

Логика работы устройства описывается тождеством: x=(y*k+x*k)/k–y, где x – предсказываемый результат, y – задуманное число, k – произвольный коэффициент.

Для упрощения вычислений значения "x" и "k" ограничивают в пределах первого десятка. Для того чтобы диалог человека с машиной не напоминал монолог птицы-говоруна, в процесс вычислений вносится элемент случайности. Для этого числа "x" и "k" генерируются случайным образом в диапазоне соответственно 1–9 и 2–9.

Расшифровка действий: "Задумайте любое число (y). Умножьте его на первое число (k). Прибавьте к результату второе число (x*k). Полученное разделите на первое число (k) и вычтите задуманное число (y)". Если вычисления проделаны без ошибок, то в результате должно получиться число "x", о чем фокусник и сообщает зрителю.

Практическое задание. Приобрести алфавитно-цифровой дисплей ЖКИ, проверить его работоспособность на тестовых программах (листинги 1, 3). Собрать "кибер-отгадчик". Программа для его прошивки будет приведена в "Ступени 7".

Литература
1. Гарднер М. Математические чудеса и тайны. – М.: Наука, 1986. – 128 с.

Микроконтроллеры AVR. Ступень 7

С.М. Рюмик, г. Чернигов

Находите время для игры, это – секрет молодости
Лотар Зайверт

В предыдущей статье цикла была приведена электрическая схема "кибер-отгадчика" (РА 6/2005, с.39, рис.9), выводящего информацию на жидкокристаллический индикатор (ЖКИ). Надеемся, что устройство успешно собрано и готово для дальнейших экспериментов.

До сих пор удавалось оберегать читателей от применения таких сложных и специфических конструкций языка Си, как указатели, классы, структуры, ассемблерные вставки. Однако минималистский подход – это не самоцель. Поэтому там, где "высшая Си-математика" нужна, она будет использоваться с кратким объяснением по сути. На практике достаточно запомнить одно правило: "Большинство программ пишутся по шаблонам".

В качестве примера рассмотрим несколько новых приемов программирования для "кибер-отгадчика" из "Ступени 6". Чтобы лучше понимать алгоритмы, необходимо в общих чертах представлять архитектуру памяти МК семейства AVR.

На **рис.1** приведена упрощенная схема, показывающая внутреннее устройство микросхемы ATmega8. Три блока (FLASH, EEPROM, фьюзы) предназначены для долговременного хранения данных. Информация в них не исчезает при выключении питания. Блок RAM – это оперативное запоминающее устройство ОЗУ, которое хранит данные только по поданном питании.

FLASH-память является самой большой по объему. В нее записываются коды управляющей программы. Те самые, которые получаются в результате компиляции. Для записи используется адаптер программатора ISP и программа PonyProg.

Фьюзы определяют аппаратную конфигурацию процессорной системы, в частности, способ подачи тактовой частоты, величину задержек, порог срабатывания детектора BOD, организацию сброса. Фьюзы также программируются через PonyProg, независимо от FLASH.

EEPROM – это отдельная область памяти, по устройству похожая на FLASH. Отличаются они не только названием, но и технологией изготовления. С точки зрения пользователя главное отличие в том, что FLASH "зашивается" только через внешний программатор, а EEPROM – дополнительно еще и от процессорного ядра МК. В итоге получается функция самопрограммирования, которая на рис.1 показана стрелкой от FLASH.

Технология FLASH появилась по времени позже, чем EEPROM и с легкой руки фирмы Intel быстро закрепилась в электронных изделиях. Ячейки FLASH-памяти занимают меньше места на кристалле и стоимость их изготовления ниже. Другое дело, что запись и стирание информации в них производится целыми блоками. Поэтому в ATmega8 оставлена небольшая часть памяти EEPROM, допускающая относительно быстрый доступ к отдельным байтам и имеющая на порядок выше число циклов перезаписи (100 тысяч).

RAM представляет собой быстродействующее статическое ОЗУ, в котором хранятся константы и переменные Си-программы, а также результаты промежуточных расчетов. Программисту полезно запомнить объем RAM в байтах, чтобы примерно ориентироваться, какой максимальной величины массив чисел можно открывать в программах. Например, в ATmega8 массивы должны иметь размерность до 1000 элементов.

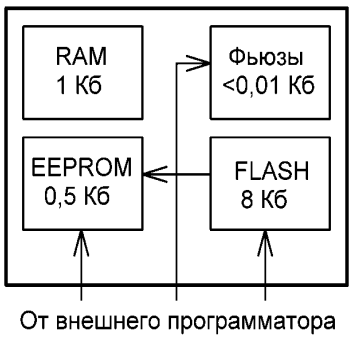

рис.1

Листинг 1

```
//Кибер-отгадчик, =AVR, ступень 7=, Радіоаматор, №7, 2005 =1
//Make: Name=avr71, MCU=atmega8, Level=2, Debug=VMLab     =2
//Фьюзы: BODEN=BODLEVEL=SUT0=CKSEL3=CKSEL2=CKSEL1="0" 1МГц=3
#include <avr/io.h>             //Библиотека ввода-вывода =4
#include <avr/eeprom.h>      //Библиотека работы с EEPROM =5
#include <stdlib.h>       //Библиотека стандартных утилит =6
#define E unsigned __attribute__ ((section(".eeprom"))) //=7
#define RS PC0         //Условное имя для сигнала RS (ЖКИ) =8
#define EN PC2         //Условное имя для сигнала E (ЖКИ) =9
#define SB PB2  //Условное имя линии для кнопки SB1 (МК) =10
#define TIME 10     //Базовая задержка при частоте 1 МГц =11
static E char t0[]="=Кибер-отгадчик==Журнал РА-7/05="; //=12
static E char t1[]="Задумайте число.  Умножьте на   "; //=13
static E char t2[]="  Прибавьте       Разделите на   "; //=14
static E char t3[]=" Отнимите заду-  манное число.  "; //=15
static E char t4[]="У вас получилось   !!! Угадал ? "; //=16
//---------------Функция задержки времени---------------- =17
void pause(unsigned int a)     //"a" - длительность паузы =18
{ unsigned int cn;              //"cn" - счетчик времени =19
  for (cn=a; cn > 0; cn--);      //Цикл задержки времени =20
}                             //Окончание функции "pause" =21
//-----------Функция записи команды в ЖКИ--------------- =22
void lcd_com(unsigned char p)        //"p" - байт команды =23
{ PORTC &= ~_BV(RS);                        //Сигнал RS=0 =24
  PORTC |= _BV(EN);                         //Сигнал EN=1 =25
  PORTD &= 0x0F; PORTD |= (p & 0xF0);    //Старший ниббл =26
  pause(TIME);                          //Длительность сигнала EN =27
  PORTC &= ~_BV(EN);   //EN=0, фронт записи команд в ЖКИ =28
  pause(TIME);                          //Длительность сигнала EN =29
  PORTC |=  _BV(EN);                          //Сигнал EN=1 =30
  PORTD &= 0x0F; PORTD |= (p << 4);      //Младший ниббл =31
  pause(TIME);                          //Длительность сигнала EN =32
  PORTC &= ~_BV(EN);   //EN=0, фронт записи команд в ЖКИ =33
  pause(5 * TIME);        //Пауза для выполнения команды =34
}                          //Окончание функции "lcd_com" =35
//-----------Функция записи данных в ЖКИ---------------- =36
void lcd_dat(unsigned char p)         //"p" - байт данных =37
{ PORTC |= _BV(RS) | _BV(EN);       //Сигналы RS=1, EN=1 =38
  PORTD &= 0x0F; PORTD |= (p & 0xF0);    //Старший ниббл =39
  pause(TIME);                          //Длительность сигнала EN =40
  PORTC &= ~_BV(EN);   //EN=0, фронт записи данных в ЖКИ =41
  pause(TIME);                          //Длительность сигнала EN =42
  PORTC |= _BV(EN);                           //Сигнал EN=1 =43
  PORTD &= 0x0F; PORTD |= (p << 4);      //Младший ниббл =44
  pause(TIME);                          //Длительность сигнала EN =45
  PORTC &= ~_BV(EN);   //EN=0, фронт записи данных в ЖКИ =46
  pause(5 * TIME);        //Пауза для выполнения команды =47
}                          //Окончание функции "lcd_dat" =48
//------------Функция инициализации ЖКИ----------------- =49
void lcd_init(void)       //Режим 4 бит, мигающий курсор =50
{ lcd_com(0x33); pause(500*TIME);             //Подготовка =51
  lcd_com(0x32); lcd_com(0x28);         //4 бит, 2 строки =52
  lcd_com(0x08);                  //Полное выключение дисплея =53
  lcd_com(0x01); pause(1000*TIME);     //Очистка дисплея =54
  lcd_com(0x06);                       //Сдвиг курсора вправо =55
  lcd_com(0x0D);    //Включение дисплея, мигающий курсор =56
}                          //Окончание функции "lcd_init" =57
//==============ОСНОВНАЯ ПРОГРАММА==================== =58
int main(void)                  //Начало основной программы =59
{ unsigned char a, b, c;                        //Счетчики =60
  unsigned int d=0, e=0;    //Генераторы случайных чисел =61
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =62
  PORTC = 0xF8; DDRC = 0x07;    //PC0...3 выходы с лог.0 =63
  lcd_init();          //Инициализация ЖКИ (4 бит, 16x2) =64
  while (1)                          //Бесконечный цикл =65
  { for (b=0; b<5; b++)   //Перебор текстов из пяти фраз =66
    { for (lcd_com(0x80), a=0; a<32; a++) //Первая строка=67
      { if (a==16) lcd_com(0xC0);        //Вторая строка =68
        if (b==0) lcd_dat(eeprom_read_byte(&t0[a])); //1 =69
        if (b==1) lcd_dat(eeprom_read_byte(&t1[a])); //2 =70
        if (b==2) lcd_dat(eeprom_read_byte(&t2[a])); //3 =71
        if (b==3) lcd_dat(eeprom_read_byte(&t3[a])); //4 =72
        if (b==4) lcd_dat(eeprom_read_byte(&t4[a])); //5 =73
      }  //Окончание вывода текста в 2 строки по 16 букв =74
      if ((b==1) || (b==2)) //Вывод чисел к фразам 2 и 3 =75
      { lcd_com(0xCE); lcd_dat(d + 0x30); //Умн., деление=76
      }     //Окончание вывода чисел умножения и деления =77
      if (b==2)       //Вывод числа прибавления к фразе 3 =78
      { c=d * e; lcd_com(0x8C);       //Установка курсора =79
        lcd_dat(c/10 + 0x30); lcd_dat(c%10 + 0x30);    //=80
      }              //Окончание вывода числа прибавления =81
      if (b==4)      //Вывод числа предсказания к фразе 5 =82
      { lcd_com(0xC1); lcd_dat(e + 0x30); //Курсор, дата =83
      }             //Окончание вывода числа предсказания =84
      while(bit_is_set(PINB,SB))  //Ждать нажатия кнопки =85
      { if (b==0)  //Начальная генерация случайных чисел =86
        { d=rand()%8 + 2; e=rand()%9 + 1; //d=2-9, e=1-9 =87
        }          //Окончание генерации случайных чисел =88
      }         //Окончание процедуры нажатия кнопки SB1 =89
      pause(65000); pause(65000);   //Пауза антидребезга =90
    }                //Переход к следующей из пяти фраз =91
  }                 //Переход к новому предсказанию числа =92
} //WinAVR-20050214, FLASH 1240 байтов, EEPROM 165 байтов=93
```

Из всех перечисленных областей памяти только EEPROM не встречалась ранее в программах. Восполнить пробел позволяет "кибер-отгадчик", программа которого приведена в **листинге 1**.

Пояснения к программе

Строка 3. Настройка фьюзов на работу от внутреннего генератора частотой 1 МГц. Новинка – обнуляются фьюзы BODEN и BODLEVEL, вследствие чего включается в работу детектор просадок напряжения питания BOD. Теперь сброс МК будет происходить при каждом снижении питания ниже 4 В (технологический разброс 3,7...4,5 В).

Почему раньше, когда в программах была задействована только FLASH-память, включение детектора BOD было не обязательным? Дело в том, что МК не может самостоятельно перепрограммировать свой FLASH (наводки на входы адаптера ISP не в счет). А вот область EEPROM он перепрограммировать может, причем даже при очень низких напряжениях питания 2...2,7 В.

Чтобы МК случайно не "запортил" данные в EEPROM при медленном нарастании или спаде питания, применяется детектор BOD. Если бы разрабатываемое устройство эксплуатировалось в заводских условиях при мощных промышленных помехах, то не помешала бы и внешняя микросхема-супервизор КР1171СП42, обеспечивающая надежный сброс МК по выводу RESET.

Строки 5, 6 – подключение двух новых системных библиотек. Первая из них обслуживает все, что связано с EEPROM в строках 7, 12–16, 69–73. Вторая библиотека нужна для одной-единственной системной функции "rand" в строке 87, которая обеспечивает генерацию псевдослучайных чисел в диапазоне 0–7FFFh.

Строка 7 содержит длинное макроопределение для буквы "E". Выражение, которое оно замещает, подставляется при компиляции в строках 12–16. Это типичный прием в языке Си по сокращению листинга и улучшению его читаемости. Вместо "E" могла быть другая буква, что не принципиально.

С физической точки зрения текстовые массивы t0[ ]–t4[ ], определенные с параметром "E", будут размещаться не в FLASH, а в EEPROM. Зачем? С целью экономии места в памяти. В данном учебном листинге это не столь критично, однако если представить, что FLASH заполнено на 99%, то лишние полкилобайта памяти сродни "палочке-выручалочке".

Строки 23–35, 37–48, 50–57 аналогичны одноименным функциям из листинга 1 "Ступени 6", но для 4-битового интерфейса связи. Функция инициализации, как и прежде, является главной. Ее текст насыщен паузами, а последовательность расстановки команд напоминает заклинания шамана, поскольку явная логика не прослеживается. Однако примерно такой алгоритм указан в DATASHEET на микросхему HD44780, являющуюся главной в применяемом ЖКИ. Абсолютные значения времени пауз выбраны с запасом по сравнению с DATASHEET, чтобы нормально проходило моделирование в среде VMLab.

Строка 66. Общение человека с "кибером" происходит с помощью пяти последовательно генерируемых фраз. Текст располагается в верхней и нижней частях экрана. Поменять его можно в строках 12–16. При этом надо знать, что первые 16 символов массивов t0[ ]–t4[ ] индицируются в верхней, а последние 16 символов – в нижней строчке.

Строки 67–74. Стандартный прием полной загрузки текста в ЖКИ. Начальное положение курсора (по-другому, начальный адрес) устанавливается только один раз в строке 67 ("lcd_com"), причем для сокращения места оператор включен прямо в состав функции "for". Дальнейший вывод текста происходит автоматически слева направо, сверху вниз.

Строки 69–73. С помощью библиотечной функции "eeprom_read_byte" читается один байт данных из EEPROM. Поскольку его абсолютный адрес не указан, то применяют так называемые "указатели" в виде знака "&" перед текстовыми массивами t0[ ]–t4[ ]. Указатели относят к базовым понятиям в языке Си. Они достаточно сложные для "объяснения на пальцах". Поэтому на первых порах надо просто запомнить шаблон и знать, что применяется он, в частности, при обращении к текстовым массивам в функции "eeprom_read_byte".

Строки 75–84. После вывода на экран фраз в строках 13, 14, 16, их надо дополнить цифрами умножения, деления и сложения. Процедура простая: в начале указывается местоположение на экране курсора ("lcd_com"), а затем код символа ("lcd_dat") согласно табл.3 в "Ступени 6" (РА 6/2005, с.37).

Строки 85–89. Процесс опроса кнопки SB1 совмещен с генерацией случайных чисел. Однако производится он только при фразе 1, во время экранной заставки. Строго говоря, функция "rand" в строке 87 возвращает псевдослучайное (почти случайное) число, но, благодаря циклическому выполнению функции "while" и произвольному во времени моменту нажатия кнопки SB1, префикс "псевдо" исчезает.

Строка 90 определяет задержку времени, учитывающую "дребезг" контактов кнопки SB1, иначе фразы могут слишком быстро проскакивать по экрану.

Строка 93. В комментариях указывается, что программировать следует две области памяти: FLASH и EEPROM. Как это сделать, будет рассказано дальше.

Симуляция "Кибер-отгадчика" в VMLab и перекодирование текста

Перед дальнейшей работой надо создать на жестком диске отдельную папку, в которую поместить файл "avr71.c" (листинг 1) и стандартный make-файл, созданный программой MFile.

Моделирование разработанного устройства в VMLab проводится по той же методике, что и для листинга 1 в "Ступени 6". Единственное, что в файле проекта "avr71.prj" надо заменить строку, отвечающую за ЖКИ, по образцу: "X1 (16 2 250K) PC0 PC1 PC2 PD7 PD6 PD5 PD4 nc3 nc2 nc1 nc0".

В результате моделирования на экране симулятора появится надпись начальной заставки (**рис.2**). Необычность ситуации в том, что фраза написана по-русски, а ведь текст еще не перекодирован! Получает-

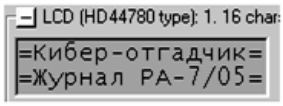

рис.2

Листинг 2

```
//Бегущий текст анонса, =AVR. Ступень 7=, РА, №7, 2005 г. =1
//Make: Name=avr72, MCU=atmega8, Level=2, Debug=VMLab     =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)      =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
#include <avr/pgmspace.h>     //Библиотека для "sigeof()" =5
#define TEMP 7000      //Скорость движения бегущей строки =6
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =7
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =8
extern void lcd_init(void);           //Инициализация ЖКИ =9
extern void pause(unsigned int p);       //Пауза для ЖКИ =10
unsigned char t0[]="               =Кибер-отгадчик= Правила\
 игры. Вы задумываете любое число и выполняете над ним прос\
тые арифметические действия, которые просит <кибер>. Фокус \
в том, что <кибер> угадает результат! Вы готовы? Нажмите кн\
опку.              ";              //Текст верхней строки =15
unsigned char t1[]="=Журнал РА-7/05=";   //Нижняя строка =16
//================ОСНОВНАЯ ПРОГРАММА=================== =17
int main(void)               //Начало основной программы =18
{ unsigned char a, b, c;                    //Счетчики =19
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =20
  PORTC = 0xF8; DDRC = 0x07;    //PC0...3 выходы с лог.0 =21
  lcd_init();          //Инициализация ЖКИ (4 бит, 16x2) =22
  lcd_com(0x0C);               //Выключение курсора ЖКИ =23
  for (lcd_com(0xC0), c=0; c<16; c++)    //Нижняя строка =24
  { lcd_dat(t1[c]);            //Вывод текущего символа =25
  }       //Окончание вывода 16 символов в нижней строке =26
  while (1)                          //Бесконечный цикл =27
  { for (b=0; b < (sizeof(t0)-16); b++) //Смещение текста=28
    { for (lcd_com(0x80), c=0; c<16; c++) //Верхн.строка =29
      { lcd_dat(t0[c+b]);       //Вывод текущего символа =30
      } //Окончание вывода 16 символов в верхней строке =31
      for (a=0; a<20; a++) pause(TEMP);       //Скорость =32
    }   //Переход к смещению текста на один символ влево =33
  }     //Переход к новому циклу движения бегущей строки =34
}              //WinAVR-20050214, длина кода 594 байтов =35
```

ся, что модель ЖКИ в VMLab подстраивается под текущий шрифт Windows и показывает то, что хотелось бы видеть пользователю, а не то, что на самом деле имеется в знакогенераторе ЖКИ. С другой стороны, это облегчает симуляцию программы, поскольку все надписи легко читаются.

Вывод – симуляцию процессов в VMLab надо проводить до перекодирования текста, т.е. сразу после составления листинга.

После завершения работы с VMLab наступает этап перекодирования шрифта и окончательной компиляции программы.

Листинг 3

```
//Файл "lcd.c", набор функций ЖКИ. =Ступень 7=, РА-7/2005 =1
//Интерфейс 4-бита по линиям PD4...PD7, WinAVR-20050214   =2
#include <avr/io.h>             //Библиотека ввода-вывода  =3
#define RS PC0          //Условное имя для сигнала RS (ЖКИ) =4
#define EN PC2           //Условное имя для сигнала E (ЖКИ) =5
#define TIME 10                 //Базовая задержка для ЖКИ  =6
//---------------Функция задержки времени---------------  =7
void pause(unsigned int p)     //"p" - длительность паузы  =8
{ unsigned int cn;               //"cn" - счетчик времени  =9
  for (cn=p; cn > 0; cn--);       //Цикл задержки времени =10
}                              //Окончание функции "pause" =11
//-------Общая часть функций lcd_com, lcd_dat----------- =12
void lcd(unsigned char p) //"p" - байт данных или команд =13
{ PORTD &= 0x0F; PORTD |= (p & 0xF0);    //Старший ниббл =14
  pause(TIME);                  //Длительность сигнала EN =15
  PORTC &= ~_BV(EN);   //EN=0, фронт записи данных в ЖКИ =16
  pause(TIME);                  //Длительность сигнала EN =17
  PORTC |= _BV(EN);                          //Сигнал EN=1 =18
  PORTD &= 0x0F; PORTD |= (p << 4);      //Младший ниббл =19
  pause(TIME);                  //Длительность сигнала EN =20
  PORTC &= ~_BV(EN);   //EN=0, фронт записи данных в ЖКИ =21
  pause(5*TIME);          //Пауза для выполнения команды =22
}                               //Окончание функции "lcd" =23
//-----------Функция записи команды в ЖКИ--------------- =24
void lcd_com(unsigned char p)       //"p" - байт команды =25
{ PORTC &= ~_BV(RS); PORTC |= _BV(EN);      //RS=0, EN=1 =26
  lcd(p);   //Вызов общей части функций lcd_com, lcd_dat =27
}                            //Окончание функции "lcd_com" =28
//-----------Функция записи данных в ЖКИ---------------- =29
void lcd_dat(unsigned char p)        //"p" - байт данных =30
{ PORTC |= _BV(RS) | _BV(EN);               //RS=1, EN=1 =31
  lcd(p);   //Вызов общей части функций lcd_com, lcd_dat =32
}                            //Окончание функции "lcd_dat" =33
//------------Функция инициализации ЖКИ----------------- =34
void lcd_init(void)       //Режим 4 бит, мигающий курсор =35
{ lcd_com(0x33); pause(500*TIME);           //Подготовка =36
  lcd_com(0x32); lcd_com(0x28);        //4 бит, 2 строки =37
  lcd_com(0x08);             //Полное выключение дисплея =38
  lcd_com(0x01); pause(1000*TIME);     //Очистка дисплея =39
  lcd_com(0x06);                  //Сдвиг курсора вправо =40
  lcd_com(0x0D);    //Включение дисплея, мигающий курсор =41
}                            //Окончание функции "lcd_init" =42
```

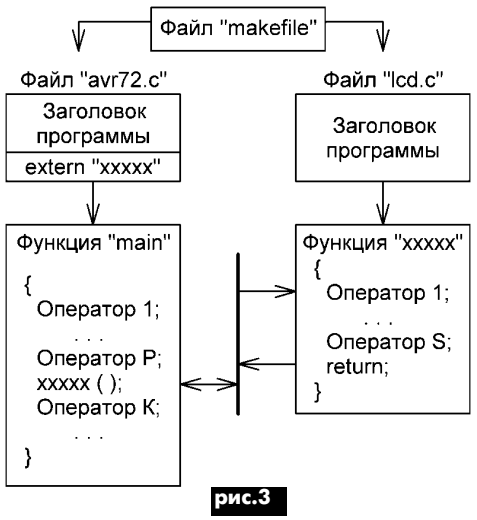

рис.3

Если этого не сделать, то реально видимые фразы в ЖКИ станут похожими на "китайскую грамоту".

Порядок действий. Скопировать с сайта РА авторскую утилиту "ruslcd.exe" в папку, где размещен файл "avr71.c". Набрать в командной строке "ruslcd avr71.c", после чего все русскоязычные фразы в строках 12–16 листинга 1 будут заменены "абракадаброй" с точки зрения IBM PC и нормальным текстом с точки зрения ЖКИ. Далее проводится стандартная компиляции программы редактором Programmers Notepad через меню "Tools – [WinAVR] Make All".

Надписи русскоязычных комментариев после обработки текста утилитой "ruslcd.exe" остаются без изменений, поэтому ее можно использовать и в других Си-программах для самых разных семейств МК.

Программирование EEPROM в PonyProg

После компиляции перекодированной программы "avr71.c", как обычно, появится файл "avr71.hex", в котором содержится информация для записи FLASH. Параллельно с этим в той же папке будет создан файл "avr71.eep", который также содержит HEX-коды, но уже для записи в EEPROM. Логично, что методика программирования МК немного изменится.

Порядок действий. Запустить на выполнение программу PonyProg. Загрузить файл "avr71.hex" через меню "File – Open Program (FLASH) File – <выбрать расширение *.hex> – <указать "avr71.hex">". Загрузить файл "avr71.eep" через меню "File Open Data (EEPROM) File – <выбрать расширение *.eep> – <указать "avr71.eep">". Открыть закладку "Command – Program Option" и к уже установленным "галочкам" возле пунктов "Reload Files", "Erase", "Write Program memory (FLASH)" поставить "галочку" в окне "Write Data memory (EEPROM)". Вернуться в начальное меню и нажатием клавиш <Ctrl>+<P> запрограммировать сразу обе области памяти МК.

ЖКИ с бегущей строкой

Бегущая строка обычно ассоциируется с рекламными титрами. Но ее также с успехом можно использовать для вывода большой по объему информации на малоразмерный экран. В качестве примера предлагается добавить в "кибер-отгадчик" бегущую строку с пояснениями правил игры.

Программное обеспечение для удобства разбито на 2 части (**листинги 2, 3**). Здесь реализована архитектура третьего типа по классификации, рассмотренной ранее для МК семейства MCS-51. На **рис.3** показаны взаимосвязи основной программы "avr72.c" (листинг 2), блока внешних функций "lcd.c" (листинг 3) и управляющего make-файла. Символом "xxxxx" обозначены вызываемые функции. Такой подход позволяет использовать файл "lcd.c" без изменений и в других программах.

Пояснения к листингу 2

Строка 5. Подключение новой системной библиотеки "pgmspace", отвечающей за обслуживание массивов памяти.

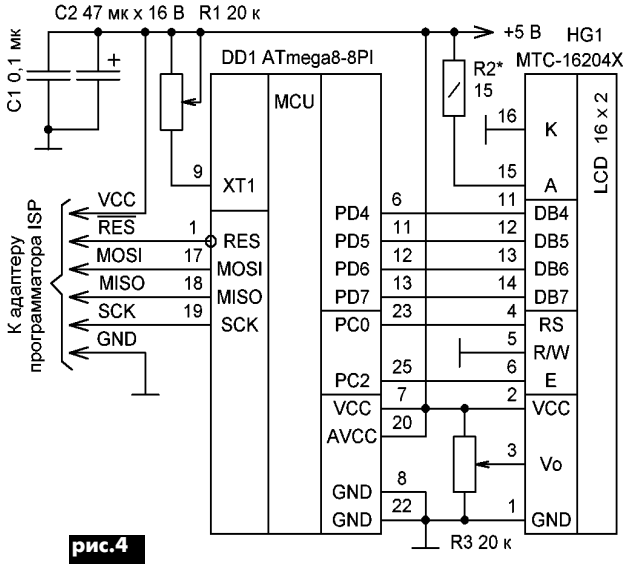

рис.4

Строки 7–10. Указание на то, что все функции для работы с ЖКИ находятся во внешнем файле (ключевое слово – "extern"). В каком именно, должно быть указано в make-файле.

Строки 11–14. Текстовый массив t0[ ] содержит настолько много элементов, что они не помещаются в одной строке. Для переноса используется знак "обратная косая черта" (\). Элементом массива он не является.

Строка 15. В конце массива поставлены пробелы. Это помогает визуально отделить окончание одного информационного блока от начала следующего повтора. Бегущая строка "бежит" по кругу.

Строка 28. Функция "sizeof(t0)" автоматически подсчитывает общее количество элементов в массиве "t0[ ]". Тем самым не приходится вручную пересчитывать их в строках 11–15. Кроме того, теперь можно добавлять или уменьшать число символов в массиве, ничего больше не корректируя в программе.

Строка 30. Сдвиг символов в бегущей строке производится изменением переменной "b". Физически можно представить себе "окно" шириной 16 знакомест, которое последовательно перемещается по всему массиву t0[ ]. Что попадает в данный момент в "окно", то и видно на экране ЖКИ.

Строка 32. Задержка времени необходима для того, чтобы пользователь смог прочитать появляющийся текст. Для изменения скорости движения бегущей строки надо подкорректировать константу TEMP в строке 6.

Пояснения к листингу 3

Текст листинга следует оформить в виде файла "lcd.c" и поместить его в одну папку с файлом "avr72.c".

Функции "lcd_com", "lcd_dat", "lcd_init", "pause" повторяют по смыслу одноименные функции из листинга 1. Однако для сокращения 80 байтов программы введена вспомогательная функция "lcd", общая для "lcd_dat" и "lcd_com".

Особенности компиляции

Поскольку программа использует не один, а два Си-файла, то этот момент надо отразить в структуре make-файла.

Порядок действий. Открыть программу MFile и стандартным способом заполнить графы для файла "avr72.c". Сохранить "makefile" в той папке, где находятся файлы "avr72.c" и "lcd.c". Запустить на выполнение редактор Programmers Notepad. Открыть в нем только что созданный "makefile". Сделать видимыми номера строк: "View – Line Numbers". Добавить в строке 63 указание на файл "lcd.c" по образцу "SRC = $(TARGET).c lcd.c". Сохранить изменения: "File – Save All".

Переконвертировать русский текст "avr72.c" программой "rusIcd.exe". Для удобства можно создать пакетный файл "avr72.bat" с одной-единственной строкой "rusIcd avr72.c" Откомпилировать программу: "Tools – [WinAVR] Make All". После этого в текущей папке появятся файлы для прошивки FLASH "avr72.hex" и EEPROM "avr72.eep". Далее – стандартное программирование через PonyProg.

"Электронная визитка"

Говорят, что по содержанию визитной карточки можно судить о характере ее владельца. Чтобы сделать бумажный вариант визитки, достаточно воспользоваться средствами любого текстового или графического редакторов. Существует даже специальная дизайнерская программа "Визитка" (фирма "Графика-М" http://www.vizit-ka.ru/rus/soft.html, 14 Мб, бесплатно), позволяющая создавать целостные художественные композиции.

Электронное устройство тоже может стать визитной карточкой, будь-то радиолюбителя или фирмы, связанной с электроникой. Главное, чтобы оно запомнилось необычным внешним видом, а заодно давало бы полезную информацию о субъекте.

На **рис.4** показана схема "электронной визитки" с использованием ЖКИ. После включения питания на табло появляется бегущая строка с реквизитами владельца.

Основой устройства является МК DD1, связанный с одно- или двухстрочным ЖКИ HG1 с помощью 4-битового интерфейса данных.

Особенность конструкции – плавное изменение скорости бегущей строки вплоть до ее полной остановки переменным резистором R1. Он входит в частотозадающую цепь тактового генератора МК и регулирует в широких пределах его частоту. На вопрос: "А где же времязадающий конденсатор?" следует ответ: "Он находится внутри микросхемы DD1 и имеет емкость 36 пФ".

В нижнем по схеме положении движка резистора R1 генерация срывается, но изображение на ЖКИ не исчезает, а останавливается, что удобно для оперативного просмотра информации, например, при запоминании адреса Email.

ЖКИ желательно взять с подсветкой, чтобы внешний вид издалека привлекал внимание. Яркость подсветки определяется резистором R2. Чем меньше его сопротивление, тем яркость больше, однако нельзя превышать допустимый ток, который у разных моделей ЖКИ разный (ориентировочно не более 120...180 мА).

Подстроечный резистор R3 регулирует контрастность изображения. Его сопротивление 20 кОм взято из ряда, принятого за рубежом. Отечественные переменные резисторы имеют сопротивление 22 кОм.

Область применения "электронной визитки" – реклама на выставках, в офисах, в отделах маркетинга, в фирменных магазинах, в точках продаж на радиорынках.

Управляющая программа "avr73.c" (**листинг 4**) содержит ссылки на функции из файла "lcd.c", следовательно, они должны компилироваться вместе, по приведенной выше методике.

Пояснения к листингу 4

Строка 3. Фьюзы настраиваются на работу от внешнего RC-генератора в диапазоне 0,1...0,9 МГц. При регулировании резистора R1 частота генерации может получаться и выше предела, но на устойчивости это не отражается. После программирования фьюзов надо помнить, что для перехода, например, обратно на внутренний генератор не следует отпаивать резистор R1. При сбоях в программировании надо попробовать установить его номинал близким к 10 кОм, чтобы частота тактового генератора стала 1 МГц.

Строка 14. Перед словом "РАДІОАМАТОР" сделаны пробелы для визуального выделения блока информации.

Строки 27–29 задают паузу в несколько секунд, чтобы можно было прочитать начальную заставку. Этого момента не было в листинге 2, поскольку там текст не останавливался.

Футбольный мультфильм

Алфавитно-цифровые ЖКИ, по определению, выводят на

Листинг 4

```
//Бегущая строка, =AVR. Ступень 7=, Радіоаматор, №7, 2005 =1
//Make: Name=avr73, MCU=atmega8, Level=2, Debug=VMLab     =2
//Фьюзы: SUT0=CKSEL3=CKSEL1=CKOPT="галочки", RC-генератор =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
#include <avr/pgmspace.h>     //Библиотека для "sigeof()" =5
#define TEMP 10000     //Скорость движения бегущей строки =6
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =7
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =8
extern void lcd_init(void);           //Инициализация ЖКИ =9
extern void pause(unsigned int p);       //Пауза для ЖКИ =10
unsigned char ra[]="==РАДІОАМАТОР== Адрес редакции-г.Киев,у\
л.Краковская,36/10. Почта: а/я-50,03110,г.Киев-110,Украина.\
Тел.(044)573-39-38. Email: redactor@sea.com.ua, http://www.\
ra-publish.com.ua         ==РАДІОАМАТОР==";      //Текст =14
//================ОСНОВНАЯ ПРОГРАММА=================== =15
int main(void)                 //Начало основной программы =16
{ unsigned char a, b, c;                        //Счетчики =17
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =18
  PORTC = 0xF8; DDRC = 0x07;    //PC0...3 выходы с лог.0 =19
  lcd_init();          //Инициализация ЖКИ (4 бит, 16x2) =20
  lcd_com(0x0C);                //Выключение курсора ЖКИ =21
  while (1)                              //Бесконечный цикл =22
  { for (b=0; b < (sizeof(ra)-16); b++) //Смещение текста=23
    { for (lcd_com(0x80), c=0; c<16; c++)  //Одна строка =24
      { lcd_dat(ra[c+b]);  //Вывод символов одной строки =25
      }           //Окончание вывода 16 текущих символов =26
      if (b==0)       //Если выведена начальная заставка =27
      { for (a=0; a<70; a++) pause(TEMP); //Начал. пауза =28
      }         //Окончание начальной паузы для заставки =29
      for (a=0; a<20; a++) pause(TEMP);       //Скорость =30
    }   //Переход к смещению текста на один символ влево =31
  }     //Переход к новому циклу движения бегущей строки =32
}               //WinAVR-20050214, длина кода 540 байтов =33
```

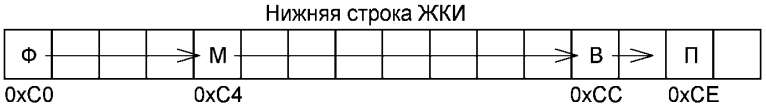

рис.5

экран буквы, цифры, знаки и спецсимволы. О графических функциях речь не идет. Тем не менее такая возможность существует.

Если вспомнить раскладку знакогенератора ЖКИ (РА 6/2005, с.38, рис.7), то первые 16 символов с адресами 0x00–0x0F оставлены пустыми. В них программист может занести изображения своих фигур размером 5x8 точек и воспроизвести их затем на любом знакоместе ЖКИ. Если в качестве фигур использовать стилизованные изображения человечков и геометрических объектов, а затем их расставлять по определенному сценарию в клетки ЖКИ, то вполне можно создать иллюзию "мультфильма". Алгоритм управления фигурами можно заимствовать из "бегущей строки".

Листинг 5

```
//Мультфильм "ПЕНАЛЬТИ", =AVR. Ступень 7=, РА, №7, 2005 г =1
//Make: Name=avr74, MCU=atmega8, Level=2, Debug=VMLab     =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="галочки" (1 МГц)      =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
#include <avr/pgmspace.h>     //Библиотека для "sigeof()" =5
#define TEMP 6000             //Скорость движения фигурок =6
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =7
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =8
extern void lcd_init(void);           //Инициализация ЖКИ =9
extern void pause(unsigned int p);       //Пауза для ЖКИ =10
unsigned char t0[]="  ==Пенальти==                  "; //=11
unsigned char t1[]="  Г-О-О-О-Л !!! ";    //Текст "ГОЛ!" =12
unsigned char t2[]={ 0xC0,0xC4,0xCC,0xCE };  //Адрес ЖКИ =13
unsigned char t3[]={ 0x00,0x04,0x00,0x03 };     //Фигуры =14
unsigned char t4[]={ 0xC0,0xC1,0xC1,0xC2,0xC2,0xC3,   // =15
 0xC4,0xC5,0xC5,0xC6,0xC6,0xC7,0xC7,0xC8,0xC8,0xC9,   // =16
 0xC9,0xCA,0xCA,0xCB,0xCB,0xCC,0xCC,0xCD,0xCE,0xCE }; // =17
unsigned char t5[]={ 0x20,0x01,0x20,0x02,0x20,0x01,   // =18
 0x20,0x04,0x20,0x05,0x20,0x06,0x20,0x07,0x20,0x07,   // =19
 0x20,0x06,0x20,0x05,0x20,0x04,0x02,0x05,0x20,0xFF }; // =20
unsigned char foot[]={ //Знакогенератор футбольных фигур =21
 0x0E,0x0E,0x04,0x1F,0x04,0x04,0x0A,0x11, //0=руки гориз.=22
 0x0E,0x0E,0x05,0x0E,0x14,0x04,0x0A,0x11, //1=бегущий    =23
 0x00,0x0E,0x0E,0x05,0x0E,0x14,0x04,0x1F, //2=падение    =24
 0x1F,0x11,0x11,0x11,0x11,0x11,0x11,0x11, //3=ворота     =25
 0x00,0x00,0x00,0x0C,0x1E,0x1E,0x0C,0x00, //4=мяч внизу  =26
 0x00,0x00,0x0C,0x1E,0x1E,0x0C,0x00,0x00, //5=мяч выше   =27
 0x00,0x0C,0x1E,0x1E,0x0C,0x00,0x00,0x00, //6=мяч выше_2 =28
 0x0C,0x1E,0x1E,0x0C,0x00,0x00,0x00,0x00 }; //7=мяч верх =29
//================ОСНОВНАЯ ПРОГРАММА=================== =30
int main(void)                 //Начало основной программы =31
{ unsigned char a, b;                          //Счетчики =32
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =33
  PORTC = 0xF8; DDRC = 0x07;    //PC0...3 выходы с лог.0 =34
  lcd_init();          //Инициализация ЖКИ (4 бит, 16x2) =35
  lcd_com(0x0C);                 //Выключение курсора ЖКИ =36
  for (lcd_com(0x40), a=0; a<64; a++)   //Загрузка фигур =37
  { lcd_dat(foot[a]);   //Текущий символ знакогенератора =38
  }          //Окончание загрузки знакогенератора фигур =39
  while (1)                            //Бесконечный цикл =40
  { for (lcd_com(0x80), a=0; a<32; a++)  //Нач. заставка =41
    { if (a==16) lcd_com(0xC0); //Переход на нижн.строку =42
      lcd_dat(t0[a]);           //Вывод текущего символа =43
    }      //Окончание вывода начальной надписи ПЕНАЛЬТИ =44
    for (a=0; a<4; a++)   //Начальное расположение фигур =45
    { lcd_com(t2[a]); lcd_dat(t3[a]);    //Адрес, фигура =46
    }           //Окончание рисования начальной картинки =47
    for (a=0; a<(sizeof(t4)-1); a=a+2)  //Мультипликация =48
    { for (b=0; b<12; b++) pause(TEMP);       //Скорость =49
      lcd_com(t4[a]); lcd_dat(t5[a]);   //Стирание фигур =50
      lcd_com(t4[a+1]); lcd_dat(t5[a+1]);    //Рисование =51
    }                  //Окончание показа "мультфильма" =52
    for (lcd_com(0x80), a=0; a<16; a++) //Верхняя строка =53
    { lcd_dat(t1[a]);           //Вывод текущего символа =54
    }                   //Окончание вывода надписи ГОЛ! =55
    for (b=0; b<150; b++) pause(TEMP);   //Длинная пауза =56
  }               //Переход к новому показу "мультфильма" =57
}               //WinAVR-20050214, длина кода 662 байта =58
```

0	0	0	0	1	1	0	0	0x0C
0	0	0	1	1	1	1	0	0x1E
0	0	0	1	1	1	1	0	0x1E
0	0	0	0	1	1	0	0	0x0C
0	0	0	0	0	0	0	0	0x00
0	0	0	0	0	0	0	0	0x00
0	0	0	0	0	0	0	0	0x00
0	0	0	0	0	0	0	0	0x00

Экран ЖКИ

рис.6

В качестве темы подойдет футбол, а конкретно – выполнение 11-метрового удара. На **рис.5** показан макет сценария, расписанный по клеткам нижней строки ЖКИ.

В фазе 1 футболист (Ф) разгоняется для удара и бежит к мячу (М). В фазе 2 мяч летит к вратарю (В). В фазе 3 вратарь пытается поймать мяч, но пропускает его в ворота (П).

Электрическую схему "мультфильм-проектора" специально собирать не надо. Подойдет схема от "кибер-отгадчика" или "электронной визитки". ЖКИ можно взять с подсветкой или без, но обязательно двухстрочный.

Управляющая программа приведена в **листинге 5**.

Строки 13, 14 содержат согласованные массивы чисел, которые задают начальное положение фигур согласно рис.5. Числа в массиве t2[ ] являются адресами знакомест нижней строки ЖКИ. Числа в массиве t3[ ] определяют номер фигуры в массиве foot[ ], соответственно в строках 22 (футболист стоит), 26 (мяч внизу), 22 (вратарь стоит), 25 (ворота).

Строки 15–20 определяют сценарий "мультфильма". Числа в массивах t4[ ], t5[ ] попарно согласованы друг с другом. Первые из них являются адресами знакомест ЖКИ, вторые – номерами выводимых символов. Например, элемент t4[0] содержит код 0xC0, это первое слева знакоместо в нижней строке ЖКИ. Элемент t5[0] содержит код 0x20, т.е. знак пробела. Получается, что первое знакоместо будет очищено. Второй элемент массива t4[1] 0xC1 указывает на следующую справа клетку ЖКИ, в который будет занесен символ из адреса 0x01. Согласно строке 23 это – фигурка бегущего человечка. Общий результат: фигурка футболиста, первоначально размещенная на поле 0xC0, будет стерта, а на следующем поле 0xC1 появится фигурка бегущего футболиста и т.д.

Строки 21–29 определяют начертания фигур, расположенные по адресам 0x00–0x07. Для примера на **рис.6** показан процесс формирования изображения мяча в верхней точке полета, строка 29, код знакогенератора 0x07. "Единицы" обозначают видимый пиксел изображения, "нули" – невидимый.

Строки 30–58 особенностей не имеют. При выводе кадров "мультфильма" используются приемы, характерные для "бегущей строки".

Практическое задание. Отладить "кибер-отгадчик". Ввести в его программу бегущую строку, для чего дополнить листинг 1 строками из листинга 2. Собрать "электронную визитку" и промоделировать ее работу в VMLab. Просмотреть футбольный "мультфильм" в режиме реального времени, довольно забавное зрелище.

Микроконтроллеры AVR. Ступень 8

С.М. Рюмик, г. Чернигов

Предварительное знание того, что хочешь сделать, дает смелость и легкость
Дени Дидро

Можно ли наделить микроконтроллер (МК) функциями вольтамперметра или многоканального запоминающего осциллографа? Немного фантазии, немного смекалки, немного ограничений – и ответ положительный. По крайней мере, с исследованием переходных процессов такие устройства справляются без проблем. В подтверждение тому будут рассмотрены практические примеры.

Входные сигналы, подаваемые на линии портов МК, могут быть цифровыми и аналоговыми. Цифровой сигнал – это лог."0" или лог."1", аналоговый – любое значение напряжения в допустимом для МК диапазоне. В частности, для ATmega8 это от –0,5 В до VCC+0,5 В, где VCC – напряжение питания.

Однако напрямую, без всякой подготовки, аналоговый сигнал в МК обрабатываться не будет. Сначала надо активизировать внутренний аналого-цифровой преобразователь (АЦП), или ADC (**A**nalog-to-**D**igital **C**onverter).

В ATmega8 имеется 6-канальный АЦП 10-битной точности. Иногда пишут о 8-канальном АЦП, но это справедливо лишь для версий ATmega8 в 32-выводных квадратных корпусах TQFP и MLF. Кроме того, бытует мнение, что 2 из 6 каналов АЦП имеют точность не 10, а 8 бит. Действительно, такие параметры были у микросхем выпуска 2001–2003 гг. Позже, благодаря изменениям кристалла, разрешение во всех каналах улучшилось до 10 бит. Об этом сделана запись в DATASHEET в разделе "Revision History, Changes from Rev. 2486M".

Отличить "старую" микросхему от "новой" можно по внешнему виду (**рис.1**). Цифры "0421" на ее корпусе обозначают дату выпуска – 21 неделя, начиная от 01.01.2004 г. Буква "G" – это литера изменения кристалла согласно разделу "Erratas" DATASHEET.

рис.1

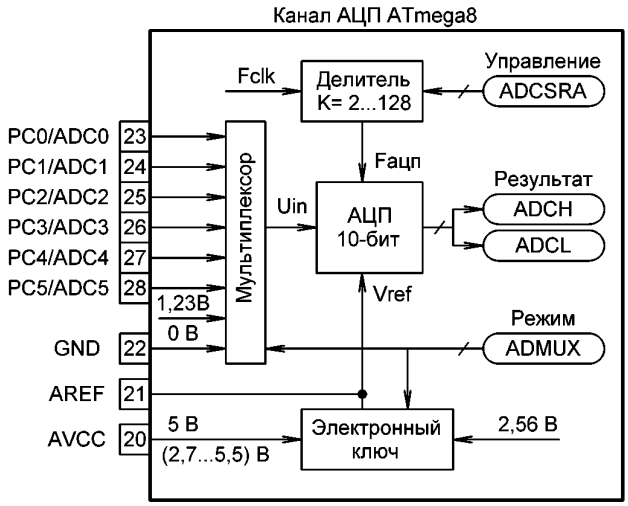

рис.2

Структурная схема канала АЦП

Насколько глубоко надо вникать в архитектуру МК при его изучении? Для семейства MCS-51 достаточно было общих рассуждений и рисунков входных-выходных линий портов.

Семейство AVR функционально сложнее, поэтому объяснить, "размахивая руками", принцип работы канала АЦП затруднительно. Другая крайность, которая нередко встречается в книгах, – это слепое копирование из документации структурной схемы и формальный перевод на русский язык встречающихся в ней терминов. Надежда на то, что кому надо, тот сам разберется.

В качестве компромиссного варианта на **рис.2** показана упрощенная схема канала АЦП ATmega8, в которой оставлены только важные для понимания элементы и логически значимые связи.

Входные аналоговые сигналы с линий PC0–PC5 поступают на мультиплексор, который в зависимости от состояния регистра ADMUX, подключает один из сигналов прямо на вход АЦП Vin. Вторые названия линий ADC0–ADC5 указывают порядковый номер канала 0–5. Кроме входного сигнала модуль АЦП требует для нормальной работы эталон напряжения Vref и тактовый сигнал Fацп от предделителя, управляемого регистром ADCSRA. Результат каждого преобразования помещается в регистры ADCH (старшие 2 бита) и ADCL (младшие 8 битов). Итого, в двух регистрах образуется 10-битное число в диапазоне 0–1023 или в шестнадцатеричной форме 0x000–0x3FF.

Судя по рис.2, "6-канальность" АЦП не означает обработку 6 аналоговых сигналов одновременно. Только по очереди, последовательно во времени. Кроме того, мультиплексор может подавать на АЦП два вспомогательных проверочных уровня: 1,23 В от внутреннего источника опорного напряжения (ИОН) и 0 В (общий провод). Это, по идее, должно пригодиться при тестировании и калибровке.

Вывод AREF имеет непосредственную гальваническую связь с модулем АЦП. Налицо прямой путь для наводок и помех, воздействие которых сразу же отразится на выходных показаниях. Именно поэтому между выводами AREF и GND рекомендуется ставить шунтирующий керамический конденсатор емкостью 0,1 мкФ.

Источниками опорного уровня Vref могут служить: во-первых, напряжение питания AVCC, во-вторых, внутренний ИОН 2,56 В, в-третьих, внешнее напряжение, подаваемое на линию AREF. Первые два источника коммутируются электронным ключом, при этом их точные значения можно измерить вольтметром между выводами 21 и 22 ATmega8.

От величины опорного напряжения зависит диапазон измерения. Например, если Vref=2,56 В, то нормально будут обрабатываться аналоговые сигналы в пределах 0...2,56 В с градацией 2,5 мВ. Отрицательные напряжения –0,5...0 В преобразуются в 0 В, а 2,56...5,5 В – в 2,56 В. Если Vref=5 В (AVCC), то диапазон расширяется до 0...5 В с одновременным увеличением градаций до 5 мВ. При измерении напряжений свыше 5 В следует использовать резистивные или другие делители.

Принцип работы АЦП

Процесс преобразования аналогового сигнала в цифровые выборки поясняет **рис.3**. Вертикальная шкала разбивается на 1024 полки, что соответствует разрешению 10 бит (число "два в десятой степени"). В момент времени T1 запоминается текущая амплитуда сигнала. В промежутке T1–T2 методом последовательных приближений вычисляется ближайший по уровню цифровой код. Результат помещается в регистры ADCH, ADCL, после чего в момент времени T2 АЦП готов к новому измерению.

Входное напряжение Vin рассчитывается по формуле:

Vin[В]=(ADCH·256+ADCL)·Vref[В]/1024.

Пример. Напряжение Vref подается от внешнего источника 4 В. Из регистров ADCH, ADCL считываются коды 0x02, 0x11. Следовательно, Vin=(2·256+17)·4/1024=2,066 В.

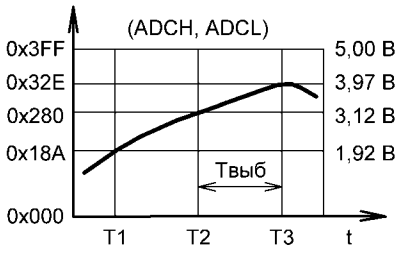

рис.3

АЦП в ATmega8 имеет еще один режим преобразования, условно называемый 8/10-бит, когда измерение производится в 10 разрядов, но индицируются только 8 старших из них. Результат помещается в регистр ADCH, а напряжение Vin рассчитывается по формуле:

Vin[В]=ADCH·Vref[В]/256.

Пример. Напряжение Vref подается от внешнего источника 2,5 В. Из регистра ADCH считывается код 0xAF. Следовательно, Vin=175·2,5/256=1,71 В.

Расстояние между выборками Твыб является величиной, обратно пропорциональной частоте дискретизации Fд. Значение Твыб рассчитывает сам пользователь, исходя из определенных критериев. Главный из них – это максимальная частота Fmax в спектре исследуемого сигнала. К примеру, телефонный канал сужает спектр человеческого голоса до 3,4 кГц, следовательно, по теореме Котельникова Fд[кГц]>2·Fmax[кГц]=2·3,4=6,8 кГц, откуда Твыб[мкс]<1000/Fд[кГц]=1000/6,8=147 мкс.

На практике чаще всего приходится иметь дело с различными датчиками, обрабатываемыми АЦП. Здесь основным критерием служит время, в течение которого состояние датчика может стать аварийным и вернуться обратно. Если АЦП не замечает подобных "просадок", то время выборки надо уменьшить.

В ATmega8 можно программно задать минимальные значения Твыб=15...260 мкс, что соответствует частотам дискретизации Fд=4...66 кГц. Разумеется, максимальные значения Твыб могут быть сколь угодно большими. Для уменьшения погрешностей АЦП рекомендуется более узкий диапазон Твыб=65...260 мкс (Fд=4...15 кГц).

Точность измерения

Разрядность АЦП 10 бит – это много или мало? Для научных экспериментов – мало. Здесь лучше применить отдельную микросхему АЦП, такую, как ADS8380 (фирма Texas Instruments), с разрядностью 18 бит при частоте дискретизации 600 кГц и цене 17USD.

Для любительских конструкций 10-разрядная точность (0,1%) может оказаться даже излишней. В частности, нет смысла измерять напряжение 5 В на выходе интегрального стабилизатора с точностью 5 мВ (10 бит), если напряжение пульсаций равно 10...20 мВ.

С другой стороны, суммарная системная погрешность АЦП в ATmega8 составляет ±1,5...2 младшие единицы, т.е. гарантированно точными во всем диапазоне температур и напряжений являются 8 разрядов из 10 (0,4%). Если добавить к этому разброс напряжений внутренних ИОН 2,56 В (2,3...2,7 В) и 1,23 В (1,15...1,4 В), а также нестабильность питания AVCC 0,5...1%, то реальная точность абсолютных измерений понизится до 5–6 разрядов (1...3%).

Важный нюанс. Не следует путать абсолютные и относительные измерения. Для сравнения, при подаче на вход АЦП заранее неизвестного напряжения, его абсолютная величина будет измерена достаточно грубо. Однако если необходимо уловить малейшее изменение относительного уровня датчика, то это будет сделано с 10-разрядной точностью.

Улучшить метрологические характеристики АЦП помогают два способа:

увеличение времени замеров с усреднением нескольких результатов подряд;

калибровка по внешнему цифровому вольтметру с подбором коэффициентов для каждого конкретного экземпляра ATmega8.

Программирование режимов АЦП

В процессе работы преобразователь может функционировать в режимах однократного или непрерывного измерений. Запуск однократного режима можно сравнить с программной кнопкой: нажал и через 65...260 мкс получил результат. Измерения в непрерывном режиме не требуют перезапуска, результат можно прочитать в любой момент времени, однако обновление показаний также будет происходить через 65...260 мкс.

Установка режимов производится записью байтов в два регистра, обозначенных на рис.2 как ADMUX (**A**nalog-to-**D**igital **MU**ltiple**X**er) и ADCSRA (**A**nalog-to-**D**igital **C**ontrol and **S**tatus **R**egister **A**). Чтобы не запутаться в назначениях битов, информация о которых разбросана по DATASHEET, в **табл.1** компактно собраны команды на языке Си, которые отвечают за тот или иной режим АЦП. Пользователю остается лишь последовательно выбрать из 7 пунктов по одной требуемой строчке и включить их в листинг программы.

Пояснения к пункту 1. Режим 8/10-бит проще в программировании и приводит к меньшему объему кода. В отличие от "чистого" 8-битного режима, здесь выше точность, так как отсутствует неопределенность младшего разряда. Сменить режим 8/10-бит на 10-бит и обратно можно программно в любой момент времени.

Пояснение к пункту 7. Следует отличать тактовую частоту МК Fclk от тактовой частоты АЦП Fацп и частоты дискретизации сигнала Fд. Первая из них определяется кварцевым резонатором или внутренним генератором МК, вторая – получается делением Fclk на число K в диапазоне 2...128, третья – ниже Fацп в 13 раз.

В **табл.2** приведены Си-команды, отвечающие за чтение данных из АЦП. Переменная "a" в режиме 8/10-бит должна иметь тип "unsigned char", в режиме 10-бит – "unsigned int".

Минимум погрешности гарантируется при Fацп=50...200 кГц, что соответствует Твыб=65...200 мкс. Именно в этот диапазон надо уложиться в режиме 10-бит. Если устраивает меньшая точность, то Fацп можно увеличить до 1 МГц. Зачем? Чем выше Fацп, тем меньше время выборки.

Пример расчета. МК работает от внутреннего генератора Fclk=8 МГц. Выборки АЦП требуется производить с частотой не менее Fд=10 кГц (Твыб=100 мс). Коэффициент деления рассчитывается по формуле:

K=Fclk[кГц]/(13·Fд[кГц])=8000/(13·10)=61.

Далее надо округлить K до ближайшего большего из ряда 2, 4,

Таблица 1

№	Операторы языка Си	Комментарии
1	ADMUX |= 0x20;	Разрядность АЦП 8/10-бит
	ADMUX &= 0xDF;	Разрядность АЦП 10-бит
2	ADMUX |= 0xC0;	Внутренний ИОН, Vref=2,56 В
	ADMUX |= 0x40; ADMUX &= 0x7F;	Внутренний ИОН, Vref=(AVCC) В
	ADMUX &= 0x3F;	Внешний ИОН, Vref=(AREF) В
3	ADMUX &= 0xF0;	Канал-0, линия PC0
	ADMUX &= 0xF1; ADMUX |= 0x01;	Канал-1, линия PC1
	ADMUX &= 0xF2; ADMUX |= 0x02;	Канал-2, линия PC2
	ADMUX &= 0xF3; ADMUX |= 0x03;	Канал-3, линия PC3
	ADMUX &= 0xF4; ADMUX |= 0x04;	Канал-4, линия PC4
	ADMUX &= 0xF5; ADMUX |= 0x05;	Канал-5, линия PC5
	ADMUX &= 0xFE; ADMUX |= 0x0E;	Тест-напряжение Vin=1,23 В
	ADMUX |= 0x0F;	Тест-напряжение Vin=0 В
4	ADCSRA |= 0x80;	Включить АЦП
	ADCSRA &= 0x7F;	Выключить АЦП
5	ADCSRA |= 0x40;	Запуск нового замера АЦП
6	ADCSRA |= 0x20;	Постоянное измерение
	ADCSRA &= 0xDF;	Однократный запуск АЦП
7	ADCSRA &= 0xF8;	Частота Fацп=Fclk /2
	ADCSRA &= 0xFA; ADCSRA |= 0x02;	Частота Fацп=Fclk /4
	ADCSRA &= 0xFB; ADCSRA |= 0x03;	Частота Fацп=Fclk /8
	ADCSRA &= 0xFC; ADCSRA |= 0x04;	Частота Fацп=Fclk /16
	ADCSRA &= 0xFD; ADCSRA |= 0x05;	Частота Fацп=Fclk /32
	ADCSRA &= 0xFE; ADCSRA |= 0x06;	Частота Fацп=Fclk /64
	ADCSRA |= 0x07;	Частота Fацп=Fclk /128

Таблица 2

Операторы языка Си	Комментарии

ADCSRA |= 0x40;
while (ADCSRA & 0x40);
a = ADCH;
	Однократный запуск,
	разрядность 8/10-бит,
	Vin = a * Vref / 256

ADCSRA |= 0x40;
while (ADCSRA & 0x40);
a = ADCL;
a += ((int) ADCH << 8);
	Однократный запуск,
	разрядность 10-бит,
	Vin = a * Vref / 1024

a = ADCH;	Постоянное измерение, разрядность 8/10-бит, Vin = a * Vref / 256
a = ADCL; a += ((int) ADCH << 8);	Постоянное измерение, разрядность 10-бит, Vin = a * Vref / 1024

while (!(ADCSRA & 0x10));
ADCSRA |= 0x10;
a = ADCH;
	Постоянное измерение,
	синхронизация запуска,
	разрядность 8/10-бит,
	Vin = a * Vref / 256

while (!(ADCSRA & 0x10));
ADCSRA |= 0x10;
a = ADCL;
a += ((int) ADCH << 8);
	Постоянное измерение,
	синхронизация запуска,
	разрядность 10-бит,
	Vin = a * Vref / 1024

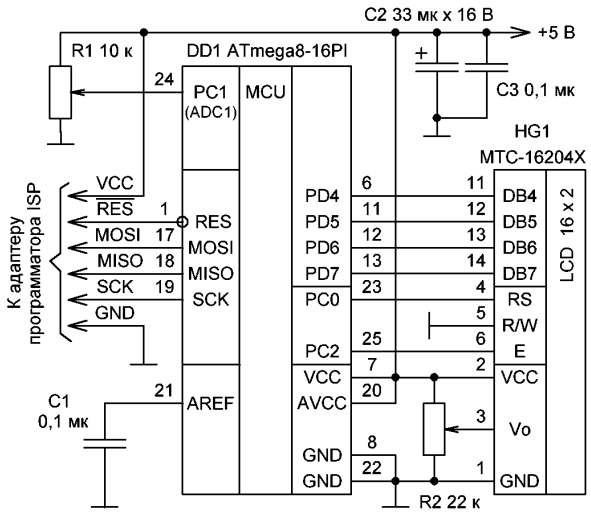

рис.4

8, 16, 32, 64, 128, выбрать по табл.1 строку Fацп=Fclk/64 и включить в программу команды "ADCSRA &= 0xFE; ADCSRA |= 0x06;".

Цифровой вольтметр

На **рис.4** показана электрическая схема простейшего вольтметра, собранного на МК DD1 и ЖКИ HG1. Измеряемое напряжение 0...5 В подается в линию PC1 от переменного резистора R1. Его сопротивление не должно быть слишком большим, поскольку минимальная погрешность АЦП гарантируется при сопротивлении источника сигнала не более 10 кОм.

Резистором R2 регулируется контрастность изображения ЖКИ. Конденсаторы C1–C3 фильтрующие. Индикатор HG1 может быть другого типа, что непринципиально.

Пояснения к управляющей программе (**листинг 1**).

Листинг 1

```
//Вольтметр на ЖКИ (АЦП),=AVR. Ступень 8=, РА, №9, 2005 г =1
//Make: avr81,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="0" (Генератор 1 МГц)  =3
#include <avr/io.h>                 //Библиотека ввода-вывода =4
#define VREF 5000          //Напряжение Vref в милливольтах =5
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =6
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =7
extern void lcd_init(void);            //Инициализация ЖКИ =8
unsigned char t[]=" ==VOLTMETER==           Volt   "; //  =9
//================ОСНОВНАЯ ПРОГРАММА=================== =10
int main(void)                    //Начало основной программы =11
{ unsigned long volt;     //Измеряемое напряжение Vin АЦП =12
  unsigned int a;                 //Вспомогательный счетчик =13
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =14
  PORTC = 0xF8; DDRC = 0x05;   //PC0, PC2 выходы с лог.0 =15
  lcd_init();            //Инициализация ЖКИ (4 бит, 16x2) =16
//Регистр ADMUX: АЦП 10 бит, Vref=AVCC, канал-1 (PC1)     =17
  ADMUX &= 0xDF & 0x7F & 0xF1; ADMUX |= 0x40 | 0x01;   //=18
//Регистр ADCSRA: вкл. АЦП, постоян. измер., Fацп=125кГц =19
  ADCSRA &= 0xFB; ADCSRA |= 0x80 | 0x40 | 0x20 | 0x03; //=20
  for (lcd_com(0x80), a=0; a<32; a++)    //Текст заставки =21
  { if (a==16) lcd_com(0xC0); //Переход на нижнюю строку =22
    lcd_dat(t[a]);                 //Вывод текущего символа =23
  }       //Окончание вывода начальной надписи VOLTMETER =24
  while (1)                    //Бесконечный цикл измерений =25
  { volt = ADCL;      //Чтение младших 8 битов результата =26
    volt += ((int)ADCH << 8);     //Плюс два старших бита =27
    volt=volt*VREF/1024; //Вычисление Vin в милливольтах =28
    lcd_com(0xC3);    //Установка курсора в нижней строке =29
    lcd_dat(volt/1000 + 0x30);  //Индикация единиц вольт =30
    lcd_dat(',');                    //Индикация запятой =31
    lcd_dat((volt/100)%10 + 0x30);     //Сотни милливольт =32
    lcd_dat((volt/10)%10 + 0x30);    //Десятки милливольт =33
    lcd_dat(volt%10 + 0x30);          //Единицы милливольт =34
    for (a=65000; a > 0; a--);      //Пауза для индикации =35
  }                        //Переход к новому измерению АЦП =36
}                //WinAVR-20050214, длина кода 680 байтов =37
```

Листинг 2

```
#define VREF 2560          //Напряжение Vref в милливольтах =5
//Регистр ADMUX: АЦП 10 бит, Vref=2,56 В, канал-1 (PC1)  =17
  ADMUX &= 0xDF & 0xF1; ADMUX |= 0xC0 | 0x01;          // =18
```

Листинг 3

```
V1 PC1 VSS SLIDER_1(0 5) ; Переменный резистор между VCC, PC1, GND
; ЖКИ 16x2        RS   R/W   E   Интерфейс 4-бит    Не подключено
;                  ---  ---  ---  ----------------  ----------------
X1 LCD(16 2 250K) PC0  VSS  PC2   PD7 PD6 PD5 PD4  nc3 nc2 nc1 nc0
.PLOT v(PC0) v(PC1) v(PC2) ; Графики на экране осциллографа
```

Строки 2, 6–8. Поскольку в схеме применяется ЖКИ, то для его обслуживания необходим соответствующий "драйвер", т.е. файл "lcd.c" с встроенными функциями ввода, вывода, инициализации. Его листинг, а также методика программирования подробно освещены в "Ступени 7". В конце строки 2 указан код, который надо не забыть включить в make-файл.

Строка 5. В качестве опорного напряжения Vref используется питание AVCC 5 В. Реально оно может иметь разброс ±1...4% от номинала. Следовательно, после первого запуска прибора в работу надо провести его калибровку, которая заключается в измерении внешним вольтметром напряжения Vref на обкладках конденсатора C1. После этого полученное число в милливольтах надо подставить в константу VREF и заново откомпилировать программу.

Аналогичную калибровку надо проводить, если Vref=2,56 В (**листинг 2**, остальное см. в листинге 1). Практика показывает, что вместо обещанных 2,56 В нередко индицируется 2,59...2,65 В. Браковать такие МК нет причины, поскольку DATASHEET допускает разброс 2,4...2,7 В.

Строка 15 конфигурирует порт C. В схеме вольтметра линии PC0, PC2 выполняют функцию обычных цифровых выходов. Это абсолютно не мешает использованию их "среднего брата" PC1 в качестве входа АЦП. Единственное, надо отключить внутренний резистор от PC1, иначе будут искажаться показания вольтметра. А вот к незадействованным линиям PC3–PC5, наоборот, необходимо программно подключить резисторы, чтобы устранить путь проникновения нежелательных помех. В итоге получается, что настройку регистров PORTC, DDRC необходимо производить для каждой схемы отдельно, с учетом свободных и занятых линий порта C.

Строки 18, 20. Числа в регистрах ADMUX, ADCSRA соответствуют табл.1, только записаны они более компактно, в порядке их извлечения из пунктов 1–7.

Строка 19. Формула для расчета: Fацп[кГц]=Fclk[кГц]/K=1000/8=125 кГц.

Строки 26–28 взяты из табл.2 для случая постоянных измерений без синхронизации запуска. Почему "без"? Потому что в данном приборе не требуется проводить быстрые замеры, следующие друг за другом. Предполагается, что вольтметр индицирует постоянные и медленно меняющиеся напряжения.

Строка 35 определяет паузу между измерениями, чтобы пользователь мог, не утомляясь, разглядеть показания на ЖКИ.

Моделирование в среде Vmlab

Заготовка файла проекта "avr81.prj" создается обычным способом, рассмотренным в "Ступени 6". В конце добавляется мнемоническое описание электрической схемы согласно **листингу 3**. Из новинок – строка, содержащая эквивалент трехвыводного переменного резистора V1, подключенного к точкам +5 В, "Общий" и к линии PC1 МК.

После запуска проекта на выполнение можно "мышью" изменить положение движка резистора R1 в меню "View – Control Panel" (ползунок S1), при этом на экране ЖКИ должны меняться числа напряжения, а на виртуальном осциллографе – "плавать" уровень. Вследствие неторопливости симулятора VMLab, отклики будут появляться не мгновенно, а через несколько секунд. На **рис.5** для примера показан случай, когда движок переменного резистора установлен в верхнее по схеме положение и на АЦП поступает полное напряжение питания AVCC. Текущее состояние регистров можно посмотреть в меню "View – Peripherals – A/D converter".

Интересная деталь. Напряжение AVCC=Vref=5 В, а ЖКИ показывает 4,995 В. Ошибки здесь нет, это следствие дискретности преобразования АЦП, когда измеряемое напряжение приводит-

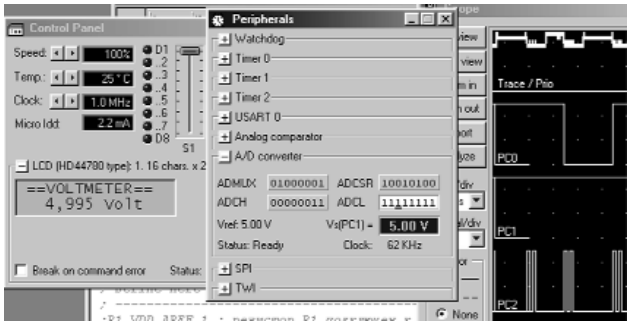

рис.5

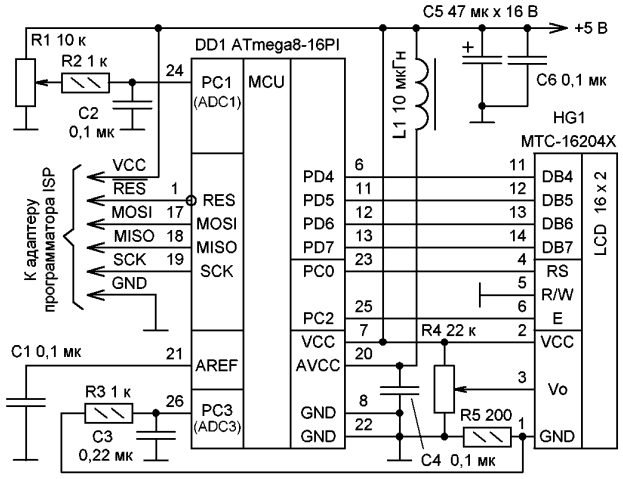

рис.6

ся к ближайшему меньшему с 10-разрядной точностью. Получается, что любое напряжение в диапазоне 4,995...5,000 В будет точно так индицироваться.

Цифровой вольтамперметр

АЦП в МК может измерять только напряжения. Все остальные физические величины должны быть пропорционально преобразованы. Для токов существует обходной маневр, который заключается в измерении разности напряжений на известном сопротивлении, а дальше расчет по закону Ома: I=(U1-U2)/R.

На **рис.6** показана электрическая схема двухканального прибора, позволяющего мерять не только напряжение на резисторе R1, но и ток потребления ЖКИ с отображением результата на самом ЖКИ. "Изюминкой" схемы является измерительный резистор R5. Падение напряжения на нем прямо пропорционально протекающему току. В общем случае резистор можно ставить в разрыв цепи GND или VCC. Его сопротивление выбирается таким, чтобы падение напряжения на нем составляло не более 1% от напряжения питания ЖКИ. Точность R5 – чем выше, тем лучше, например, 0,5...1%. Можно применить и обычный резистор с допуском 5%, но тогда потребуется предварительно измерить его сопротивление точным омметром и скорректировать константу RIZM в программе.

RC-цепочки R2C2, R3C3 выполняют функцию фильтров, не позволяющих прибору реагировать на импульсные помехи и шумы. Частоту среза фильтров подбирают экспериментально, в зависимости от условий эксплуатации. Чаще всего 1...5 кГц, поскольку внутри АЦП уже имеется фильтр с верхней частотой 38 кГц.

Резисторы R2, R3 должны иметь сопротивление 1...10 кОм. Их вспомогательная роль (а в некоторых случаях и главная) заключается в ограничении входных токов при подаче на АЦП отрицательных или больших положительных напряжений. Максимум тока не более 20 мА, чтобы не повредить защитные диоды, находящиеся внутри МК.

Уменьшению помех способствует дроссельная развязка L1C4, номиналы которой рекомендованы фирмой Atmel. Как ни парадоксально, но чаще всего приходится защищаться от импульсных помех, генерируемых по питанию самим МК!

Аппаратная фильтрация помех повышает точность замеров в режиме 10-бит, а также снижает уровень наводок от промышленного оборудования. В лабораторных (читай, домашних) условиях наличие фильтров чаще всего не обязательно.

Кроме аппаратных, существуют и программные способы повышения точности, один из которых будет рассмотрен в управляющей программе.

Пояснения к **листингу 4**.

Строки 18, 36, 42. Вольтамперметр использует два канала АЦП: канал-1 для измерения напряжения на движке резистора R1 и канал-3 для измерения тока, протекающего через резистор R5. Переключаются каналы поочередно во времени, записью в регистр ADMUX значений из табл.1. Точность в канале-3 выбрана 8 бит, поскольку ток ЖКИ не очень стабилен.

Строки 22-27, 37-41. Усреднение по 10 выборкам повышает точность измерений прибора. Для сравнения, показания вольтметра с управляющей программой из листинга 1 меняются с дискретом 5 мВ, а в данном устройстве – через 1 мВ.

Строка 26 содержит число "0x03", отсутствующее в командах

Листинг 4

```
//Вольтамперметр на ЖКИ,=AVR. Ступень 8=, РА, №9, 2005 г  =1
//Make: avr82,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="0" (Генератор 1 МГц)  =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
#define VREF 5000      //Напряжение Vref в милливольтах =5
#define RIZM 200  //Сопротивление измер. резистора в Омах =6
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =7
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =8
extern void lcd_init(void);            //Инициализация ЖКИ =9
//================ОСНОВНАЯ ПРОГРАММА=================== =10
int main(void)                  //Начало основной программы =11
{ unsigned long volt, amp; //Измеренное напряжение и ток =12
  unsigned int a;              //Вспомогательный счетчик =13
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =14
  PORTC = 0xF0; DDRC = 0x05;   //PC0, PC2 выходы с лог.0 =15
  lcd_init();          //Инициализация ЖКИ (4 бит, 16x2) =16
//Регистр ADMUX: АЦП 10 бит, Vref=AVCC, канал-1 (PC1)    =17
  ADMUX &= 0xDF & 0x7F & 0xF1; ADMUX |= 0x40 | 0x01;  // =18
//Регистр ADCSRA: вкл. АЦП, одиночный запуск, Fацп=62кГц =19
  ADCSRA &= 0xDF & 0xFC; ADCSRA |= 0x80 | 0x04;       // =20
  while (1)                              //Бесконечный цикл =21
  { for (volt=0, a=10; a>0; a--) //10 замеров напряжения =22
    { ADCSRA |= 0x40;       //Запуск нового измерения АЦП =23
      while (ADCSRA & 0x40); //Проверка окончания замера =24
      volt += ADCL;  //Чтение младших 8 битов результата =25
      volt += ((int)(ADCH & 0x03) << 8); //Плюс два бита =26
    }                        //Окончание 10 замеров напряжения =27
    volt=volt*VREF/10240;   //Среднее напряжение Vin, мВ =28
    lcd_com(0x83);  //Установка курсора в верхней строке =29
    lcd_dat(volt/1000 + 0x30);  //Индикация единиц вольт =30
    lcd_dat(',');                    //Индикация запятой =31
    lcd_dat((volt/100)%10 + 0x30);    //Сотни милливольт =32
    lcd_dat((volt/10)%10 + 0x30);   //Десятки милливольт =33
    lcd_dat(volt%10 + 0x30);         //Единицы милливольт =34
    lcd_dat(0x20); lcd_dat('V');   //Буква < V> (вольты) =35
    ADMUX &=0xF3; ADMUX |= 0x20 | 0x03; //8-10 бит,кан-3 =36
    for (amp=0, a=10; a>0; a--)         //10 замеров тока =37
    { ADCSRA |= 0x40;       //Запуск нового измерения АЦП =38
      while (ADCSRA & 0x40); //Проверка окончания замера =39
      amp += ADCH;   //Чтение старших 8 битов результата =40
    }                        //Окончание 10 замеров тока =41
    ADMUX &=0xDF & 0xF1; ADMUX |= 0x01; //10 бит,канал-1 =42
    amp=(100*(amp+5)*VREF/2560)/RIZM;  //Средний ток, мА =43
    lcd_com(0xC3);  //Установка курсора  в нижней строке =44
    lcd_dat(amp/100 +0x30);   //Единицы миллиампера (мА) =45
    lcd_dat(',');                    //Индикация запятой =46
    lcd_dat((amp/10)%10 +0x30); //Десятые доли миллиамп. =47
    lcd_dat(amp%10 + 0x30);     //Сотые доли миллиампера =48
    lcd_dat(0x20); lcd_dat('m'); lcd_dat('A');   //< mA> =49
    for (a=60000; a > 0; a--);     //Пауза для индикации =50
  }                          //Переход к новому измерению АЦП =51
}              //WinAVR-20050214, длина кода 860 байтов =52
```

табл.2. Это своеобразная плата за смену режимов. Дело в том, что в регистре ADCH после измерения в режиме 8/10-бит остается байт данных, который может исказить показания АЦП в режиме 10-бит. Следовательно, число "0x03" принудительно очищает 6 старших битов регистра ADCH.

Строка 28. Поскольку переменная "volt" содержит сумму из 10 выборок, то делитель в формуле пропорционально увеличен с 1024 до 10240.

Строка 43. Расчет тока по закону Ома. В подобных формулах надо внимательно следить за порядком расположения чисел, чтобы не происходило промежуточное деление меньшей величины на большую.

Число "100" округляет ток до сотых долей миллиампера. Число "5" введено для повышения точности. Это усредненная прибавка к ADCH, которая накопилась за 10 замеров (половина последнего разряда). Здесь учтен факт, что в режиме 8/10-бит округление всегда происходит в меньшую сторону с отбрасыванием двух младших разрядов. Порядок выполнения операций слева направо.

Пример. Из регистра ADCH считывается код 0x80, что при Vref=5 В означает напряжение Vin=2,5 В. Однако этот же код будет считываться при Vin=(2,50...2,52) В. Чтобы уменьшить погрешность, надо к Vin прибавить половину разности диапазона и предположить, что Vin=2,51 В.

Цифровой ваттметр

Способ калибровки, рассмотренный в листинге 1, учитывает, что напряжение питания 5 В стабильно во времени. А как быть при батарейном или аккумуляторном источнике, когда по мере разряда, его напряжение постоянно уменьшается? Здесь можно предложить вариант автокалибровки, которая будет производиться всякий раз при подаче питания.

Электрическая схема для экспериментов остается прежней (рис.6), но индицироваться для разнообразия будет не ток, а мощность потребления ЖКИ.

Пояснения к **листингу 5**.

Строка 6. Константа ION содержит число, которое на 2 мВ меньше номинального напряжения ИОН 1,23 В. Измерить это на-

Листинг 5

```
//Автокалибровка АЦП,=AVR. Ступень 8=, РА, №9, 2005 г     =1
//Make: avr83,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="0" (Генератор 1 МГц)  =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
#define RIZM 200  //Сопротивление измер. резистора в Омах =5
#define ION 1298 //Напряжение внутреннего ИОН (1,23) в мВ =6
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =7
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =8
extern void lcd_init(void);            //Инициализация ЖКИ =9
unsigned char t0[]=" ==WATTMETER==          mWt     "; //=10
unsigned long vref=0,volt,watt,delta,i,d=200,avcc;     //=11
unsigned int a;                  //Вспомогательный счетчик =12
//-----------Функция автокалибровки АЦП----------------- =13
void calib (void)   //Поиск оптимального напряжения Vref =14
{ //Регистр ADMUX: АЦП 10 бит, Vref=AVCC (5В), ИОН=1,23В =15
  ADMUX &= 0xDF & 0x7F & 0xFE; ADMUX |= 0x40 | 0x0E;   //=16
//Регистр ADCSRA: вкл. АЦП, одиночный пуск, Fацп=62 кГц  =17
  ADCSRA &= 0xDF & 0xFC; ADCSRA |= 0x80 | 0x40 | 0x04; //=18
  for (volt=0, a=100; a>0; a--) //Усреднение 100 замеров =19
  { ADCSRA |= 0x40;         //Запуск нового измерения АЦП =20
    while (ADCSRA & 0x40);   //Проверка окончания замера =21
    volt += ADCL;    //Чтение младших 8 битов результата =22
    volt += ((int)ADCH << 8);    //Плюс два старших бита =23
  }                       //Окончание 100 замеров напряжения =24
  for (avcc=4750; avcc<5250; avcc++) //Диапазон AVCC, мВ =25
  { i = volt*avcc/102400; //Текущее значение ИОН (1,23В) =26
    if (i > ION) delta=i-ION;   //Положительная разность =27
    else delta=ION-i;           //Отрицательная разность =28
    if (delta < d)    //Если меньше минимальной разности =29
    { d=delta;    //Запомнить новую минимальную разность =30
      vref=avcc; //Запомнить оптимальное напряжение Vref =31
    }                  //Окончание сохранения новых значений =32
  }      //Окончание поиска оптимального напряжения Vref =33
}                  //Окончание функции автокалибровки АЦП =34
//================ОСНОВНАЯ ПРОГРАММА=================== =35
int main(void)                 //Начало основной программы =36
{ PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =37
  PORTC = 0xF0; DDRC = 0x05;   //PC0, PC2 выходы с лог.0 =38
  lcd_init();           //Инициализация ЖКИ (4 бит, 16x2) =39
  for (lcd_com(0x80), a=0; a<32; a++)  //Начальный текст =40
  { if (a==16) lcd_com(0xC0); //Переход на нижнюю строку =41
    lcd_dat(t0[a]);               //Вывод текущего символа =42
  }       //Окончание вывода начальной надписи WATTMETER =43
  calib(); //Автокалибровка АЦП по внутреннему ИОН 1,23В =44
  ADMUX &= 0xF3; ADMUX |= 0x03;   //Подключение канала-3 =45
  ADCSRA |= 0x20 | 0x40;   //Пуск постоянных замеров АЦП =46
  while (1)                              //Бесконечный цикл =47
  { for (a=65000; a>0; a--);       //Пауза для индикации =48
    volt = ADCL;     //Чтение младших 8 битов результата =49
    volt += ((int)ADCH << 8);    //Плюс два старших бита =50
    watt=(vref-volt*vref/1024)*(volt*vref/1024)/RIZM; // =51
    lcd_com(0xC4);               //Установка курсора ЖКИ =52
    lcd_dat(watt/1000 + 0x30); //Единицы милливатт (мВт) =53
    lcd_dat(',',); lcd_dat((watt/100)%10 +0x30); //0,1мВт =54
  }                      //Переход к новому измерению АЦП =55
}              //WinAVR-20050214, длина кода 1342 байтов =56
```

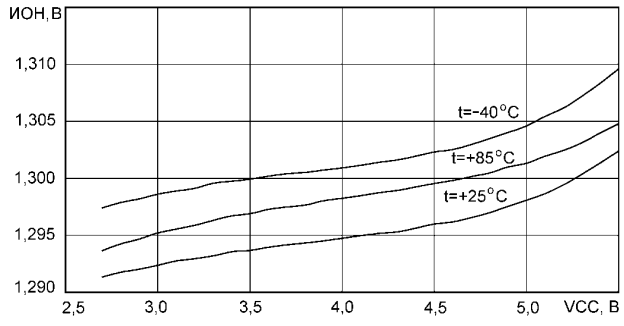

рис.7

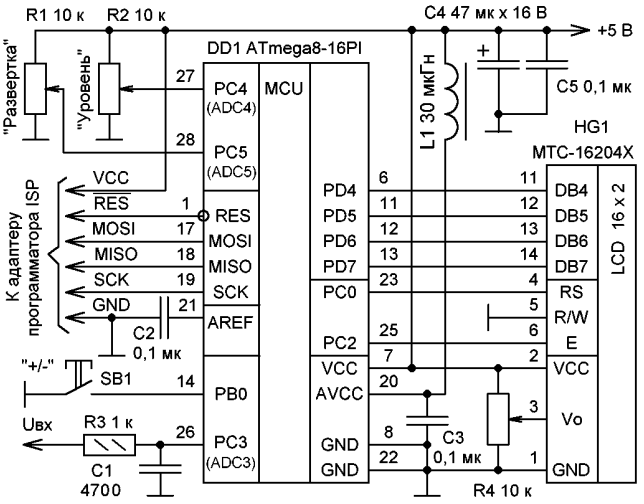

рис.8

пряжение внешним вольтметром нельзя, так как оно не выводится наружу. Однако в DATASHEET в разделе "Typical Characteristics" изображен график зависимости напряжения "bandgap" (это и есть ИОН 1,23 В) в зависимости от температуры и питания (**рис.7**). Константа ION выбрана по графику в точке пересечения напряжения 5 В и линии комнатной температуры. Разумеется, приведенные зависимости типовые и у разных экземпляров микросхем могут отличаться, но практика показывает высокую достоверность результатов.

Строка 25. Диапазон сканирования напряжения AVCC выбран ±5% от номинала 5 В. При другом разбросе питания числа 4750 и 5250 надо скорректировать.

Строки 26–32. Принцип автокалибровки заключается в подборе такого виртуального напряжения AVCC, при котором измеренная величина ИОН будет максимально близкой к константе ION. Результат работы функции "calib()"помещается в переменную "vref".

Строка 44. Функция "calib()" работает только при первом включении питания, но теоретически ее можно вызвать из других частей программы, проводя периодическую самокалибровку без участия человека.

Строка 51. Формула для расчета мощности потребления ЖКИ: P=(VCC–Ur)·Ir, где Ur, Ir соответственно напряжение и ток на резисторе R5.

Строки 53, 54. Индикация мощности ограничена двумя значащими цифрами, хотя можно было бы вывести все четыре. Решающим фактором здесь стала предварительная практическая проверка, которая показала, что мощность потребления носит импульсный быстропеременный характер. Действительно, нет смысла индицировать сотые и тысячные доли милливатта, когда даже десятые доли меняются 2–3 раза в секунду.

Цифровой осциллограф

В наше время портативным цифровым осциллографом с экраном на ЖКИ никого не удивишь. Фирмы Fluke, Velleman, Metrix, Hameg, Tektronix предлагают любую модель, но цены...

А ведь сделать самодельный осциллограф не так-то и сложно, особенно если освоена работа с АЦП. Из крупных деталей – всего ничего: МК, ЖКИ, 3 переменных резистора да кнопка (**рис.8**).

Параметры осциллографа:

диапазон входных напряжений 0...5 В;

диапазон входных частот 0...33 кГц;

количество отображаемых на ЖКИ аналоговых уровней – 8;

максимальная частота дискретизации – 66 кГц;

число элементов в одной строке развертки – 16;

кнопочный выбор синхронизации по переднему и заднему фронтам входного сигнала;

режим запоминания одной строки изображения;

плавная регулировка скорости развертки;

плавная регулировка уровня синхронизации.

Главным элементом осциллографа является МК DD1. Он обслуживает три канала АЦП: PC3 (входной сигнал), PC4 (регулятор уровня синхронизации), PC5 (регулятор скорости развертки). Индикатор HG1 любой двухстрочный, совместимый с системой команд HD44780, с подсветкой или без нее. Цепочка R3C1 – входной фильтр и, по совместительству, токоограничитель. Элементы L1C3 фильтруют питание АЦП. Кнопка SB1 в исходном состоянии формирует режим синхронизации от низкого уровня к высокому ("+"), в нажатом состоянии – от высокого к низкому ("–").

Переменные резисторы R1, R2 специально подключены к линиям PC4, PC5. С них снимаются данные о положении угла поворота резисторов, для чего достаточно точности 8 бит. Линии PC4, PC5 в "старых" микросхемах ATmega8 как раз и имели пониженную точность 8 бит, поэтому в осциллограф можно ставить любой экземпляр МК независимо от года выпуска.

Практическое задание. Отмакетировать электрические схемы вольтметра, амперметра, ваттметра, проверить их работу в симуляторе VMLab. Собрать схему цифрового осциллографа.

От редакции. *Описание программы для осциллографа, его модификации и пример практического использования будут рассмотрены в "Ступени 9".*

Микроконтроллеры AVR. Ступень 9

С.М. Рюмик, г. Чернигов

Любители построили "Ковчег", а профессионалы – "Титаник". (Из Интернета)

В предыдущей статье цикла (РА 9/2005) была приведена электрическая схема цифрового осциллографа на ЖКИ. "Вдохнуть в него жизнь" поможет управляющая программа для микроконтроллера (МК) DD1, но перед ее составлением надо продумать логику обработки данных.

Казалось бы, чего проще – оцифровал через АЦП текущую амплитуду входного сигнала и вывел ее на экран ЖКИ. Не тут то было! Основная прелесть любого осциллографа заключается в возможности синхронизации развертки, когда периодический сигнал можно сделать неподвижным на экране, а затем не спеша разглядеть его детали.

В цифровом осциллографе, выполненном на матричном ЖКИ, электронный луч и растр отсутствуют. Для синхронизации развертки придется применить нестандартный прием.

Алгоритм работы цифрового осциллографа

На **рис.1** показана логическая структура ячеек памяти и два вида входных сигналов Uвх. Предположим, что в программе создан массив из 32 однобайтовых чисел osc[32]. Под каждое знакоместо экрана отводится один элемент массива. Чтобы информация была непрерывной во времени, массив распространяется как на видимые, так и на невидимые участки ЖКИ.

Алгоритм работы цифрового осциллографа можно условно разделить на 3 фазы.

В *фазе измерения* происходит оцифровка через АЦП амплитуды входного сигнала с выбранной частотой дискретизации. При этом последовательно заполняются все 32 элемента массива osc[32]. Каждый элемент содержит число, пропорциональное уровню сигнала в конкретный момент времени.

В *фазе синхронизации* анализируется содержимое массива osc[32] и определяется ближайший момент перехода чисел от меньшего к большему (или наоборот) с заданным порогом. Фактически в массиве ищется передний или задний фронт сигнала, что отмечено на диаграммах Uвх, а) и Uвх, б) стрелками. Поскольку начало заполнения массива произвольно во времени, то фронты обычно не попадают в одинаковые по номерам ячейки. Их надо сместить во времени или, по-другому, синхронизировать. Для этого выполняется следующая процедура. Как только фронт найден, от него отсчитываются вправо 16 порядковых элементов массива, которые и выводятся на экран ЖКИ. Например, для сигнала Uвх, а) это ячейки 5–21, для сигнала Uвх, б) – 15–31. Теперь понятно, почему в массиве 32, а не 16 элементов, иначе "хвост" сигнала будет "отрезан".

В *фазе индикации* число, содержащееся в текущем элементе массива osc[32], преобразуется в символ псевдографики, который представляет собой горизонтальную линию той или иной высоты. В итоге на экране ЖКИ появляется графический рисунок из 16 разновысотных линий, по которым и судят о форме входного сигнала. Число возможных горизонталей – 8, следовательно, разрешение по амплитуде составляет 3 разряда.

При быстром изменении уровня сигнала возможны наложения одних горизонтальных линий на другие. Чтобы этого не происходило, в конце фазы индикации вводится задержка времени примерно на 0,1 с. Диапазон паузы можно менять программно вплоть до полной остановки изображения (фотографический снимок экрана).

Управляющая программа для цифрового осциллографа приведена в **листинге 1**.

Пояснения к листингу 1

Строки 2, 5–7 указывают на то, что компилировать программу "avr91.c" нужно совместно с файлом "lcd.c", в котором находятся внешние функции управления ЖКИ (см. "Ступень 7").

Строка 3. Фьюзы настраивают внутренний генератор МК на частоту Fclk=8 МГц, чтобы повысить быстродействие обработки информации. Типичная ошибка, когда фьюзы по забывчивости не меняют, и они остаются от прежнего эксперимента, например, на частоте 1 МГц. В результате вновь собранное устройство начинает "тормозить".

Строка 8. Текст заставки написан по-русски, но перед компиляцией программы надо обработать файл "avr91.c" утилитой "rusIcd.exe" *(имеется на сайте журнала "Радіоаматор"*

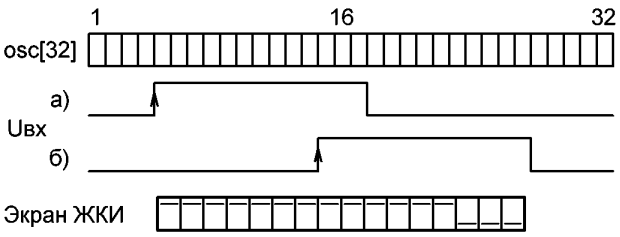

рис.1

Листинг 1

```
//Осциллограф на ЖКИ (АЦП),=AVR. Ступень 9=, РА, №10-2005 =1
//Make: avr91,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c =2
//Фьюзы: SUT0=CKSEL3=CKSEL1=CKSEL0="0" (Генератор 8 МГц)  =3
#include <avr/io.h>              //Библиотека ввода-вывода =4
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =5
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =6
extern void lcd_init(void);            //Инициализация ЖКИ =7
unsigned char t[]="        мкс/дел ";    //Текст заставки =8
//================ОСНОВНАЯ ПРОГРАММА==================== =9
int main(void)                 //Начало основной программы =10
{ unsigned char a, b, c, scan, ur, h;          //Счетчики =11
  unsigned int izm, d, k;        //Счетчики больших чисел =12
  unsigned int osc[32];   //Массив амплитуд осциллографа =13
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =14
  PORTC = 0xC2; DDRC = 0x05;   //PC0, PC2 выходы с лог.0 =15
  ADMUX &= 0x7F; ADMUX |= 0x20 | 0x40;  //8-10 бит, AVCC =16
//Регистр ADCSRA: вкл.АЦП, постоян. измерен., Fацп=1 МГц =17
  ADCSRA &= 0xFB; ADCSRA |= 0x80 | 0x40 | 0x20 | 0x03; //=18
  lcd_init();           //Инициализация ЖКИ (4 бит, 16x2) =19
  lcd_com(0x40); lcd_dat(0x00); //Начало знакогенератора =20
  for (a=1; a<63; a++)  //Загрузка 8 свободных знакомест =21
  { if (a%7 == 0) lcd_dat(0x1F);      //В строке 5 точек =22
    else lcd_dat(0x00);       //Пустая строка, нет точек =23
  }       //Окончание загрузки 62 байтов знакогенератора =24
  lcd_dat(0x00); //Последний (64-й) байт знакогенератора =25
  for (lcd_com(0xC0), a=0; a<16; a++) lcd_dat(t[a]);  // =26
  while (1)                  //Бесконечный цикл измерений =27
  { ADMUX &=0xF5; ADMUX |=0x05;     //Канал-5, РАЗВЕРТКА =28
    for (a=5; a>0; a--) for (d=60000; d>0; d--); //Пауза =29
    scan = (ADCH <= 5)? 1 : (ADCH - 4); //Развертка, мкс =30
    ADMUX &=0xF4; ADMUX |=0x04; //Канал-4, УРОВЕНЬ СИНХР =31
    lcd com(0xC2);                 //Установка курсора ЖКИ =32
    for (k=13*scan, d=10000, b=5; b > 0; b--, d=d/10) // =33
    { lcd_dat(((k / d)%10) + 0x30); //Вывод текущ. цифры =34
    }  //Окончание вывода 5 цифр времени развертки в мкс =35
    ur = (ADCH <= 10)? 0 : ADCH; //Уровень синхрониз., В =36
    ADMUX &= 0xF3; ADMUX |= 0x03; //Канал-3, ВХОД осцил. =37
    for(lcd_com(0x80), a=0; a<32; a++)   //32 замера АЦП =38
    { for (izm=0, b=scan; b > 0; b--)  //Время развертки =39
      { while (!(ADCSRA & 0x10));   //Проверка измерения =40
        ADCSRA |= 0x10;   //Разрешение следующего замера =41
        izm += ADCH;              //Накопление результата =42
      }                 //Окончание очередного замера АЦП =43
      osc[a] = izm;         //Заполнение массива амплитуд =44
    }               //Окончание 32 замеров амплитуды АЦП =45
    lcd_com(0x0D);                 //Включение курсора ЖКИ =46
    if (!ur) for (a=0; a<16; a++) lcd_dat((osc[a]/scan)/32);
    else  //Если синхронизация или остановка изображения =48
    { if (ur < 0xF0)     //Если нет остановки изображения =49
      { for (a=b=c=h=0; a<16; a++) //Поиск синхронизации =50
        { if (bit_is_set(PINB,PB0)) c=1; //Кнопка SB1(+) =51
          else b=1;   //Иначе синхронизация от <+> к <-> =52
          if((osc[a+b]/scan<(ur-3))&&(osc[a+c]/scan>(ur+3)))
          { h=a; a=32;       //Досрочный выход из поиска =54
          }              //Синхронизация выполнена успешно =55
        }     //Окончание процедуры поиска синхронизации =56
        for (a=h; a<(h+16); a++) lcd_dat((osc[a]/scan)/32);
      }  //Окончание прорисовки графика с синхронизацией =58
      else lcd_com(0x0C);  //Выкл. курсора при остановке =59
    }         //Завершение процедуры поиска синхронизации =60
  }              //Переход к новому циклу измерений АЦП =61
}              //WinAVR-20050214, длина кода 882 байтов =62
```

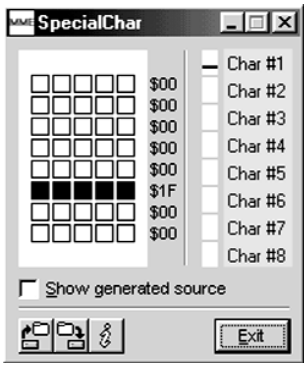

рис.2

http://www.ra-publish.com.ua *в дополнительных материалах к "Ступени 7").*

Строка 13. Размерность массива osc[32] выбрана "int", а не "char". Дело в том, что по ходу измерений будет производиться усреднение результатов с предварительной записью в массив чисел, больших, чем 255.

Строки 16–18 взяты из табл.1 "Ступени 8". Частота Fацп=1 МГц получается делением частоты Fclk на коэффициент 8. Значение Fацп выбрано максимально допустимым для ATmega8, чтобы обеспечить высокую частоту дискретизации. Плата за удовольствие – снижение точности замеров АЦП на 3 единицы. Для нашего случая это не принципиально, ведь на экране ЖКИ индицируются только 8 уровней из 256 возможных.

Строки 20–25 загружают в ЖКИ собственный знакогенератор по адресам 0x00–0x07 аналогично "Ступени 7". Каждый символ знакогенератора состоит из одной горизонтальной линии, пропорциональной по высоте амплитуде входного сигнала. В частности, нулевой уровень 0...0,625 В соответствует самой нижней, а напряжение питания 4,375...5 В – самой верхней линии.

На **рис.2** для примера показан процесс формирования одного символа знакогенератора в программе SpecialChar (автор B. Mueller, ftp://ftp.mme-berlin.de/avr/specialchar.zip, 98 Кб). Как видно, все байты нарисованного символа содержат нули $00, кроме кода $1F в шестом элементе. Именно это число и фигурирует в строке 22.

Строки 28–30 – измерение амплитуды сигнала в канале-5. На схеме осциллографа (рис.8, "Ступень 8") к этому каналу подключен переменный резистор R1 "Развертка". В зависимости от положения его движка на вывод 28 МК будет подаваться напряжение 0...5 В. Внутренний АЦП измеряет это напряжение и заносит в переменную "scan" число 0x00–0xFF. Получается простейший преобразователь "угол поворота резистора – числовой код".

Пауза в *строке 29* выполняет две функции. Во-первых, она нужна, чтобы в АЦП зафиксировалось устойчивое значение амплитуды после смены каналов. Во-вторых, именно она определяет время паузы для наблюдения данных на экране ЖКИ. Увеличить или уменьшить ее можно изменением выражения "a=5".

Строка 30. Типичный прием программного устранения так называемого "нулевого люфта" переменного резистора, когда в крайних положениях его движка имеется определенное начальное (часто неустойчивое) сопротивление. Изменить порог можно числами "5" и "4".

Строки 31–36 аналогичны строкам 28–30, но для канала-4, к которому подключен переменный резистор R2 "Уровень". Паузу обеспечивают строки 32–35, в которых на экран ЖКИ выводятся цифры времени развертки в микросекундах. Число "13" в строке 33 означает минимальную длительность одного деления 13 мкс. Это расчетная величина для Fацп=1 МГц.

Строка 33. Пример сверхкомпактного построения функции "for", когда в ее теле находятся целых три чужеродных элемента – два в начале и один в конце. Рекомендовать такой прием для всеобщего применения нельзя, так как сложно писать к нему комментарии.

Строка 34. На экран выводится 5 значащих цифр длительности развертки (определяется числами "10000" и "5" в строке 33), но реально используются только последние 4 (максимальное значение 03263 мкс). Старший разряд всегда равен "0", но он пригодится на будущее, например, при сдвиге шкалы развертки в сторону больших значений.

рис.3

Строка 36. Переменная "ur" содержит число, пропорциональное углу поворота резистора R2. Но вблизи крайнего положения движка оно принудительно обнуляется. Получается небольшая "мертвая" зона, когда сопротивление резистора меняется, а измеренный код все равно равен "0". Этот момент будет использован далее в строке 47 для срыва синхронизации.

Строки 40. Хотя в строке 18 установлен режим постоянной работы АЦП, но проверка окончания измерения в цикле "while" поможет избежать пропусков текущих значений. Кстати, режим постоянной работы АЦП на 7% быстрее, чем режим одиночных измерений.

Строки 46, 59. Курсор на экране ЖКИ представлен в виде мигающего прямоугольника. Его наличие означает, что осциллограф проводит измерения, а отсутствие – изображение остановлено для просмотра.

Строка 47. Еще один пример компактной упаковки листинга в одну строку. Выражение "!ur" эквивалентно "ur==0", но на 2 символа короче. Функции "for" и "lcd_dat" записаны без фигурных скобок, что допускается правилами языка Си для однокомпонентных выражений. Физически в данной строке производится вывод на ЖКИ осциллограммы из 16 псевдосимволов, абсолютный адрес которых получается делением содержимого массива osc[32] на переменную "scan" и число "32".

Строки 50–56. Процесс поиска синхронизации проиллюстрирован ранее на рис.1. В зависимости от положения кнопки SB1 в массиве osc[32] ищется восходящий или нисходящий фронт. В строке 53 введен небольшой гистерезис "ur-3", "ur+3", чтобы синхронизация не наступала от незначительных колебаний амплитуды. Если известно, что сигнал имеет крутые фронты, то число "3" можно увеличить.

Эксплуатация цифрового осциллографа

После прошивки МК осциллограф готов к работе. Желательно, чтобы резисторы R1, R2 имели линейную зависимость сопротивления от угла поворота (характеристика "А"), иначе будет сложнее проводить точную подстройку изображения.

Первоначально надо установить движок резистора R2 "Уровень" в среднее положение и убедиться, что на осциллографе появились цифры длительности развертки вместе с мигающим курсором. Далее подать на вход прибора исследуемый сигнал или, для эксперимента, постоянное напряжение в диапазоне 0...5 В. Вращая резистор R1 "Развертка", добиться максимальной амплитуды и стабильности сигнала. Резистором R2 в нижнем по схеме положении можно остановить картинку (курсор перестанет мигать). В верхнем положении принудительно срывается синхронизация, при этом изображение начинает "плыть", показывая истинную динамику формы сигнала.

Небольшое пояснение. В цифровых осциллографах в отличие от обычных аналоговых существует проблема неоднозначности измерений, вызванная самой природой аналого-цифрового преобразования. Например, если индицировать прямые выборки АЦП, то высокочастотные импульсы при большой длительности развертки могут давать на экране паразитную амплитудную модуляцию, очень похожую на настоящий сигнал. Увидеть этот эффект позволяет замена строки 44 листинга 1: "osc[a]=ADCH;". Чему же верить, если на экране осциллографа постоянно возникают "двойники"?

На помощь приходит цифровая фильтрация. В разрабатываемом осциллографе это простое усреднение замеров, которое эквивалентно фильтру низких частот. В итоге амплитуда паразитных сигналов резко уменьшается, правда, вместо "леса" вертикальных черточек, как в аналоговом осциллографе при больших развертках, появляется средний постоянный уровень.

Другая особенность осциллографа заключается в нестабильности воспроизведения импульсных сигналов с крутыми фронтами. На **рис.3** показано, как "плавают" по амплитуде передние и задние уровни, что связано, в первую очередь, с недостаточно высокой частотой дискретизации. Наблюдение таких сигналов носит оценочный характер, равно как и сигналов с частотами выше 25...30 кГц.

Уменьшить "дрожание" фронтов можно специальной математической обработкой выборок массива osc[32], адаптивным алгоритмом синхронизации, а также переходом от внутреннего генератора 8 МГц ATmega8 на внешний кварцевый. Кстати, в последнем случае улучшается достоверность измерений времени по экрану ЖКИ.

Практическое задание. Откомпилировать программу "avr91.c" и опробовать в работе осциллограф.

Микроконтроллеры AVR. Ступень 10

С.М. Рюмик, г. Чернигов

"Omnia mea mecum porto"
(Лат. "Все свое ношу с собой")

Цифровой осциллограф, о котором шла речь в предыдущих статьях цикла, позволяет видеть "живую" картинку сигналов на экране ЖКИ, а также запоминать 16 последних выборок АЦП. При исследовании редко повторяющихся процессов хотелось бы увеличить глубину выборок. Для этого необходим специальный запоминающий осциллограф. Хорошо, если бы он оказался простым, компактным и легко транспортируемым прибором.

По определению, запоминающий осциллограф – это устройство, позволяющее "сфотографировать" форму исследуемого сигнала и отображать ее на экране индикатора длительное время. Первые запоминающие осциллографы были аналоговыми с использованием электронно-лучевых трубок с большим временем послесвечения люминофора. Затем появились цифровые запоминающие осциллографы, принцип работы которых связан с аналого-цифровым преобразованием входного сигнала и запоминанием выборок во внутреннем ОЗУ или в энергонезависимом ПЗУ.

Если не ставить перед собой сверхзадачу и удовлетвориться количеством хранимых отсчетов в пределах 1000, то запоминающий осциллограф нетрудно выполнить на микроконтроллере (МК) ATmega8. Он будет полезен в радиолюбительской практике при анализе переходных и редко повторяющихся во времени процессов. Дополнительные требования к прибору: многоканальность, вывод информации на ЖКИ, кнопочный интерфейс просмотра формы сигналов, отдельный вход для синхронизации старта измерений.

Электрическая схема запоминающего осциллографа показана на **рис.1**. Поскольку на экране ЖКИ имеются две строки, то и число каналов выбрано таким же. Сигналы первого канала поступают на линию PC4, второго – на PC5. Если принять допустимую точность преобразования входных сигналов 8 разрядов, то в приборе можно использовать микросхемы ATmega8 любого года выпуска (см. "Ступень 9").

Пусковой синхронизирующий сигнал подается на линию PC3 микросхемы DD1. Он может быть как цифровым, так и аналоговым, порог напряжения срабатывания задается программно. Для навигации по массиву "сфотографированных" данных служат кнопки SB1, SB2, которые "листают" страницы осциллограмм сигналов вправо или влево по оси времени.

Резисторы R1–R3 защитные, на случай подачи внешних напряжений более +5,5 В и менее –0,5 В. Конденсатор C1 ослабляет помехи на синхровходе. Аналогичные конденсаторы могут быть установлены на выводах 27, 28 DD1, что определяется экспериментально. Их емкость зависит от частоты входных сигналов и уровня помех, обычно 4700 пФ...0,22 мкФ.

Нагрузками кнопок SB1, SB2 служат внутренние резисторы МК сопротивлением 20...50 кОм. Устанавливать токоограничивающие резисторы последовательно с кнопками в данном случае необязательно, ведь они нажимаются кратковременно. Кроме того, если произойдет сбой в программе и по "закону бутерброда" входы перенастроятся на выходы (на практике это маловероятно), то легко сделать оперативное обнуление прибора кнопкой сброса SB3.

Индикатор HG1 любой двухстрочный, совместимый с системой команд HD44780, с подсветкой или без нее. Подстроечный резистор R4 регулирует контрастность изображения. Элементы L1, C2-C5 – фильтры аналогового и цифрового питания.

Конструктивно осциллограф может быть выполнен в виде малогабаритного переносного прибора с питанием от батареи 4,5...4,8 В. Ток потребления 8...10 мА.

Управляющая программа для МК DD1 приведена в **листинге 1**.

Строка 8. Константу TIME выбирают из расчета 250 мкс на один замер АЦП. Например, при TIME=30 замеры будут следовать через каждые 7,5 мс. Допустимый диапазон TIME=1...65535, что позволяет проводить измерения с частотой от 4000 раз в секунду до 1 раза в 16 с.

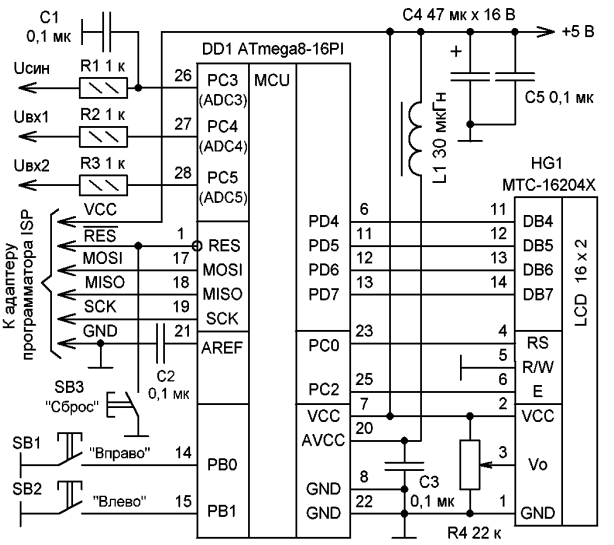

рис.1

Листинг 1

```
//Запоминающий осциллограф,=AVR. Ступень 10=, РА №11-2005 =1
//Make:avr101,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c =2
//Фьюзы: SUT0=CKSEL3=CKSEL2=CKSEL1="0" (Генератор 1 МГц)  =3
#include <avr/io.h>                 //Библиотека ввода-вывода =4
extern void lcd_com(unsigned char p);   //Ввод команд ЖКИ =5
extern void lcd_dat(unsigned char p);   //Ввод данных ЖКИ =6
extern void lcd_init(void);            //Инициализация ЖКИ =7
#define TIME 30 //Условная длительность одного замера АЦП =8
unsigned char t[]="Старт измерений";     //Текст заставки =9
//================ОСНОВНАЯ ПРОГРАММА=================== =10
int main(void)                  //Начало основной программы =11
{ unsigned char u1[450], u2[450]; //Массивы данных осцил.=12
  unsigned int a, b, c, d, h=0;         //Счетчики данных =13
  PORTB = DDRD = 0xFF; //B=входы с резисторами, D=выходы =14
  PORTC = 0xC2; DDRC = 0x05;   //PC0, PC2 выходы с лог.0 =15
  ADMUX &= 0x7F; ADMUX |= 0x20 | 0x40;  //8-10 бит, AVCC =16
//Регистр ADCSRA: включить АЦП, однократно, Fацп=125 кГц =17
  ADCSRA &= 0xDF & 0xFB; ADCSRA |= 0x80 | 0x40 | 0x03; //=18
  lcd_init();           //Инициализация ЖКИ (4 бит, 16x2) =19
  lcd_com(0x40); lcd_dat(0x00); //Начало знакогенератора =20
  for (a=1; a<63; a++)  //Загрузка 8 свободных знакомест =21
  { if (a%7 == 0) lcd_dat(0x1F);       //В строке 5 точек =22
    else lcd_dat(0x00);         //Пустая строка, нет точек =23
  }       //Окончание загрузки 62 байтов знакогенератора =24
  lcd_dat(0x00); //Последний (64-й) байт знакогенератора =25
  for(lcd_com(0x80),a=0; a<15; a++) lcd_dat(t[a]); //ЖКИ =26
  do  //Цикл проверки снижения напряжения Usan менее 4 В =27
  { ADMUX &=0xF3; ADMUX |=0x03; ADCSRA |=0x40; //Канал-3 =28
    while (ADCSRA & 0x40);   //Проверка окончания замера =29
  } while (ADCH > 0xCC); //Проверять, пока уровень > 4 В =30
  lcd_com(0x0C); //Выключение курсора при начале замеров =31
  for (a=0; a<450; a++) //Цикл заполнения массива данных =32
  { for (c=d=0, b=TIME; b>0; b--)    //Усреднение замеров =33
    { ADMUX &=0xF4; ADMUX |=0x04; ADCSRA |=0x40; //Кан.4 =34
      while (ADCSRA & 0x40); //Проверка окончания замера =35
      c += ADCH;  //Накопление амплитуды АЦП по каналу-4 =36
      ADMUX &=0xF5; ADMUX |=0x05; ADCSRA |=0x40; //Кан.5 =37
      while (ADCSRA & 0x40); //Проверка окончания замера =38
      d += ADCH;  //Накопление амплитуды АЦП по каналу-5 =39
    }          //Окончание цикла замеров с каналов-4 и -5 =40
    u1[a] = c/(TIME*32);  //Усредненный текущий замер U1 =41
    u2[a] = d/(TIME*32);  //Усредненный текущий замер U2 =42
  }          //Окончание заполнения массива данных U1, U2 =43
  while (1) //Бесконечный цикл индикации, повтор - сброс =44
  { lcd_com(0x80);  //Установка курсора в верхней строке =45
    for(a=h; a < h+15; a++) lcd_dat(u1[a]);  //График U1 =46
    lcd_com(0xC0);   //Установка курсора в нижней строке =47
    for(a=h; a < h+15; a++) lcd_dat(u2[a]);  //График U2 =48
    lcd_dat(0x30 + h/15);  //Условный номер блока данных =49
    if (bit_is_clear(PINB,PB0))      //Нажатие кнопки SB1 =50
    { if ((h += 15) > 435) h=0;  //Следующий блок данных =51
      for (c=65000; c>0; c--);  //Длительность индикации =52
    }          //Окончание увеличения номера блока данных =53
    if (bit_is_clear(PINB,PB1))      //Нажатие кнопки SB2 =54
    { h = (h < 15)? 435 : (h-15); //Предыдущий блок дан. =55
      for (c=65000; c>0; c--);  //Длительность индикации =56
    }          //Окончание уменьшения номера блока данных =57
  }              //Переход к прорисовке графиков данных =58
}              //WinAVR-20050214, длина кода 752 байтов =59
```

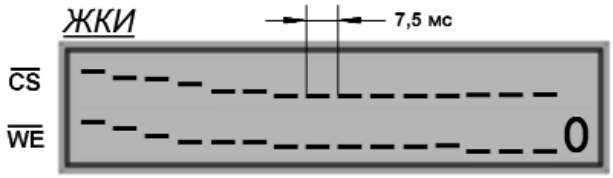

рис.2

Листинг 2

```
//Бортовой самописец-"черный ящик",=AVR.Ступень 10=, PA/11-2005 =1
//Make: avr102,atmega8,Level=2,VMLab,SRC=$(TARGET).c lcd.c      =2
//Фьюзы: BODLEVEL=BODEN=SUT0=CKSEL3=CKSEL2=CKSEL1="0" (1 МГц)   =3
#include <avr/io.h>                        //Библиотека ввода-вывода =4
#include <avr/eeprom.h>    //Библиотека функций работы с EEPROM =4a
. . .
    u1[a] = c/(TIME*32);        //Усредненный текущий замер U1 =41
    if (bit_is_set(PINB,PB0)) eeprom_write_byte(a+10,u1[a]); //=41a
. . .
    for(a=h; a < h+15; a++) lcd_dat(eeprom_read_byte(a+10)); //=46
. . .
}                          //WinAVR-20050214, длина кода 842 байтов =59
```

Строка 12. Общий объем двух массивов – 900 байтов. При желании его можно увеличить, но ненамного, так как объем ОЗУ в ATmega8 равен 1024 байта, из них примерно 20 байтов надо оставить на нужды программы.

Строки 16–18. Режим измерений 8/10-бит выбран по одной простой причине: в массивах u1[450], u2[450] как раз помещается по одному байту информации на каждый замер, т.е. по 8 разрядов. Строго говоря, для индикации формы сигнала на ЖКИ достаточно и трех разрядов, т.е. теоретически можно упаковать выборки в 2,6 раза плотнее. Но, вдруг точная 8-разрядная информация понадобится в дальнейшем для обработки в компьютере?

Строка 26 формирует текст начальной заставки.

Строки 27–30. Цикл проверки синхровхода PC3. Пока напряжение на нем больше, чем 4 В (определяется числом 0xCC в строке 30), то программа зацикливается, о чем свидетельствует мигающий курсор в верхней строке ЖКИ. Как только на PC3 появляется лог."0", курсор выключается (строка 31) и начинается цикл заполнения массива данных (строки 32–43).

Строки 36, 39, 41, 42. Каналы PC4, PC5 опрашиваются поочередно во времени, сначала PC4, затем PC5. Форма входных сигналов для каждого канала выводится в свою строку ЖКИ. Поскольку время между опросами каналов невелико (десятки микросекунд), то можно считать, что обе осциллограммы синхронно идут в режиме реального времени.

Строка 44. Для упрощения конструкции вместо отдельной кнопки повторного старта используется кнопка сброса SB3, после чего все измерения начинаются заново.

Строка 49. В крайний справа столбец нижней строки ЖКИ выводится условный номер текущей страницы – от цифры "0" (первая) до буквы "M" (последняя). Итого получается 30 страниц по 15x2 знакоместа.

Строки 50–57. Кнопки SB1, SB2 по кругу "пролистывают" страницы осциллограмм вправо (вперед во времени) или влево (назад во времени).

Эксперименты с запоминающим осциллографом

Промоделировать в симуляторе VMLab работу Си-программы листинга 1, к сожалению, не удастся. Модель ЖКИ в симуляторе "не понимает" загрузку символов внешнего знакогенератора, поэтому осциллограмм видно не будет. Придется проводить натурные испытания.

В качестве примера на **рис.2** показаны осциллограммы переходного процесса сигналов /CS и /WE в многоразовом SEGA-картридже [1]. Они были сняты запоминающим осциллографом сразу после выключения питания. Естественно, синхронизирующим сигналом выступало само напряжение питания картриджа. Как только оно понизилось с 5 до 4 В, автоматически началась запись показаний с частотой 1 раз в 7,5 мс. Аналогичные осциллограммы были сняты для разных схем хранения данных в ОЗУ SEGA-картриджа, что позволило выбрать наиболее устойчивый к просадкам питания вариант.

Бортовой самописец – "черный ящик"

В авиационной и космической технике часто упоминаются так называемые "черные ящики". В них записываются телеметрические данные о состоянии бортовых систем в ходе полета. В случае аварии специалисты могут расшифровать информацию, сохранившуюся в "черном ящике" и сделать определенные выводы о причинах неполадок.

Первые бортовые самописцы были аналоговыми. Их устройство, как правило, напоминало магнитофон с ударостойким лентопротяжным механизмом, где вместо магнитной ленты использовалась проволока. В дальнейшем появились цифровые "черные ящики", в которых телеметрия записывалась в магнитно-транзисторные ячейки, а затем и в микросхемы ПЗУ. Для сведения, у современных самописцев объем памяти исчисляется гигабайтами.

Почувствовать тонкости работы "черного ящика" можно на примере запоминающего осциллографа (рис.1), вспомнив, что МК DD1 может самостоятельно записывать информацию во внутреннее EEPROM. В чем разница между запоминающим осциллографом и самописцем? В данном примере, в том, что при выключении питания или нажатии кнопки сброса SB3 данные о замерах в осциллографе теряются, а в самописце – сохраняются.

В **листинге 2** приведены строки, которые необходимо вставить или заменить в листинге 1, чтобы осциллограф приобрел функцию самописца.

Строка 3. Напоминание о флажках BODLEVEL, BODEN, определяющих порог срабатывания детектора BOD на уровне 4 В, чтобы при медленном нарастании (спаде) питающего напряжения не повредилась информация в EEPROM. Это необходимо учесть при прошивке "фьюзов".

Строка 4а вводится между строками 4 и 5. Здесь подключается системная библиотека EEPROM, первое знакомство с которой состоялось в "Ступени 7".

Строка 41а вводится между строками 41 и 42. В ней проверяется состояние кнопки SB1, которая подключена к линии PB0. Если при включении питания или начальном сбросе она не нажата, то массив данных u1[450] при измерениях будет записываться напрямую в EEPROM, а затем высвечиваться на ЖКИ. Если кнопка SB1 нажата, то запись блокируется, и на индикацию будет выведен последний сохранившийся массив (вот он, "черный ящик"!).

Функция "eeprom_write_byte" имеет в скобках два операнда: первым указывается адрес записи (a+10), вторым – данные записи (u1[a]). Поскольку переменная "a" меняется от 0 до 449, то и данные в EEPROM будут записываться по адресам 10...459 (всего доступно 0...511). Количество допускаемых процедур записи в ATmega8 не менее 100 тысяч раз.

Почему запись ведется с адреса 10, а не с нуля? По рекомендациям бывалых электронщиков из Интернета. Они помнят, как в микросхемах AVR первого поколения возникали проблемы с искажением данных чаще всего в начальных ячейках EEPROM. Микросхемы второго поколения имеют встроенный детектор BOD и практически полностью защищены от этого эффекта. Но, "привычка – вторая натура" и "кашу маслом не испортишь".

Строка 46 обеспечивает чтение данных из EEPROM по методике, рассмотренной ранее в "Ступени 7".

Строка 59 содержит информацию об увеличившемся объеме кода программы. Интересный нюанс выявится при компиляции. Появятся два сообщения: "avr102.c:43: warning: passing arg 1 of `eeprom_write_byte' makes pointer from integer without a cast" и "avr102.c:49: warning: passing arg 1 of `eeprom_read_byte' makes pointer from integer without a cast". Их суть в том, что компилятор предупреждает программиста о возможности выхода переменной "a" за допустимый для EEPROM предел 512 байтов. Надо мысленно поблагодарить WinAVR за напоминание и продолжить работу дальше.

Практическое задание. Отладить запоминающий осциллограф и самописец. Изменить программу так, чтобы объем хранимых выборок увеличился сначала до 2600 (переход к трехразрядному разрешению), а затем до 3000...4000 за счет применения алгоритмов сжатия данных по аналогии с zip-файлами.

Литература

1. Рюмик С.М. Многоразовый SEGA-картридж с сохранением позиций//Радіоаматор. – 2005. – №10, №11.