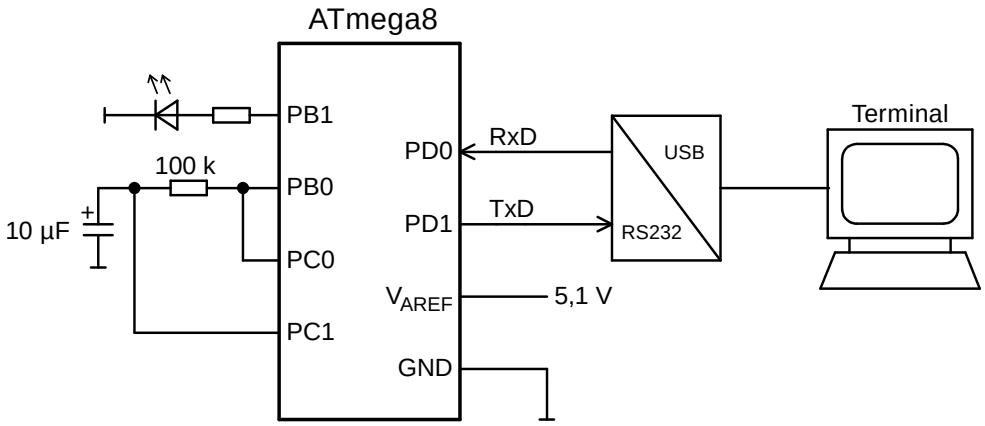
Ein Datenlogger

Mit einem Datenlogger oder Speicheroszilloskop können mit Hilfe von geeigneten Sensoren zeitlich veränderliche Größen wie Temperatur, Druck, Luftfeuchtigkeit,... erfasst, gespeichert und aufbereitet werden. Die Daten werden über einen definierten Zeitraum vom analogen Eingang des Mikrocontrollers erfasst, im Datenspeicher abgelegt und dann über die RS232-Schnittstelle zum PC gesendet und dort grafisch dargestellt. Die Samplerate bestimmt die Anzahl der Messungen pro Sekunde.

In diesem Beispiel soll die Auf- und Entladung eines Kondensators über einen Widerstand gemessen und grafisch am PC angezeigt werden. Das untenstehende zeigt den Aufbau des Systems. An PB0 des Mikrocontrollers wird zweimal ein Rechteckimpuls mit der Dauer von 1 Sekunde ausgegeben. Dadurch wird der Kondensator nach einer e-Funktion geladen. Die Spannung an PB0 wird über den analogen Eingang PC0 gemessen. Die Spannung am Kondensator wird über den Eingang PC1 gemessen. Die Dauer der Messung beträgt insgesamt 4 Sekunden. Die Samplerate beträgt 100 Messungen pro Sekunde. Solange die Messung läuft leuchtet eine LED an PB1.



Das Programm für den Mikrocontroller:

Im Programm werden die serielle Schnittstelle, der AD-Wandler und der Timer1 initialisiert. Nach Programmstart wird auf den Empfang eines Zeichens gewartet. Die eigentliche Messung erfolgt danach in einer Interrupt-Routine, welche alle 10ms aufgerufen wird. Dazu arbeitet der Timer1 im CTC-Modus.

Innerhalb der Interrupt-Routine werden sowohl die Spannung an PC0 als auch an PC1 gemessen. Die Werte werden in den Datenfeldern wert1 bzw. wert2 abgelegt. Nach 400 Messungen wird der Vorgang abgeschlossen und die Werte werden zum PC gesendet. Es werden also insgesamt 800 Bytes zum PC gesendet.

Nach jeweils 100 Messungen (dem entspricht 1 Sekunde) wird der Zustand an PB0 invertiert, woraus sich das Rechtecksignal ergibt.

```
/**********************************
Beispiel: Datenlogger
**********************************/
#include <avr/io.h>
#include <avr/interrupt.h>
volatile uint8_t wert1[400]; //Werte für AD-Werte PC0
volatile uint8_t wert2[400]; //Werte für AD-Werte PC1
volatile uint16_t count1=0;  //Anzahl der Messungen
volatile uint8_t count2=0;   //Zähler für Impulsdauer

void UARTInit()
{
      UBRRL=0x17; //Baudrate 9600Bd
      UCSRB=0x18; //RXEN=1; TXEN=1
      UCSRC=0x86; //Asynchroner Modus, keine Parität, 8 Datenbits, 1 Stoppbit
}

uint8_t UARTGetChar()
{
      while(!(UCSRA&(1<<RXC)));     //warte auf Empfang
      return(UDR);                  //empfangene Daten
}

void UARTPutChar(uint8_t val)
{
      while(!(UCSRA&(1<<UDRE)));    //warte auf freies Senderegister
      UDR=val;                      //sende Daten
}

void ADInit()
{
      ADMUX=(1<<ADLAR);             //linksb. Ausgabe
      ADCSRA|=(1<<ADEN|1<<ADPS2|1<<ADPS0);//ADFR=0, ADEN=1 und Teilerfaktor 32
}

uint8_t GetADWert(uint8_t port)
{
      ADMUX=(ADMUX&0x00)|(1<<ADLAR)|port; //linksb. Ausgabe, MUX0 bis MUX3=port
      ADCSRA|=1<<ADSC;                    //Wandelung starten
      while (ADCSRA & (1<<ADSC));         //warte bis ADSC = 0
      return(ADCH);                       //Analogwert zurückgeben
}

void Timer1Init()
{
      TCCR1B|=(1<<WGM12)|(1<<CS12);//CTC-Modus, interner Takt - F_CPU/256
      OCR1A=143;                   //Vergleichswert des Timers; 10 ms
}
```

```
int main()
{
	uint16_t i=0;
	DDRB=0x03;			//PB0 und PB1 zur Ausgabe
	UARTInit();
	ADInit();
	Timer1Init();
	sei();				//Interrupts generell zulassen
	while(1)
	{
		UARTGetChar();		//warte auf Zeichen
		count1=0;
		count2=0;
		TCNT1=0;
		PORTB|=(1<<PB0)|(1<<PB1);
		TIMSK|=(1<<OCIE1A);		//Timer1-Interrupt ermöglichen
		while(count1<400);
		TIMSK&=~(1<<OCIE1A);		//Timer1-Interrupt sperren
		PORTB&=~((1<<PB0)|(1<<PB1));
		for(i=0;i<400;i++)
		{
			UARTPutChar(wert1[i]);	//Werte von PC0 senden
		}
		for(i=0;i<400;i++)
		{
			UARTPutChar(wert2[i]);	//Werte von PC1 senden
		}
	}
	return(0);
}

ISR(TIMER1_COMPA_vect)
{
	wert1[count1]=GetADWert(0x00);
	wert2[count1]=GetADWert(0x01);
	count1++;
	count2++;
	if(count2%100==0)
	{
		PORTB^=(1<<PB0);
		count2=0;
	}
}
```

Das C#-Programm für den PC:

Für das PC-Programm erstellen Sie bitte eine Windows C#-Forms-Anwendung mit dem Namen Datenlogger. Übertragen Sie in die Form eine comboBox1 für die Auswahl des COM-Ports und eine comboBox2 für die Auswahl der Baudrate.

Mit einem Button `btnMessen` wird ein Zeichen zum Mikrocontroller gesendet und die Messung gestartet. Mit einem Button `btnBeenden` wird das Programm geschlossen.

Die grafische Ausgabe erfolgt nach dem Empfang von 800 Bytes in einer `pictureBox1` von der Größe 400 mal 265 Pixel. Jedes Pixel entspricht dabei einem Messwert.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Datenlogger
{
    public partial class Form1 : Form
    {
        private byte[] wert = new byte[800];

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            int i = 0;
            int[] baudrate = { 4800, 9600, 19200, 57600, 115200 };
            for (i = 0; i < 5; i++)
            {
                comboBox2.Items.Add(baudrate[i]);
            }
            for (i = 1; i < 10; i++)
            {
                serialPort1.PortName = "COM" + i.ToString();
                try
                {
                    serialPort1.Open();
                    comboBox1.Items.Add(serialPort1.PortName);
                    serialPort1.Close();
                }
                catch
                { }
            }
        }
```

```
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                serialPort1.PortName = Convert.ToString(comboBox1.SelectedItem);
                serialPort1.Open();
            }
            catch
            {
                MessageBox.Show(this, "Fehler!");
            }
        }

        private void serialPort1_DataReceived(object sender,
            System.IO.Ports.SerialDataReceivedEventArgs e)
        {
/**************End of File empfangen ausblenden*****************/
            if (e.EventType == System.IO.Ports.SerialData.Eof) return;
            int anzahl = serialPort1.BytesToRead;
            int i = 0;
            if (anzahl >= 800)
            {
                serialPort1.Read(wert, 0, 800);
                Point[] points1 = new Point[400];
                Point[] points2 = new Point[400];
                for (i = 0; i < 400; i++)
                {
                    points1[i].X = i;
                    points1[i].Y = 256 - wert[i];
                }

                for (i = 0; i < 400; i++)
                {
                    points2[i].X = i;
                    points2[i].Y = 256 - wert[i+400];
                }

                Graphics g = pictureBox1.CreateGraphics();
                g.Clear(Color.White);
                Pen Stift0 = new Pen(Color.Black,1);
                Pen Stift1 = new Pen(Color.Red, 2);
                Pen Stift2 = new Pen(Color.DarkBlue, 2);
                g.DrawLine(Stift0, 0, 256, 400, 256);
                g.DrawLines(Stift1, points1);
                g.DrawLines(Stift2, points2);
            }

        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            serialPort1.BaudRate = (int)comboBox2.SelectedItem;
        }
```

```
        private void btnBeenden_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            Application.Exit();
        }

        private void btnMessen_Click(object sender, EventArgs e)
        {
            serialPort1.DiscardInBuffer();
            serialPort1.Write("0");
        }
    }
}
```

Das folgende Bild zeigt das Ergebnis der Messung.