

AVR-микроконтроллеры: большое в малом.

Микроконтроллеры серии AT90S фирмы ATMEL (AVR-микроконтроллеры) при своих скромных внешних габаритах имеют весьма насыщенное внутреннее содержание. Яркая иллюстрация – микросхема ATtiny15. Этот микроконтроллер, выпускаемый в корпусе с 8 выводами, помимо расширенного RISC-ядра с 32 регистрами-аккумуляторами, объединенными в регистровый файл, и программируемой в системе памяти программ (1 килобайт FLASH memory) и памяти данных (64 байта EEPROM), содержит также 4-канальный 10-битный АЦП с возможностью работы одного канала в дифференциальном режиме с дополнительным усилением 26 дБ. Остается только добавить, что в этой «крошке» (tiny – в переводе с английского – крошечный, читается – «тайни») есть внутренний генератор 1,6 МГц со схемой ФАПЧ, обеспечивающей входную частоту для внутренних таймеров 25,6 МГц, аналоговый компаратор и настраиваемая схема слежения за питающим напряжением, обеспечивающая перезапуск микроконтроллера при снижении напряжения питания ниже заданного уровня. На этом фоне 20-выводной микроконтроллер AT90S2313 выглядит тривиальной микросхемой. Однако, на базе AT90S2313 возможно построение достаточно широкого спектра изделий. В предлагаемой статье описывается устройство ввода аналоговых и цифровых данных в компьютер с возможностью их отображения на ЖК-индикаторе.

Микроконтроллер AT90S2313 удобно использовать в приложениях, где требуется передавать информацию по порту RS-232. Аппаратный UART, входящий в состав микроконтроллера, полностью реализует эту функцию, освобождая ресурсы процессора. При правильном выборе частоты можно обеспечить достаточно высокую скорость передачи данных.

Например, при использовании кварца с частотой 7,3728 МГц, скорость передачи данных может быть установлена в диапазоне от 2,4 до 460 Кбод.

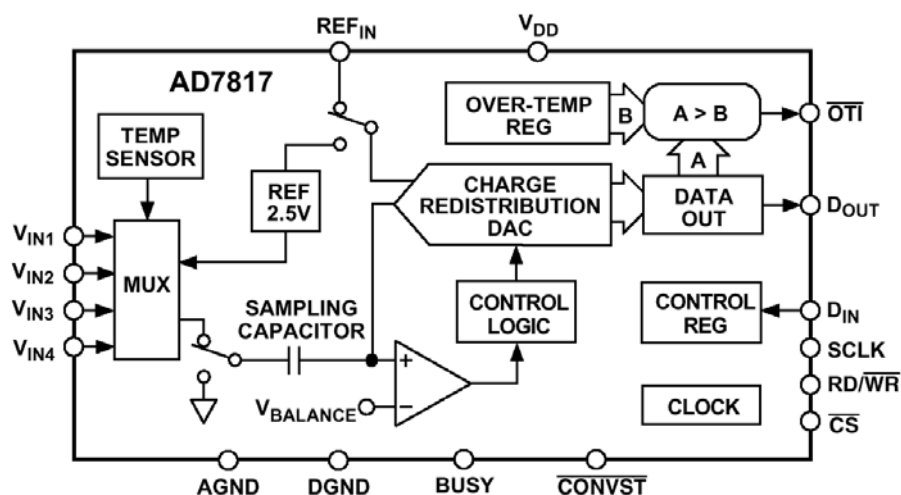
Описываемое ниже устройство выполняет функции сбора аналоговых и дискретных сигналов и отображает информацию на цифробуквенном жидкокристаллическом индикаторе. Устройство может работать автономно, а также передавать информацию в компьютер через порт RS-232. Питание устройства осуществляется от внешнего источника напряжением +5 Вольт. Потребляемый ток – не более 10 мА.

Состав устройства:

- 4-канальный 10-бит АЦП
- время преобразования – 9 мкс на канал
- двухстрочный ЖК индикатор
- порт RS-232 со скоростью передачи до 230 Кбод.

“Аналоговые” характеристики устройства определяются использованным АЦП – микросхемой AD7817. Эта микросхема представляет собой четырехканальный аналого-цифровой преобразователь со встроенным тактовым генератором и источником опорного напряжения 2,5 В (рис. 1).

Рисунок 1.

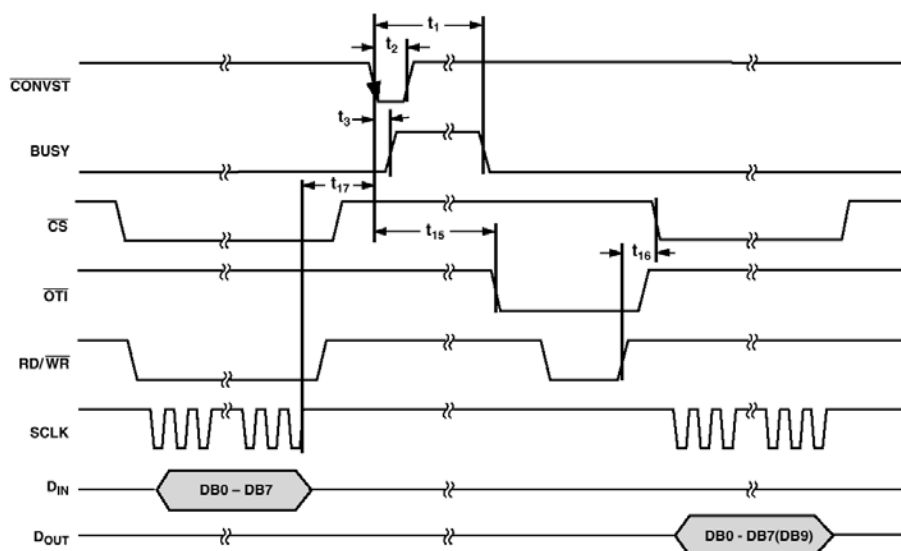


Для повышения термостабильности внутреннего источника опорного напряжения используется специальная схема термокомпенсации с встроенным термодатчиком. Выход этого термодатчика подключен к пятому каналу АЦП, таким образом, кроме четырех внешних аналоговых сигналов микросхема AD7817 может также измерять температуру кристалла. Точность измерения температуры составляет $\pm 1^\circ\text{C}$ в диапазоне температур $-40 \dots +85^\circ\text{C}$ и $\pm 2^\circ\text{C}$ в диапазоне температур $-55 \dots +125^\circ\text{C}$. Учитывая весьма незначительную среднюю собственную потребляемую мощность, составляющую 4 мкВт при периоде измерений 100 мс и 40 мкВт при периоде измерений 1 мс, можно считать, что измерения с достаточной степенью точности отражают температуру окружающего воздуха вблизи микросхемы AD7817. Из сервисных возможностей можно отметить также наличие специального регистра для записи порогового значения температуры, при превышении которого формируется сигнал на выводе микросхемы.

Интерфейс микросхемы совместим с интерфейсом SPI, то есть существуют выводы «вход данных» (Din), «выход данных» (Dout) и «синхроимпульсы» (SCLK). Вход RD/WR предназначен для управления направлением передачи данных, а по входу CS осуществляется выбор микросхемы. В приложениях, где используется только одна микросхема AD7817, вывод CS можно заземлить. Для уменьшения общего количества связей между АЦП и микроконтроллером допускается соединение выводов Din и Dout.

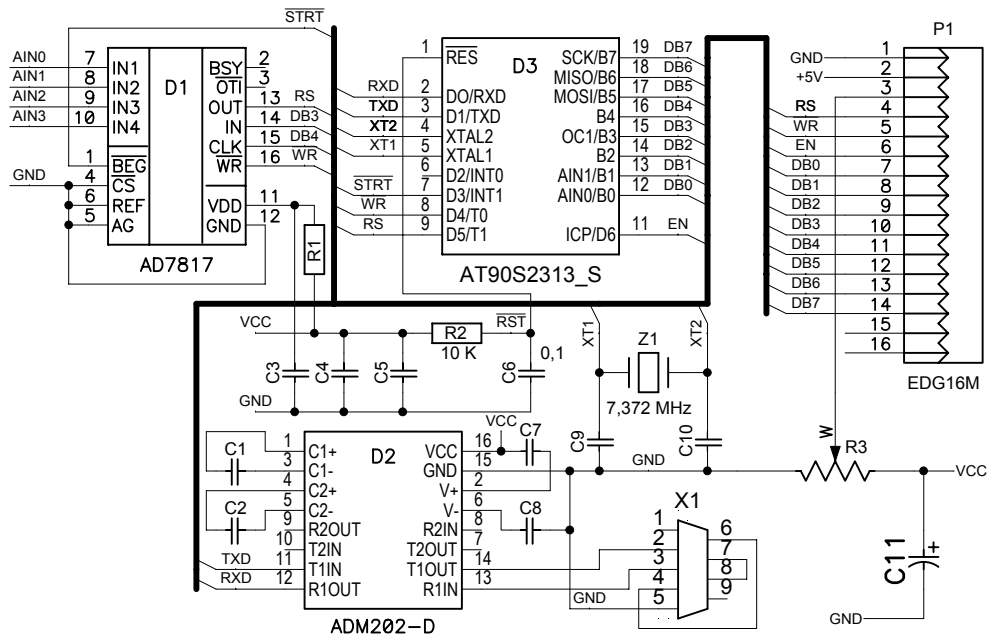
АЦП имеет два режима работы. Режим определяется уровнем сигнала CONVST. АЦП работает в режиме 1, если в момент окончания преобразования на выводе 1 CONVST присутствует высокий уровень («короткий» CONVST). Это стандартный режим работы. В приложениях с критичным значением потребляемой мощности используется режим 2, который задается низким уровнем CONVST в момент окончания преобразования («длинный» CONVST). В этом режиме АЦП после окончания преобразования переходит в энергосберегающий режим. Возврат в активное состояние происходит после подачи высокого уровня на CONVST. В описываемом устройстве микросхема AD7817 работает в режиме 1. Временная диаграмма работы AD7817 представлена на рис.2.

Рисунок 2.



В описываемом устройстве для упрощения программирования используется раздельное подключение выводов Din и Dout микросхемы AD7817 к микроконтроллеру. С ЖК индикатором микроконтроллер связан через восьмиразрядную шину данных, что также упрощает программирование интерфейса. Интерфейс RS-232 реализован по стандартной схеме на микросхеме ADM202. Здесь можно отметить, что при использовании интерфейсной микросхемы ADM3202 с расширенным питанием (от 3 до 6 Вольт) устройство будет работоспособно при пониженном питании, так как напряжение питания микроконтроллера AT90S2313 также может составлять от 2,7 до 6 Вольт.

Принципиальная схема устройства сбора и отображения данных приведена на рис. 3.



Вследствие ограниченного числа выводов у AT90S2313, некоторые из них имеют двойное назначение. Для работы с ЖК индикатором используется 8-битный интерфейс (весь порт В микроконтроллера), однако, так как данные передаются только в одном направлении, часть выводов порта В AT90S2313 управляет работой АЦП. На ЖК индикаторе отображается измеренное значение входного напряжения, пересчитанное в Вольты, номер канала измерения, а также температура окружающей среды. Без каких-либо модификаций схемы можно использовать индикаторы от однострочных на 8 знакомест до четырехстрочных на 40 знакомест. В отличие от ЖК индикатора, для общения с микросхемой АЦП используется двунаправленный обмен, потому что необходимо выбрать канал измерения и выдать импульс начала преобразования.

Во время отладки работы устройства была решена небольшая техническая проблема. Дело в том, что на начальном этапе устройство в произвольные моменты времени “самоперезапускалось”. В результате анализа проблемы выяснилось, что на вход RESET микроконтроллера действовала наводка, приводившая к его перезапуску. В соответствии с техническими условиями на микросхему AT90S2313, вывод RESET можно никуда не подключать. Однако, как выяснил эксперимент, это справедливо для идеального случая, когда цепи питания разведены на печатной плате отдельными слоями, и вблизи микроконтроллера нет источника импульсов. Так в жизни не бывает, поэтому были применены стандартные меры – между входом RESET и общим проводом установлен керамический конденсатор номиналом 0,1 мкФ, а на цепь питания “повешен” резистор 10 кОм. После этого никаких проблем со сбоями замечено не было, что лишний раз подтвердило истину: в цепи сброса надо использовать конденсатор (лучше керамический). Внешний вид окончательного устройства приведен на рис. 4.

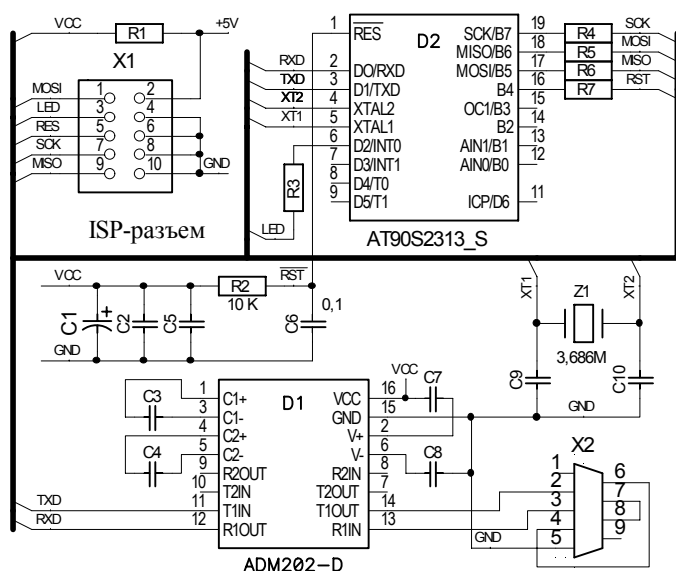
Рисунок 4.



Описанная схема оказалась настолько удачной, что была использована еще в двух приложениях. Когда потребовалось повысить точность измерений и количество входных каналов, пришлось всего лишь заменить микросхему AD7817 на AD7888, которая также выполнена в корпусе SOIC16, однако, имеет 12-битный 8-канальный АЦП с временем преобразования 8 мкс на канал. “Третья жизнь” схемы началась после августа 1999 г. Именно это время фирма ATMEL объявила о прекращении выпуска своего недорогого программирующего кабеля AT90ISP, которым многие разработчики пользовались для внутрисхемного программирования AVR-микроконтроллеров через SPI-порт непосредственно на плате. Решение фирмы ATMEL по сути правильное, так как этот кабель уже морально устарел – у него была слишком низкая скорость работы и маленький перечень поддерживаемых микросхем. Это объясняется использованием в AT90ISP контроллера AT90S1200, который имеет всего 1 Кбайт ПЗУ программ и не оснащен аппаратным последовательным портом. Проблема заключалась в том, что вместо упомянутого кабеля не было предложено никакого подобного устройства.

Таким образом, возник вакуум, и под напором заказчиков, которым нужен недорогой внутрисхемный программатор, за короткое время было создано устройство, схема которого приведена на рис. 5.

Рисунок 5.



Можно легко видеть, что эта схема представляет собой урезанную предыдущую, но с добавлением резисторов, защищающих микроконтроллер от неправильного подключения к программируемой плате. Увеличенная в два раза память программ и наличие аппаратного последовательного порта позволило создать качественно новое изделие. Аппаратная реализация интерфейса обмена с компьютером и тщательно настроенная временная диаграмма обмена по SPI-порту (на что, по правде говоря, было положено немало сил и времени), позволила увеличить скорость обмена по сравнению с изделием AT90ISP в восемь раз. Это существенно при работе с микросхемами, оснащенными большой памятью программ, например с ATmega103, имеющей ПЗУ объемом 128 кбайт.

Для программирования AVR-микроконтроллеров можно использовать разнообразные средства. Наиболее известен пакет AVR-Studio, бесплатно распространяемый фирмой ATMEL.

В состав пакета входит текстовый редактор, компилятор с ассемблера и симулятор. Симулятор выполнен с особенной любовью – можно работать с большим количеством разнообразных окон и настраивать интерфейс на свой вкус. Стоит отметить, что для работы с аппаратным внутрисхемным эмулятором для AVR-микроконтроллеров, ICE200, используется та же среда, что безусловно, удобно. При запуске программы AVR-Studio проверяет все COM-порты компьютера, и, в случае обнаружения на любом из них подключенный эмулятор, переходит в режим аппаратной отладки.

Для AVR-микроконтроллеров существует много компиляторов языков высокого уровня. Внутренняя архитектура AVR аппаратно оптимизирована для выполнения программ, написанных на языке Си. Одним из удобных пакетов является компилятор Си фирмы ImageCraft. Фирма предоставляет для ознакомления бесплатную 30-дневную демо-версию компилятора. Приятно, что в демо-версии нет ограничения на размер кода, поэтому можно создавать проект любого размера и для любого AVR-микроконтроллера, включая новинку фирмы ATMEL – микросхемы серии AT94, содержащие ядро AVR, работающее на частоте до 40 МГц. Демо-версию компилятора можно переписать отсюда: <http://atmel.argussoft.ru/soft.htm>.

В результате компиляции формируются два файла, - один, в формате COFF, для последующей отладки в AVR Studio, а второй, в формате HEX, для загрузки в программатор, причем запрограммировать микросхему можно прямо из среды ImageCraft.

Программа для устройства сбора и отображения данных, приведенная ниже, написана именно с использованием Си-компилятора фирмы ImageCraft.

```
#include <io2313.h>

typedef unsigned char byte;
typedef unsigned short word;
typedef unsigned long dword;

#pragma interrupt_handler timer_handler:5

void delay(dword ticks) { while(ticks--); }

// schematic description:
// =====
// pin  LCD  ADC
// pb0  |
// pb1  |
// pb2  d
// pb3  a    in
// pb4  t    clk
// pb5  a
// pb6  |
// pb7  |
// -----
// pd0  -- DEBUG --
// pd1
// pd2
// pd3      strt
// pd4  rw  rw
// pd5  rs  out
// pd6  e
// =====

//
// LCD interfacing
//

#define LCD_E (1 << 6)
#define LCD_RS (1 << 5)
#define LCD_RW (1 << 4)

#define lcd_set_e() (PORTD |= LCD_E)
#define lcd_set_rs() (PORTD |= LCD_RS)
#define lcd_set_rw() (PORTD |= LCD_RW)

#define lcd_clear_e() (PORTD &= ~LCD_E)
#define lcd_clear_rs() (PORTD &= ~LCD_RS)
#define lcd_clear_rw() (PORTD &= ~LCD_RW)

void lcd_pulse(void) // strobe
{
```

```
lcd_set_e();
delay(15);
lcd_clear_e();
delay(15);
}
```

```
void lcd_wait(void) // long delay
{
delay(2500);
}
```

```
void lcd_send(byte data)
{
PORTB = data;
lcd_pulse();
}
```

```
void clrscr(void) // clear display
{
delay(5);
lcd_clear_rs();
delay(5);
lcd_clear_rw();
delay(5);
lcd_send(0x01);
lcd_wait();
}
```

```
void initgraph(void) // initialize
{
DDRB = 0xFF;
DDRD |= (LCD_E | LCD_RS | LCD_RW);
lcd_clear_rs();
lcd_clear_rw();
lcd_wait();
lcd_send(0x3C);
lcd_wait();
lcd_send(0x3C);
lcd_wait();
lcd_send(0x3C);
lcd_wait();
lcd_send(0x06);
lcd_wait();
lcd_send(0x0C);
lcd_wait();
lcd_send(0x01);
lcd_wait();
}
```

```
void gotoz(byte z) // goto absolute address
{
lcd_clear_rs();
lcd_clear_rw();
lcd_send(z | 0x80);
}
```

```
#define gotoxy(x,y) gotoz(x|(y<<6))
```

```
void putchar(char c) // write char to display
```

```

{
lcd_clear_rw();
lcd_set_rs();
lcd_send(c);
}

void outtext(char* text) // output '\0'-padded string
{
byte i;
for(i = 0; text[i] && i < 16; i++)
putchar(text[i]);
}

//
// shared pins
//

#define STRT (1 << 3)
#define WR (1 << 4)
#define OUT (1 << 5)

#define IN (1 << 3)
#define CLK (1 << 4)

// don't forget to change shared pin(s) direction !!!

void switch2lcd(void)
{
DDRD = 0xFF;
}

void switch2adc(void)
{
DDRD = (byte)(~OUT); // catch warning
}

//
// ADC interfacing
//

void initadc(void)
{
DDRD |= ~OUT;
PORTD &= ~OUT;
}

void strobe0(void) // adc strobe down
{
delay(5);
PORTB &= ~CLK;
delay(5);
}

void strobe1(void) // adc strobe up
{
delay(5);
PORTB |= CLK;
delay(5);
}

```

```

void strobe(void) // adc strobe [--\__/--]
{
  strobe0();
  strobe1();
}

word read_temperature(void)
{
  word temperature = 0;
  byte i;

  switch2adc();

  // step 1 - send address, address 0 <=> temperature

  PORTB &= ~IN;

  delay(4);

  PORTD &= ~WR;
  delay(4);

  for(i = 0; i < 8; i++) // 8 bit address (0x00)
    strobe();

  delay(4);

  // step 2 - start conversion

  PORTD &= ~STRT;
  delay(4);
  PORTD |= STRT;
  delay(50);

  // step 3 - read temperature

  PORTD |= WR;
  delay(4);

  temperature = 0;
  for(i = 0; i < 8; i++)
  {
    strobe();
    temperature *= 2;
    if(PIND & OUT)
      temperature++;
  }

  if(temperature != 0xFF) // if valid, result in celsius
    temperature -= 103; // else 0xFF

  // done

  switch2lcd();

  return temperature;
}

```

```

//
// main()
//

void timer(void);

byte s = 0, m = 0, h = 12;

void main(void)
{
  DDRD |= 1;
  PORTD &= ~1;

  TCCR1A = 0x00;
  TCCR1B = 0x0C; // ck/256, clear on compare match

  OCR1H = 0x8C; // second
  OCR1L = 0xA0;

  TIMSK = (1<<6); // t/c 1 compare match int enabled

  read_temperature(); // first time may be incorrect - skip it

  switch2lcd();
  initgraph();
  clrscr();

  timer();

  asm("sei");
}

//
// timer (every second)
//

// convert integer to bcd to avoid divisions
void byte2chr(byte v, char* a, char* b)
{
  (*a) = (*b) = 0;
  (*b) = v;
  while(*b >= 10)
  {
    (*b) -= 10;
    (*a)++;
  }
  (*a) += '0';
  (*b) += '0';
}

// timer routine
void timer(void)
{
  char c1,c2,c3,c4,c5,c6,c7,c8; // to avoid screen flicker

  word temperature = read_temperature();

  s++;
  if(s == 60)

```

```

{
s = 0;
m++;
if(m == 60)
{
m = 0;
h++;
if(h == 13)
h = 1;
}
}

byte2chr(h,&c1,&c2);
byte2chr(m,&c3,&c4);
byte2chr(s,&c5,&c6);
byte2chr((byte)temperature,&c7,&c8);

gotoxy(0,0);
outtext("T=");

if(temperature == 0xFF) // no adc connected
{
putchar '[';
putchar 'N';
putchar 'C';
putchar ']';
}
else
{
putchar(c7);
putchar(c8);
putchar('Я');
putchar('C');
}

putchar(' ');

putchar(c1);
putchar(c2);
putchar(':');
putchar(c3);
putchar(c4);
putchar(':');
putchar(c5);
putchar(c6);
}

// interrupt - call timer func
void timer_handler(void)
{
timer();
}

```

Длина выходного hex-файла составляет 1280 байт.

Консультацию по применению AVR-микроконтроллеров можно получить у авторов статьи, написав письмо по адресу korolev@argussoft.ru или по телефону (095) 216-5929.