

Н. Королёв т. (095) 217-2487
Д. Королёв т. (095) 217-2519

e-mail:korolev@argussoft.ru
e-mail:atmel@argussoft.ru

ATMEL: AVR-микроконтроллеры в 2001 году.

Микроконтроллеры семейства AT90S (AVR-микроконтроллеры) за сравнительно короткое время завоевали заслуженную популярность во всем мире. Совокупность характеристик - современная RISC архитектура, многократные перепрограммируемые ФЛЭШ-ПЗУ программ и ЭСПЗУ данных, возможность программирования в системе и наличие битов защиты от несанкционированного копирования - делает AVR-микроконтроллеры исключительно удобной элементной базой для построения разнообразных приборов – от простейших домашних игрушек до серьезных систем промышленной автоматики и устройств автомобильной электроники. В предлагаемой статье приводится обзор AVR-микроконтроллеров, выпускаемых в настоящее время, а также объявленных к выпуску в 2001 году.

AVR-микроконтроллеры разделяются на три семейства: tiny, classic и mega. Сначала члены этих семейств имели четко выраженные “родовые” признаки, проявляющиеся в числе выводов корпусов: tiny – корпус 8 выводов, classic – корпус 40-44 вывода, mega – 64-выводной корпус. В настоящее время граница между семействами стала весьма размытой: с одной стороны, выпущена микросхема Attiny28 в 20-выводном корпусе, с другой стороны, объявлены микросхемы Atmega161, которые будут поставляться в 40- и 44- выводных корпусах. Маленькая подсказка: в названии AVR-микроконтроллеров первые цифры обозначают объем ПЗУ программ в килобайтах. Таким образом, микросхема AT90S100 имеет 1 кбайт ПЗУ, AT90S8515 - соответственно 8 кбайт (однако, ATmega103 содержит 128 кбайт программной памяти). В 2000 г. Фирма ATMEL производит реорганизацию производства. Этот процесс затронул и AVR-микроконтроллеры. Многие кристаллы будут выпускаться по технологическим нормам 0,35 мкм, что позволит увеличить объем производства популярных микроконтроллеров и снимет проблему дефицита, возникшую вследствие повышения спроса на них в мире.

Микросхемы *tiny* оптимальны для использования в недорогих приборах типа интеллектуальных датчиков. Они характеризуются малой потребляемой мощностью. Нижняя граница напряжений питания составляет 1,8 В для микросхем с индексом «V», 2,7 В для микросхем с индексом «L» и 4,0 В для микросхем без индекса. Верхняя граница напряжения питания для всех микросхем составляет 5,5 В. При напряжении питания 3 Вольта и частоте генератора 4 МГц ток потребления в активном режиме составляет 2,2 мА, в режиме Idle – 0,5 мА, в режиме Power Down – менее 1 мкА. Выход из “спящего” состояния происходит по изменению сигнала на любом выводе микроконтроллера. Микросхемы ATtiny10 представляют собой вариант ATtiny11 с однократным программированием. Эти микросхемы поставляются в партиях от 10 тысяч штук. ATtiny28 оптимизирована для работы в пультах дистанционного управления.

Расширенным набором функций в семействе Attiny обладает микросхема ATtiny12. В этой микросхеме шесть выводов имеют функции ввода/вывода сигналов, в качестве задающего генератора можно использовать дополнительный RC-генератор, размещенный на кристалле. Особенность этого генератора – возможность подстройки частоты путем записи байта в специальный регистр OSCCAL. Запись значения 00 соответствует минимальной частоте генератора, запись значения от 01 до FF приводит к пропорциональному увеличению значения частоты RC-генератора.

Микроконтроллер ATtiny12 оснащен схемой слежения за уровнем питающего напряжения (BOD, brown – out detector). Если работа этой схемы разрешена (установлен бит BODEN), то, при снижении уровня напряжения ниже порога на время, превышающее 7 наносекунд, схема вырабатывает сигнал сброса. Порог срабатывания может быть выбран из двух значений: 1,8 В или 2,7 В. Работу схемы слежения обеспечивает встроенный источник опорного напряжения 1,22 В, который

может быть использован для формирования порогового напряжения встроенного аналогового компаратора.

Наиболее многофункциональным представителем семейства ATtiny является микроконтроллер ATtiny15. В дополнение к вышеперечисленным возможностям ATtiny15 имеет также быстрый ШИМ-модулятор и 4-канальный 10-битный АЦП. Неплохо для 8-выводного корпуса! Скорость ШИМ-модулятора повышена за счет использования более высокой задающей частоты, которая формируется из базовой (1,6 МГц) путем умножения на 16. Максимальная частота ШИМ-модулятора составляет 100 кГц.

Несколько слов об АЦП. Он работает как с одиночными, так и с дифференциальными входными сигналами. Для дифференциального входа предусмотрен входной усилитель с коэффициентом усиления 20. АЦП может работать в одном из двух режимов – одиночный запуск или непрерывная работа. Скорость работы АЦП зависит от задающей частоты, которая формируется из системной путем деления в 2... 128 раз. Рекомендованная максимальная входная частота АЦП – 200 кГц, хотя можно использовать и более высокие частоты. При этом реальная точность АЦП уменьшается до 8 – 9 бит. На частоте 200 кГц время преобразования составляет 65 микросекунд. Для снижения уровня помех от ядра микроконтроллера рекомендуется на время работы АЦП перевести микроконтроллер в спящий режим.

В таблице 1 представлены краткие технические характеристики микроконтроллеров *tiny*.

Таблица 1. Tiny AVR

Микросхема	ПЗУ программ	ОЗУ	ЭСПЗУ	Таймеры	Напр. питания	АЦП	Рабочая частота	Корпус выв.
ATtiny10	1 КВ	0	0	1	4,0 – 5,5	нет	0-6 MHz	8
ATtiny11	1 КВ	0	0	1	4,0 – 5,5	нет	0-6 MHz	8
ATtiny11L	1 КВ	0	0	1	2,7 – 5,5	нет	0-2 MHz	8
ATtiny12	1 КВ	0	64 b	2	4,0 – 5,5	нет	0-8 MHz	8
ATtiny12L	1 КВ	0	64 b	2	2,7 – 5,5	нет	0-4 MHz	8
ATtiny12V	1 КВ	0	64 b	2	1,8 – 5,5	нет	0-1 MHz	8
ATtiny15	1 КВ	0	64 b	2	4,0 – 5,5	4 вх.	1,6 MHz	8
ATtiny22L	2 КВ	128 b	128 b	1	2,7 – 5,5	нет	1 MHz	8
ATtiny28L	2 КВ	0	0	1	2,7 – 5,5	нет	0-4 MHz	28

Микроконтроллеры семейства *classic* не требуют подробного рассказа – о них уже достаточно написано. Следует отметить только изменения в номенклатуре. После перехода на технологические нормы 0,35 мкм некоторые микросхемы более не будут выпускаться. Это позиции, у которых есть аналоги с большим объемом памяти. Таким образом, вместо микросхем AT90S2333, AT90S4414 и AT90S4434 следует использовать соответственно AT90S4333, AT90S8515 и AT90S8535 в идентичных корпусах.

В таблице 2 представлены краткие технические характеристики микроконтроллеров *classic*.

Таблица 2. classic AVR

Микросхема	ПЗУ программ	ОЗУ	ЭСПЗУ	Таймеры	Посл. порт	АЦП	Рабочая частота	Корпус, выв.
AT90S1200*	1 КВ	0	64 b	1	нет	нет	0-12 MHz	20
AT90S2313	2 КВ	128 b	128 b	2	есть	нет	0-10 MHz	20
AT90S2323	2 КВ	128 b	128 b	2	нет	нет	0-10 MHz	8
AT90LS2323	2 КВ	128 b	128 b	2	нет	нет	0-4 MHz	8
AT90S2343*	2 КВ	128 b	128 b	2	нет	нет	0-10 MHz	8
AT90LS2343*	2 КВ	128 b	128 b	2	нет	нет	0-4 MHz	8

AT90S4433	4 КВ	128 b	256 b	2	есть	6 вх.	0-8 MHz	28, 32
AT90LS4433	4 КВ	128 b	256 b	2	есть	6 вх.	0-4 MHz	28, 32
AT90S8515	8 КВ	512 b	512 b	2	есть	нет	0-8 MHz	40, 44
AT90S8535	8 КВ	512 b	512 b	2	есть	8 вх.	0-8 MHz	40, 44
AT90LS8535	8 КВ	512 b	512 b	2	есть	8 вх.	0-4 MHz	40, 44

*Микросхемы AT90S1200 и AT90S2343 имеют встроенный RC-генератор на 1МГц.

Семейство *mega* на сегодняшний день представлено единственным членом - микроконтроллером ATmega103 (вариант с низковольтным питанием называется ATmega103L). Несмотря на небогатый выбор, микросхема оказалась настолько удачной, что потребность в этих микросхемы превысила все прогнозы. Объем выпуска ATmega103 в 2000 году практически удвоился, однако ажиотажный спрос на них во всем мире повлек за собой ужесточение сроков поставок и повышение отпускных цен в полтора раза. В 2001 году именно в семействе ATmega ожидается значительное пополнение.

Прежде всего – ATmega161. Этот микроконтроллер совместим по цоколевке с микросхемой AT90S8515 и включает в себя несколько новых блоков: аппаратный умножитель, второй последовательный порт, блок автопрограммирования.

Команда умножения двух 8-разрядных операндов (как знаковых, так и беззнаковых) выполняется за два такта, умножение двух 16-разрядных операндов занимает 17 тактов для беззнаковых чисел и 19 – для знаковых. Умножение с накоплением также выполняется за 19 тактов.

Микроконтроллер ATmega161 содержит два последовательных порта, имеющих идентичные характеристики. Порты аппаратно поддерживают режим работы в многопроцессорных конфигурациях и могут передавать данные на скорости до 912600 бод при частоте кварца 7,3728 МГц.

Режим автопрограммирования удобен для замены программы в удаленном микроконтроллере. Для реализации автопрограммирования в ПЗУ выделяется область для программы-загрузчика (boot-block) размером от 256 байт до 2 кбайт (устанавливается программно). Время записи сектора ПЗУ (128 байт) составляет 10 мс.

В таблице 3 представлены краткие технические характеристики микроконтроллеров *mega*.

Таблица 3. mega AVR

Микросхема	ПЗУ программ	ОЗУ	ЭСПЗУ	Таймеры	Посл. порты	АЦП	Рабочая частота	Корпус, выв.
ATmega103	128 КВ	4 КВ	4 КВ	3	есть	8 вх.	0-6 MHz	64
ATmega103L	128 КВ	4 КВ	4 КВ	3	есть	8 вх.	0-4 MHz	64
ATmega603	64 КВ	4 КВ	2 КВ	3	есть	8 вх.	0-6 MHz	64
ATmega603L	64 КВ	4 КВ	2 КВ	3	есть	8 вх.	0-4 MHz	64
ATmega161	16 КВ	1 КВ	512 b	3	есть	нет	0-6 MHz	40, 44
ATmega161L	16 КВ	1 КВ	512 b	3	есть	нет	0-4 MHz	40, 44

Отдельно следует рассказать о микросхеме ATmega163. Полные технические характеристики этого микроконтроллера в настоящее время не публикуются, поэтому он не внесен в таблицу. Эта “темная лошадка” также будет выпускаться в 40- и 44-выводных корпусах, однако без второго UART. Вместо этого у ATmega163 есть АЦП с расширенными функциями – два дифференциальных канала и входной предусилитель с коэффициентом усиления 1, 10 и 200. ATmega163 был запланирован к серийному выпуску на вторую половину 2000 г., однако впоследствии перенесен на весну 2001 г. Вариант ATmega163 с 8 кбайт ПЗУ программ будет называться Atmega83. А еще говорят о микросхемах Atmega85 и Atmega32...

Однако, вернемся к реальной жизни. Микросхема ATmega103 всем удобна, однако, для макетирования приходится искать плату, на которую можно распаять корпус TQFP-64, единственный корпус для ATmega на сегодня. Часто удобным выходом является приобретение набора STK300 фирмы ATMEL. Это макетная плата, на которой установлена микросхема ATmega103 или ATmega103L, панельки для внешнего ОЗУ 32 кбайта и для регистра-защелки адреса типа 74C373,

разъем для внешнего ЖКИ и микросхема ADM202 для стыковки с портом RS-232. В комплект также входит программирующий кабель, подключаемый к параллельному порту компьютера. Использование такой платы существенно ускоряет этап разработки, однако опыт работы выявил некоторые неудобства STK300. В итоге в фирме АРГУССОФТ Компани было создано аналогичное устройство с расширенными функциями - плата AS-mega, принципиальная схема которой приведена на рисунках 1 и 2. Отличия от STK300 состоят в следующем. Плата AS-mega предназначена для использования не только в качестве учебной, для изучения работы микроконтроллера ATmega103, но и для использования в составе конечного устройства с повышенными требованиями к надежности. Поэтому из платы исключены все панельки, а микросхема внешнего ОЗУ 62256 и регистр-защелка адреса 74C373 распаяны непосредственно на плату. Во многих случаях в конечном устройстве используется ЦАП, и в STK300 приходится добавлять внешнюю плату, подключаемую к разъему STK300. На плате AS-mega распаяна микросхема 8-разрядного последовательного ЦАП с выходом по напряжению AD5300 в корпусе microSOIC8. Вместо AD5300 можно запаять 10- или 12-разрядный ЦАП серии AD53XX в таком же корпусе. К выходу ЦАП подключен один канал операционного усилителя AD8532 с повышенной нагрузочной способностью. Второй канал этого ОУ подключен к одному из входов внутреннего АЦП ATmega103 в качестве входного усилителя с коэффициентом усиления 15. Практика показала, что на плате удобно иметь накопитель данных достаточно большой емкости. В качестве такого ПЗУ на плате AS-mega можно использовать микросхему последовательного ФЛЭШ-ПЗУ серии AT45D021... AT45D161 емкостью соответственно от 2 до 16 Мбит в корпусе SOIC28. Таким образом, плата AS-mega представляет собой законченное решение, имеющее блок ввода аналоговой информации, блок обработки оцифрованных данных, блок хранения данных и блок вывода аналоговой информации. Габаритные размеры платы – 95 x 86 мм. К плате также может быть подключен стандартный знакобуквенный ЖКИ с 8-разрядным интерфейсом. Для проверки функционирования узлов платы AS-mega, в нее “зашивается” демонстрационная программа, показывающая работу АЦП, ЦАП и последовательного порта ATmega103. Исходный текст этой программы приведен на рисунке 3. Работой платы управляет программа As-mega, функционирующая в среде Windows

Микроконтроллер ATmega103 программируется в схеме по интерфейсу SPI через стандартный 10-контактный разъем, идентичный разъему платы STK300. Для программирования платы AS-mega можно пользоваться загрузочным кабелем, входящим в состав STK300, однако, параллельный порт в компьютере обычно занят принтером или ключом защиты какого-либо программного пакета. Кроме того, программное обеспечение этого загрузочного кабеля неустойчиво работает под Windows NT/2000, что вынуждает пользователя устанавливать на компьютер Windows 98. Альтернативой является использование внутрисхемного программатора AS1, разработанного специалистами АРГУССОФТ Компани. Этот программатор подключается к компьютеру через последовательный порт, который обычно свободен. При этом скорость работы этого программатора в несколько раз выше. Например, чтение содержимого ПЗУ программы микроконтроллера ATmega103 через кабель из состава STK300 занимает почти две минуты (точнее - 105 секунд), а при использовании AS1 чтение происходит менее, чем за 17 секунд. Таким образом, при многократном перепрограммировании ATmega103 в течение рабочего дня достигается ощутимая экономия времени. Программное обеспечение программатора AS1 – программа ASisp - имеет более удобный пользовательский интерфейс и функционирует под всеми версиями Windows. Так как программа разработана в АРГУССОФТ Компани, всегда можно проконсультироваться по вопросу ее использования с разработчиками. Программа ASisp постоянно совершенствуется и дополняется новыми функциями.

Таким образом, применение AVR-микроконтроллеров фирмы ATMEL позволяет достигать конечного результата в минимальные сроки, а, учитывая возможность быстрого перепрограммирования непосредственно в конечном изделии, проводить модернизацию серийно выпускаемых приборов без каких-либо монтажных работ.

Текущую версию программы можно переписать с сайта <http://atmel.argussoft.ru>. Получить консультацию по применению платы AS-mega и программатора AS1 можно у специалистов фирмы АРГУССОФТ Компани по тел. (095) 217-2487, (095) 217-2519.

Рис.3 Демонстрационная программа для Atmega103.

```
; ASmega demo program
; version 1.0
;ARGUSSOFT Company - 2000. http://atmel.argussoft.ru
```

```
.include "m103def.inc"
```

```
; int vectors
interrupts:
.org 0
    rjmp reset
.org OVF0addr
    rjmp timer
.org 4*24
```

```
; register usage
```

```
.def zdata =r0 ; z register data
.def dac_1 =r1 ; dac temporary 1
.def dac_2 =r2 ; dac temporary 2
.def dac_3 =r3 ; dac temporary 3
.def dac_4 =r4 ; dac temporary 4
.def adc_low =r10; adc data low
.def adc_hi =r11 ; adc data high
.def temp =r16
.def offset =r17 ; dac buffer offset
.def u_data =r18 ; uart data
.def freq =r19 ; dac signal frequency
```

```
; reset vector
```

```
reset:
    ; stack pointer
    ldi temp,low(ramend)
    out spl,temp
    ldi temp,high(ramend)
    out sph,temp

    ; clear led
    sbi ddrb,0
    sbi portb,0

    ; button init
    cbi ddrd,0
    sbi portd,0

    ; uart init
    rcall uart_init

    ; adc init
    rcall adc_init

    ; dac init
    rcall dac_init

    ldi freq,0xFF-41

    ; timer init
    ldi temp,0b00000010 ; PCK0/8
    out tccr0,temp
    ldi temp,0b00000001 ; int enable
    out tmsk,temp
```

```

    out    tcnt0,freq

    ; load signal 0
    ldi    temp,2
    rcall load_signal

    sei

    ; loop
forever:
    rcall uart_receive          ; wait command

    cpi    u_data,1            ; set led
    breq   set_led
    cpi    u_data,2            ; clear led
    breq   clr_led
    cpi    u_data,'M'          ; change output signal
    breq   change_signal
    cpi    u_data,'F'
    breq   change_frequency    ; change frequency
    cpi    u_data,'A'
    breq   read_adc            ; one time adc read

next:
    ldi    u_data,0x0D
    rcall  uart_send
    rjmp   forever

set_led:
    cbi    portb,0
    rjmp   next

clr_led:
    sbi    portb,0
    rjmp   next

change_signal:
                                ; change dac signal form
    rcall  uart_receive
    mov    temp,u_data
    andi   temp,3
    rcall  load_signal
    rjmp   next

change_frequency:
                                ; change frequency
    rcall  uart_receive
    mov    freq,u_data
    rjmp   next

read_adc:
                                ; read adc
    rcall  adc_read
    mov    u_data,adc_hi
    rcall  uart_send
    mov    u_data,adc_low
    rcall  uart_send
    rjmp   next

; load address of temp signal buffer (256 bytes) into z-register
; temp = 0 : rectangle
; temp = 1 : peak
; temp = 2 : triangle
; temp = 3 : sine

```

```

load_signal:
    ldi    zl,low(signal0*2) ; load signal 0
    ldi    zh,high(signal0*2)
    add    zh,temp           ; add 256*temp
    clr    offset           ; clear offset
    ret

; uart interfacing

; init speed
uart_init:
    cbi    ddre,0
    sbi    ddre,1
    ldi    temp,11           ; 19200
    out    ubrr,temp
    ldi    temp,0b00011000
    out    ucr,temp
    ret

; uart send byte (u_data)
uart_send:
    sbis   usr,udre         ; data reg empty
    rjmp  uart_send
    out    udr,u_data
    ret

; uart receive byte (u_data)
uart_receive:
    sbis   usr,rxcom       ; rx complete
    rjmp  uart_receive
    in    u_data,udr
    ret

; adc interfacing

; init
adc_init:
    ldi    temp,1
    out    admux,temp
    sbi    adcsr,aden
    sbi    adcsr,0
    sbi    adcsr,adsc
adc_init_wait:
    sbis   adcsr,adif
    rjmp  adc_init_wait
    sbi    adcsr,adif
    ret

; read adc
adc_read:
    sbi    adcsr,adif
    sbi    adcsr,adsc

adc_read_wait:
    sbis   adcsr,adif
    rjmp  adc_read_wait

    sbi    adcsr,adif

    in    adc_low,adcl
    in    adc_hi,adch

```

```

    ret

; dac interfacing

; data == pe2
; ~sync == pe3
; sclk = pd6

; dac_1 = 0
; dac_2 = 0
; dac_3 = clock1 (porte)
; dac_4 = clock2 (porte)

; init
dac_init:
    ; dac pins

    sbi    ddre,2
    sbi    porte,2
    sbi    ddre,3
    sbi    porte,3
    sbi    ddrd,6
    sbi    portd,6

    ; these registers are used for fast dac operation
    clr    temp
    mov    dac_1,temp

    ret

; write byte
; fast write using registers dac_1 .. dac_4
dac_write:
    in     dac_2,porte
    ldi    temp,0xFF-(1<<3)
    and    dac_2,temp

    in     dac_3,portd
    in     dac_4,portd
    ldi    temp,(1<<6)
    or     dac_4,temp
    com    temp
    and    dac_3,temp

    cbi    porte,3 ; ~sync

    bst    dac_1,0 ; next bit
    bld    dac_2,2 ; to data
    out    porte,dac_2
    out    portd,dac_3 ; clock
    out    portd,dac_4

    bst    dac_1,0 ; next bit
    bld    dac_2,2 ; to data
    out    porte,dac_2
    out    portd,dac_3 ; clock
    out    portd,dac_4

    bst    dac_1,0 ; next bit
    bld    dac_2,2 ; to data

```

```
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  dac_1,0 ; next bit
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

;;;;;;;;;;;;;

bst  zdata,7 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,6 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,5 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,4 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,3 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,2 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,1 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

bst  zdata,0 ; next bit (data!)
bld  dac_2,2 ; to data
out  porte,dac_2
out  portd,dac_3 ; clock
out  portd,dac_4

;;;;;;;;;;;;;
```

```
bst dac_1,0 ; next bit
bld dac_2,2 ; to data
out porte,dac_2
out portd,dac_3 ; clock
out portd,dac_4
```

```
bst dac_1,0 ; next bit
bld dac_2,2 ; to data
out porte,dac_2
out portd,dac_3 ; clock
out portd,dac_4
```

```
bst dac_1,0 ; next bit
bld dac_2,2 ; to data
out porte,dac_2
out portd,dac_3 ; clock
out portd,dac_4
```

```
bst dac_1,0 ; next bit
bld dac_2,2 ; to data
out porte,dac_2
out portd,dac_3 ; clock
out portd,dac_4
```

```
sbi porte,3 ; ~sync
ret
```

```
; audio timer event
```

```
timer:
```

```
push temp ; push temp & sreg
in temp,sreg ; all other registers are used in cli-sti blocks
push temp
```

```
out tcnt0,freq ; init next cycle
```

```
push zl
push zh
```

```
clr temp
add zl,offset ; offset
adc zh,temp
```

```
lpm ; zdata = next data byte
```

```
rcall dac_write
```

```
inc offset
```

```
pop zh
pop zl
```

```
timer_done:
```

```
pop temp
out sreg,temp
pop temp
```

```
reti
```