Pokročilé programovanie mikropočítačov v jazyku C. Diel č.2

Publikované: 15.11.2015, Kategória: Mikroprocesory

www.svetelektro.com

Skúmame ADCčko

Ahojte programátori. Analog to Digital Converter (ADC) patrí bezpochyby medzi najdôležitejšie periférie každého MCU. Zámerne píšem „MCU“ pretože existujú aj „MPU“, ktoré ADCčko pri svojej činnosti takmer vôbec nepotrebujú. Nechcem tu rozoberať načo nám ADCčko slúži, ale skôr sa zamerať na jeho vlastnosti a parametre z hľadiska dymaniky a presnosti, ktoré potrebujem poznať pred použitím vo finálnej aplikácii. Ako inak... použijem na to Atmegu8 s 10 bitovým ADCčkom. Hneď na úvod treba poznamenať , že je to prevodník s postupnou aproximáciu. Existuje mnoho iných typov AD prevodníkov. Najčastejšie sa v bežných MCU stretneme práve s prevodníkom s postupnou aproximáciou. MCU, ktoré sú určené na veľmi presné meranie, obsahujú takzvaný sigma-delta prevodník. Sigma-delta prevodník je všeobecne v porovnaní s prevodníkom s postupnou aproximáciou pomalší ale naopak je presnejší, dosahuje väčšie rozlíšenie a dosahuje vysokú linearitu v celom rozsahu merania. O AD prevodníkoch existuje mnoho článkov na webe, napr. tento.
Datasheet Atmegy8 obsahuje okrem hlavnej kapitoly o ADCčku aj trochu skrytú kapitolu alebo tabuľku s charakteristikou ADCčka. Tabuľka je na strane 241 datasheetu. Spomínaná tabuľka je priložená aj v tomto článku a sú v nej vyznačené údaje, ktoré ma zaujímajú najviac.

ADC Characteristics

Table 103. ADC Characteristics

Symbol	Parameter	Condition	Min(1)	Typ(1)	Max(1)	Units
	Resolution	Single Ended Conversion		10		Bits
	Absolute accuracy (including INL, DNL, Quantization Error, Gain, and Offset Error)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1.75		
		Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 1MHz		3		
	Integral Non-linearity (INL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		0.75		LSB
	Differential Non-linearity (DNL)	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		0.5		
	Gain Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		
	Offset Error	Single Ended Conversion V_{REF} = 4V, V_{CC} = 4V ADC clock = 200kHz		1		
	Conversion Time(4)	Free Running Conversion	13		260	µs
	Clock Frequency		50		1000	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3(2)		V_{CC} + 0.3(3)	V
V_{REF}	Reference Voltage		2.0		AV_{CC}	
V_{IN}	Input voltage		GND		V_{REF}	
	Input bandwidth			38.5		kHz
V_{INT}	Internal Voltage Reference		2.3	2.56	2.9	V
R_{REF}	Reference Input Resistance			32		kΩ
R_{AIN}	Analog Input Resistance		55	100		MΩ

Notes:
1. Values are guidelines only
2. Minimum for AV_{CC} is 2.7V
3. Maximum for AV_{CC} is 5.5V
4. Maximum conversion time is 1/50kHz × 25 = 0.5ms

Tabuľka s charakteristikami ADCčka

Prvé čo ma zaujíma je Resolution teda rozlíšenie. ADCčko na Atmege8 má 10 bitové rozlíšenie.Väčšina „modernejších“ MCU má ADCčko s nastaviteľným alebo meniteľným rozlíšením, napr. 10, 12, 14 ,16 a podobne. 10 bitové rozlíšenie

v jednoduchosti znamená, že AD prevodník je schopný rozlíšiť 2^10 (1024) úrovní, pričom jedna úroveň je U_{ref}/2^10 (pre Atmegu 8 je to 5V/1024 = 4.88mV). Táto jedna úroveň sa označuje aj ako LSB (Least significant bit).
Druhá vec čo ma zaujíma je Absolute Accuracy alebo aj presnosť ADCčka. Tu si môžeme všimnúť, že chyba ADCčka sa zvyšuje s narastajúcimi frekvenciou hodín ADCčka. 3 LSB pri maximálnej frekvencii hodín je, myslím si, veľmi slušná presnosť aj pri 10 bitovom rozlíšení. Datasheet Atmegy8 na stranách 196 až 198 stručne popisuje chyby ADCčka ako je INL, DNL, Offset a Gain Error.
Ďalšia dôležitá vec je Conversion Time teda doba trvania jedného prevodu vo Free running mode. Je to mód kedy ADCčko v momente keď dokončí jeden prevod začne hneď druhý prevod. ADCčko Atmegy 8 dokáže jeden prevod vykonať za 13 us, ak je živené maximálnou frekvenciou hodín, teda podľa datasheetu 1MHz. Čím bude hodinový signál (frekvencia hodín) ADCčka nižšie tým sa doba prevodu predlžuje. S najkratšou dĺžkou prevodu suvisí aj posledný dôležitý riadok z tabuľky a to je Input Bandwidth. Je to vlastne maximálna frekvencia signálu, ktorú dokážem týmto ADCčkom na tomto MCU odmerať s tým, že výsledok mi bude dávať aspoň aký taký zmysel. Ako sa k tejto hodnote dopracovali? Pre tých, čo poznajú Schannon-Kotelnikov teorém je to viac menej jasné. Ale v podstate veľmi jednoducho. Zoberem si najkratšiu dobu prevodu teda 0,000013s dam na mínus prvú (dostanem frekvenciu) a podelím ešte dvomi. Preco delím dvoma? Presne o tom hovorí Shannon-Kotelnikov teorém. V skratke, **frekvencia vzorkovania signálu musí byť minimálne dvakrát väčšia ako je frekvencia meraného signálu.** Viac sa dočítate, keď zadáte Shannon – Kotelnik teorem do googlu.
Dobre, nechajme už teóriu teóriou a podme si niečo aj reálne vyskúšať. Hneď na úvod som vytvoril tabuľku frekvencie hodín pre ADCčko aj s dobou trvania prevodu pre jednotlivé hodiny ADCčka. Vstupné hodiny pre ADCčko idú najskôr cez deličku. Najmenšia delička je 2 a najväčšia 128. Delička hodín sa nastavuje v registri ADCSRA pomocou spopných troch bitov ADPS0, ADPS1 a ADPS2. Dobu trvania prevodu som meral veľmi podobne ako aj časy v predchádzajúcom článku. ADCčko som si spustil do free running modu a v prerušení od skončenia prevodu som nastavil ľubovolný pin do jednotky, prečítal som výsledok a pin som hodil naspät do nuly. Oscilátor atmegy som mal nastavený na 8 Mhz a postupne som menil hodnoty preddeličky hodín pre ADCčko. Meral som čas medzi dvoma prerušeniami od ADCčka.

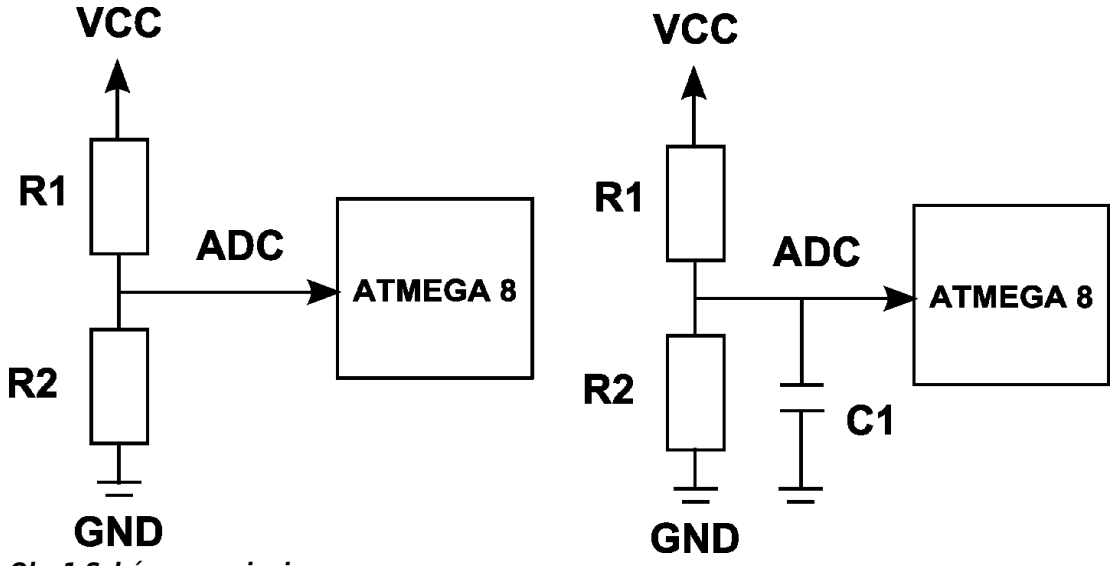
Ukážka kódu :

```
#include <avr/io.h>
#include <avr/interrupt.h>

#ifndef F_CPU
#define F_CPU 8000000
#endif
//globalna premenna adc
volatile unsigned int adc;

// prerusenie skoncenie AD prevodu
ISR(ADC_vect){
   PORTD |= (1 << PD2); // PD0 do jednotky
   adc = ADC; // zapis vysledok prevodu
  PORTD &= ~(1 << PD2); // PD0 do nuly

}

int main(){
         OSCCAL= 0xBE;

       DDRD = 0x04;         // PD2 ako vystupný pin ostatné ako vstupne
             PORTD= 0x00;         // PD2 do nuly

          ADMUX |=   (1 << MUX0); //pin na ktorom meriam

   // zapnutie AD prevodnika, Free runing, preddelicka 128 pri frek. hodin 8MHz
   ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADFR) | (1 << ADIE) | (1 <<ADPS2) | (1 <<ADPS1)  | (1
<<ADPS0);


   sei();  //povol globalne prerusenia

   while(1){ }

}
```

Tab.1 Tabuľka nameraných a dopočítaných hodnôt

Frekvencia MCU	**8MHz**			**4MHz**	**1MHz**
Preddelička ADC	**Frekvencia hodín pre ADC [kHz]**	**Doba prevodu [us]**	**Max frekvencia meraneho signalu [kHz]**	**Frekvencia hodín pre ADC [kHz]**	**Frekvencia hodín pre ADC [kHz]**
2	4000	4.4	113.64	2000	500
4	2000	6	83.33	1000	250
8	1000	13	38.46	500	125
16	500	26	19.23	250	62.5
32	250	52	9.62	125	31.25
64	125	106	4.72	62.5	15.625
128	62.5	212	2.36	31.25	7.8125

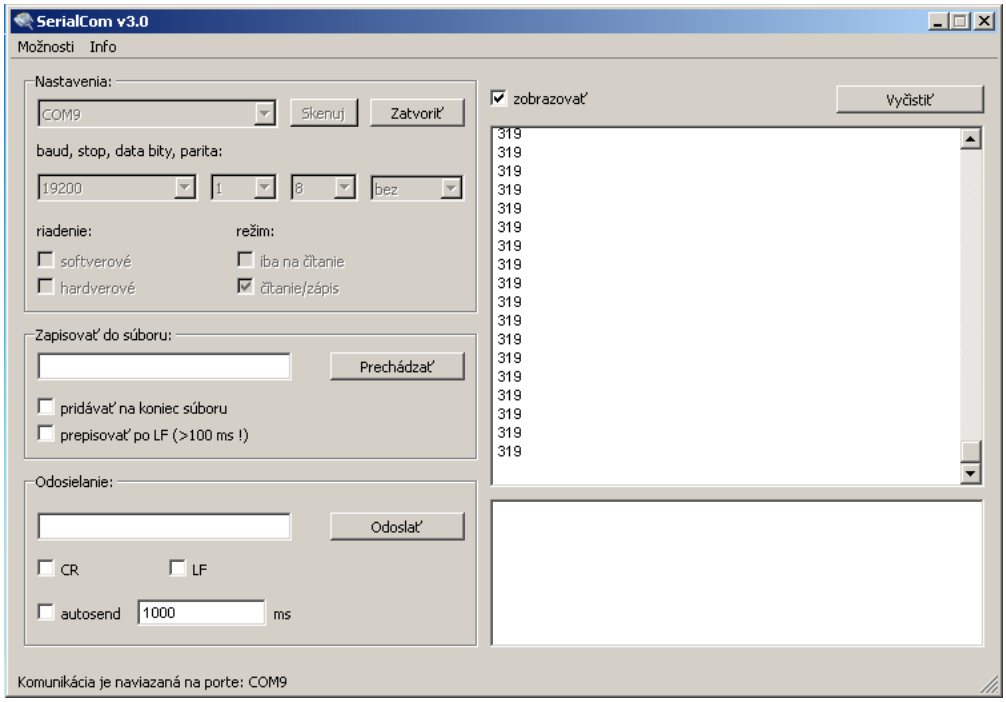
Meranie som robil len pre frekvenciu MCU8 MHz. Z merania je vidieť, že doba prevodu je lineárne závislá od frekvencie hodín ADCčka. Preto som pre 4MHz a 1 MHz už meranie ani nerobil pretože očakávam rovnakú závislosť a väčšina nastavení sa prelína. Čo ma prekvapilo je to, že som mohol nastaviť hodiny pre ADCčko aj vyššie ako 1 MHz. 1Mhz je uvedený v datasheete ako maximálna hodnota pre hodiny ADCčka. Preto som sa v druhom príklade zameral na prenosť ADCčka v závislosti od nastavenia frekvencie hodín ADCčka.
V druhom príklade opäť mením frekvenciu hodín ADCčka pomocou preddeličky. Po inicializácii ADCčka spustím prevod a uložím si prvých 256 výsledkov z meraní do poľa. Potom pomocou UARTU a ľubovolného softwaru pre prácu so sériovou linkou si celé pole vypíšem na obrazovku PC. Dáta z obrazovky jednoducho prekopírujem do excelu a analyzujem. Ja som použil Serial Com v3.0.0 (obr.2) od Luboša (http://lubosweb.php5.sk/serial-com-v3-0/). Knižnicu pre UART komunikáciu som použil od zawina, ktorú zverejnil vo svojom článku (http://svetelektro.com/clanky/programujeme-avr-v-jazyku-c-8-cast-511.html).
Meral som napätie na napäťovom delici (R1 a R2). Najskôr bez filtračného kondenzátora C1 a potom aj s filtračným kondezátorom. (schéma zapojenia na obr.1) Výsledky sú zobrazené na grafoch 1 a 2. Meral som jednosmerné napätie preto sa mi C1 zišiel. Existujú však prípady, kedy filtračný C1 nie je vhodné použiť. Problematika AD prevodníkov je veľmi rozsiahla a líší sa projekt od projektu.



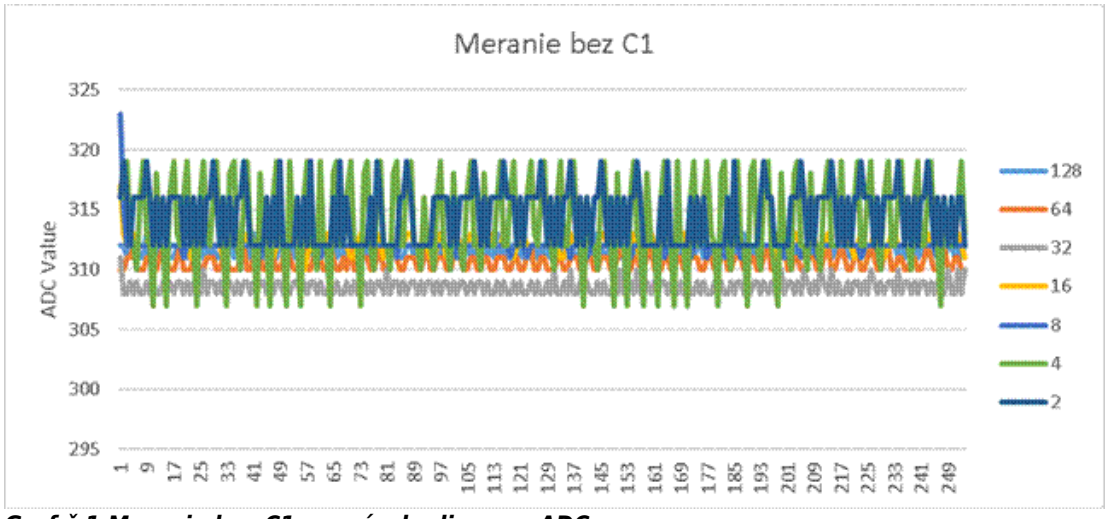
Obr.1 Schéma zapojenia

Ukážka kódu:

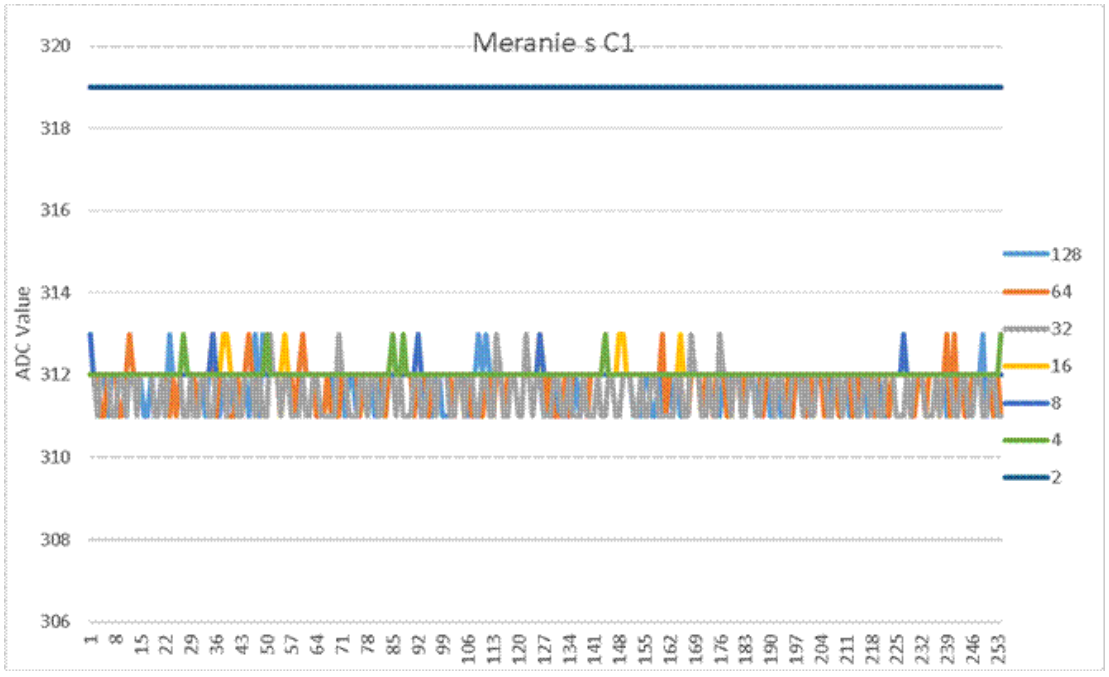
```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include "uart.h"

#ifndef F_CPU
#define F_CPU 8000000
#endif
//globalna premenna adc
volatile unsigned int adc;
volatile unsigned int result[256];
volatile char counter = 0;

// prerušenie na akciu skončenia AD prevodu
ISR(ADC_vect){
  result[counter]=ADC; // zapis vysledok prevodu
  counter++;
}

int main(){

    char text[32];
        int i;
        OSCCAL= 0xBE;
   ADMUX |=   (1 << MUX0); //referencia 5V, nastavenie pinu na ktorom meriam.

    // zapnutie AD prevodnika, Free runing, preddelicka 128 pri frek. hodin 8MHz
    ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADFR) | (1 << ADIE) | (1 <<ADPS2) | (1 <<ADPS1)  | (1
<<ADPS0);

    // inicializacia uart na rychlost 19200bd
    uart_init(19200);

    sei();  //povol globalne prerusenia

    while(1){
        if (counter == 255)
    {
         ADCSRA = 0x00;  //vypnutie ADCcka
         // vypis vysledkov na obrazovku PC
              for (i=0;i<255;i++)
         {
         sprintf(text,"%dn",result[i]);
         uart_puts(text);
         }
         counter=0;
    }
   }
}
```



Obr.2 Serial Com 3.0, software pre prácu so sériovou linkou od Luboša



Graf č.1 Meranie bez C1, mením hodiny pre ADC.



Graf č.2 Meranie s C1 , menším hodiny pre ADC

Záver si môže spraviť každý. Ja sa ho pokúsim zhrnúť do nasledovných bodov:

- Čím má ADCčko nižšiu frekvenciu hodín, tým meria alebo malo by merať presnejšie ale zároveň aj dlhšie.
- ADCčko na Atmege 8 dokáže merať aj s frekvenciou hodín vyššou ako je datasheetovych 1MHz ale presnosť klesá, aj keď nie nejako rapídne. Podľa mňa je to stále veľmi slušná presnosť.
- Presnosť merania závisí aj od návrhu hardweru. Meranie s C1 (graf č.2) bolo presnejšie pre všetky hodnoty clockov v porovnaní s meraním bez C1 (graf č.1).
- **Výber vhodného ADCčka a aj MCU treba vždy správne uvážiť. Treba brať dôraz na to čo idem merať , aký signál chcem merať, akú presnosť požadujem , ako často potrebujem merať , či potrebujem aj niečo s výsledkom ďalej robiť a podobne.**

Použitý hardware:

- R1 = 680kΩ, R2 = 300kΩ, C1 = 0,1uF a pár káblíkov
- Atmega 8 Development board.
- USB to RS232 prevodník
- AVR STK500 ako programátor

Toľko tento článok. Chcem na záver len pripomenúť, že somv tomto článku pracoval s AD prevodníkom s postupnou aproximáciou. Niektoré závery nemusia pre iné typy prevodníkov platiť. V budúcom článku sa trochu viac pohráme s úpravou a spracovaním nameraného signálu. Skúsime si nameraný signál softwarovo filtrovať.