ADC & UART FUNCTIONS

Hi friend,

This document provides various functions for using ADC & UART on AVR controllers. I've tested these functions using ATmega32, with little changes it can be adapted to other atmega devices as well.

These document does not include the main() function. For testing, the user can make a small main function. Basically, the code was designed to read all the ADC channels of atmega32 and send the values of the voltages applied to them to the PC via serial port in a ASCII string fromat.

The main program should enable the ADC & UART and then call the function `ADC_transmitValue()`.

For further info and screen shot of PC, refer to my post: "Using M32_Card for Data Acquisition with on-board 8 channnel ADC" at www.dharmanitech.com

Code:

```
//*****************************************************************
//****   ADC ROUTINES ******************
//*****************************************************************
//Controller:   ATmega32 (16MHz Crystal)
//Compiler:     ICCAVR
//Author:       CC Dharmani, Chennai
//*****************************************************************

#include <iom32v.h>
#include <macros.h>

#define ADC_ENABLE                    ADCSRA |= (1<<ADEN)
#define ADC_DISABLE                   ADCSRA &= 0x7F
#define ADC_START_CONVERSION          ADCSRA |= (1<<ADSC)
#define TX_NEWLINE    {transmitByte(0x0d); transmitByte(0x0a);}

void ADC_init(void);
int ADC_read(void);
float ADC_calculateTemp(int);
unsigned char* updateTempDisplay(float);
unsigned char* temporary(void);
void ADC_transmitValue(void);


unsigned char receiveByte(void);
void transmitByte(unsigned char);
void transmitString_F(const unsigned char*);
void transmitString(unsigned char*);
unsigned char valueDisplay[]=":  .   volt";

float Vref = 5.00; // Reference voltage Vref of ADC
```

```
//*****************************************************************
// Initialize the ADC
//*****************************************************************
void ADC_init(void)
{
 ADCSRA = 0x00; //disable adc
 ADMUX = 0x40; //select adc input 0
 ADCSRA = 0x86;
}

//*******************************************************************
// Do a Analog to Digital Conversion
//*******************************************************************
int ADC_read(void)
{
    char i;
    int ADC_temp, ADCH_temp;
    int ADC_var = 0;

    ADC_ENABLE;

    ADC_START_CONVERSION;        //do a dummy readout first

    while(!(ADCSRA & 0x10));     // wait for conversion done, ADIF flag
                                 // active
    ADCSRA|=(1<<ADIF);

    for(i=0;i<8;i++)             //do the ADC conversion 8 times for
                                 //better accuracy
    {
       ADC_START_CONVERSION;
        while(!(ADCSRA & 0x10));    // wait for conversion done, ADIF
                                    //flag active
        ADCSRA|=(1<<ADIF);

        ADC_temp = ADCL;            // read out ADCL register
        ADCH_temp = ADCH;           // read out ADCH register
           ADC_temp +=(ADCH_temp << 8);
        ADC_var += ADC_temp;      // accumulate result (8 samples) for
                                  //later averaging
    }

    ADC_var = ADC_var >> 3;     // average the 8 samples

    ADC_DISABLE;

    return ADC_var;
}
```

```
//*********************************************************************
//To calculate Voltage
//*********************************************************************
 float ADC_calculateValue(int inputValue)
 {
    float actualValue;
    actualValue=(inputValue * Vref/1024.0);
                                  //calculates the voltage present
    return actualValue;
 }

//********************************************************************
//To update the valueDisplay string based on the latest voltage read
//********************************************************************
  unsigned char* updateDisplay(float actualValue)
 {

     int temp;
     unsigned char c;
     temp=(int)(actualValue*100.0);   //to include decimal point for
                                      //display

     if((actualValue*100.0 - temp) >= 0.5) temp=temp+1;

     valueDisplay[5] = ((unsigned char)(temp%10)) | 0x30;
     temp=temp/10;
     valueDisplay[4] = ((unsigned char)(temp%10)) | 0x30;
     temp=temp/10;
     valueDisplay[2] = ((unsigned char)(temp%10)) | 0x30;
     temp=temp/10;

     return valueDisplay;

}

//********************************************************************
//to transmit the voltage values at 8 ADC channels
//********************************************************************
void ADC_transmitValue(void)
{
  int value;
  float value1;
  unsigned char i;


  for(i=0; i<8; i++)
  {
    ADMUX &= 0xe0;
    ADMUX |= i;   //select channel
    value = ADC_read();
    value1 = ADC_calculateValue(value);

//following three functions are part of the UART routines, sending the
//voltage values to serial port. UART must be initialized before
//calling them

    transmitString_F("   Channel ");
    transmitByte(i | 0x30);
    transmitString(updateDisplay(value1));
    TX_NEWLINE;
  }
}

//****************************************************************
//******** FUNCTIONS FOR SERIAL COMMUNICATION USING UART *******
//****************************************************************

//****************************************************
//Function to initialize UART
//****************************************************
//UART0 initialize
// desired baud rate: 19200
// actual: baud rate:19231 (0.2%)(at 16MHz crystal)
// char size: 8 bit
// parity: Disabled

void uart0_init(void)
{
 UCSRB = 0x00; //disable while setting baud rate
 UCSRA = 0x00;
 UCSRC = BIT(URSEL) | 0x06;
 UBRRL = 0x33; //set baud rate lo
 UBRRH = 0x00; //set baud rate hi
 UCSRB = 0x98;
}

//***************************************************
//Function to receive a single byte
//**************************************************
unsigned char receiveByte( void )
{
      unsigned char data, status;

      while(!(UCSRA & (1<<RXC)));      // Wait for incomming data

      status = UCSRA;
      data = UDR;

      return(data);
}
```

```
//**************************************************
//Function to transmit a single byte
//**************************************************
void transmitByte( unsigned char data )
{
     while ( !(UCSRA & (1<<UDRE)) )
          ;                   //Wait for empty transmit buffer
     UDR = data;              //Start transmition
}


//**************************************************
//Function to transmit a string stored in Flash
//**************************************************
void transmitString_F(const unsigned char* string)
{
    while (*string)
     transmitByte(*string++);
}


//**************************************************
//Function to transmit a string from RAM
//**************************************************
void transmitString(unsigned char* string)
{
    while (*string)
     transmitByte(*string++);
}



//*******************  END  *******************
```