RĪGAS TEHNISKĀ UNIVERSITĀTE

DATORZINĀTNES UN INFORMĀCIJAS TEHNOLOĢIJAS FAKULTĀTE

DATORVADĪBAS, AUTOMĀTIKAS UN DATORTEHNIKAS INSTITŪTS

Datoru tīklu un sistēmas tehnoloģijas katedra

Mikroprocesoru tehnika

4. laboratorijas darbs

Izpildīja: Eduards Šarņeckis
Grupa: III RDB F02
Apl. numurs: 101RDB121

2012./2013. māc. gads

Saturs

Uzdevums.....5
Programmas pirmteksts.....6
Programmas pirmteksta algoritma blokshēma.....9
Pirmkoda darbības pārbaude.....10
ATmega128 analogsciparu pārveidotājs.....11
Secinājumi.....14

Uzdevums

1. Izmainīt piedāvāto 8 bitu ADC pārveidošanas kodu, lai būtu iespēja strādāt ADC 10 bitu režīmā.
2. Izmainīt mērāmo signālu uz sin().
3. Pierādīt koda pareizo darbību.

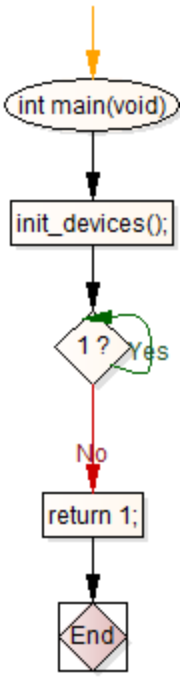
Programmas pirmteksts

```
/********************* Biblioteeku ieklaushana ******************/
#include <avr/io.h>
#include <avr/interrupt.h>
/****************************************************************/

/************** Papilduzdevums - ADC partraukums ****************/
ISR(ADC_vect)
{
 uint8_t i; //8 bitu integer mainigais
 uint16_t tenbits;  //16 bitu integer mainigais
 i = ADCL;  //pieshkiram ADCL, lai tam varetu pieklut
 tenbits = (ADCH << 2)|(i >> 6); //veidojam pilnus 10 bitus

 if(tenbits > 512)
 {
  PORTD |= 0x01;
 }
 else
 {
  PORTD &= 0x00;
 }
}
/****************************************************************/

/************************ Inicializacija ************************/
void init_devices(void)
{
 DDRD = 0xFF; //visas porta D linijas uz IZvadi

 ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Iestada ADC prikshdalitaju uz
128 – 125KHz sample rate @ 16MHz

 ADMUX |= (1 << REFS0); // Set ADC reference to AVCC
 ADMUX |= (1 << ADLAR); // Pa kreisi nobidita ADC vertiba
 ADMUX |= (1 << MUX1);
 ADMUX |= (1 << MUX0);

 ADCSRA |= (1 << ADFR); // Set ADC to Free-Running Mode
 ADCSRA |= (1 << ADEN); // Iesledz ADC
 ADCSRA |= (1 << ADIE); // Atlauj ADC partraukumus

 sei(); // Atlauj globalos partraukumus

 ADCSRA |= (1 << ADSC); // Start A2D Conversions

}
/****************************************************************/
```

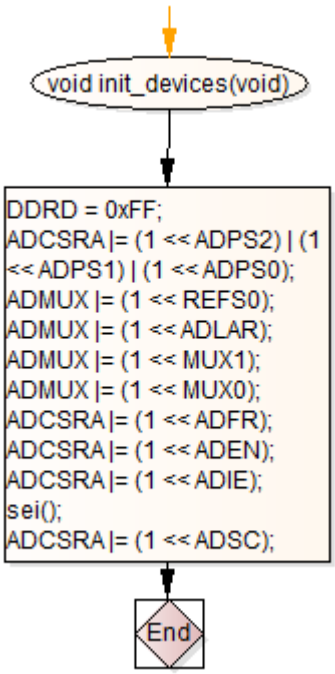
```
/******************** Main funkcija ****************************/
int main(void)
{
  init_devices(); //Inicializejam kontrolleri un ADC

  while(1) //Muzigais cikls, lai programa nekad nebeigtos
  {
  }

  return 1; //Funkcija main atgriez vertibu 1
}
/***************************************************************/
```

Programmas pirmteksta algoritma blokshēma

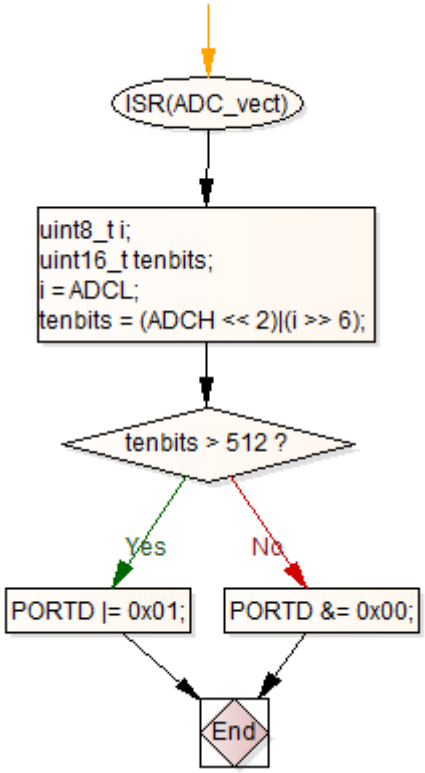
Funkcija main()



Funkcija init_devices()

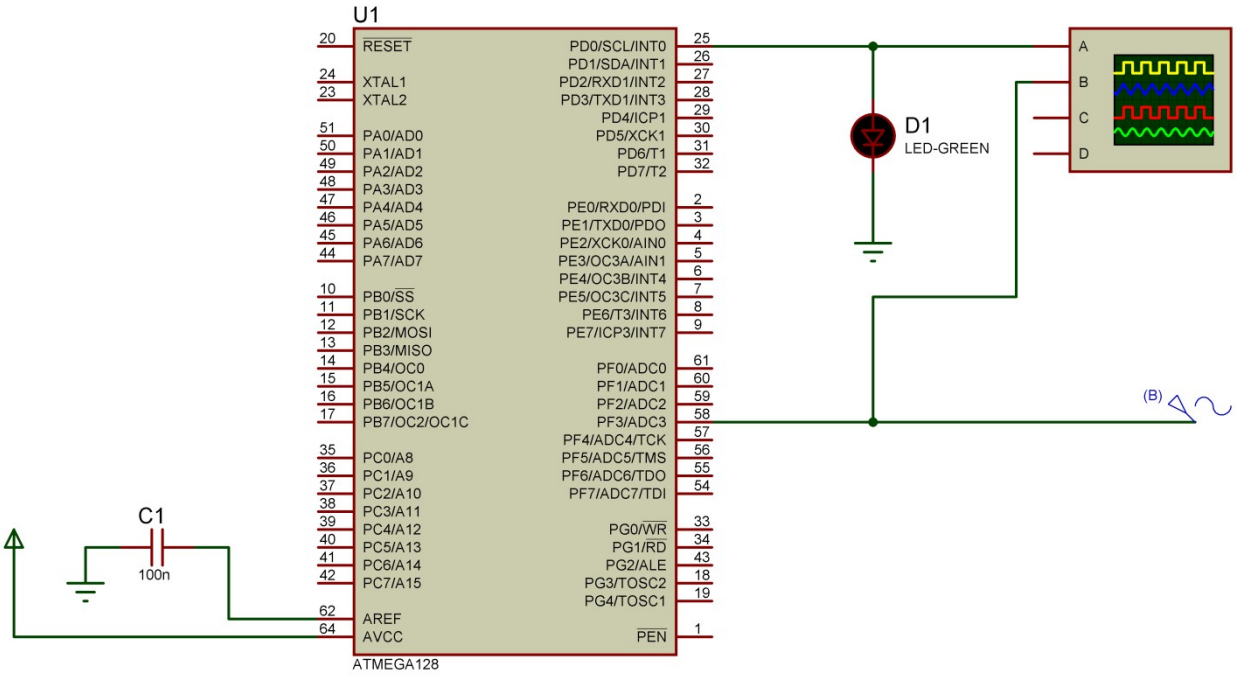


Pārtraukuma funkcija ISR(ADC_vect)



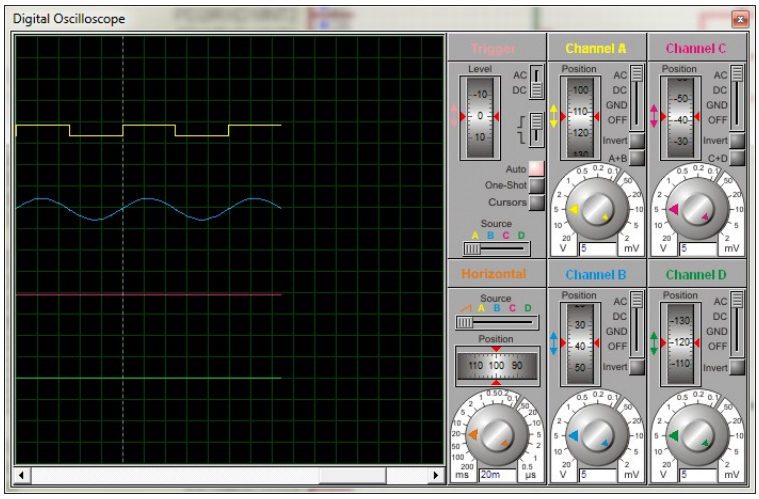
Pirmkoda darbības pārbaude

Lai pārbaudītu programmas darbību, tika izmantota lekcijas slaidos piedāvātā ATmega128 shēma Proteus ISIS vidē.



1. attēls ATmega128 shēma koda un ADC darbības testēšanai

Pēc koda modificēšanas osciloskopa izvade nav mainījusies, kas vizuāli pierāda modificētā koda pareizo darbību.



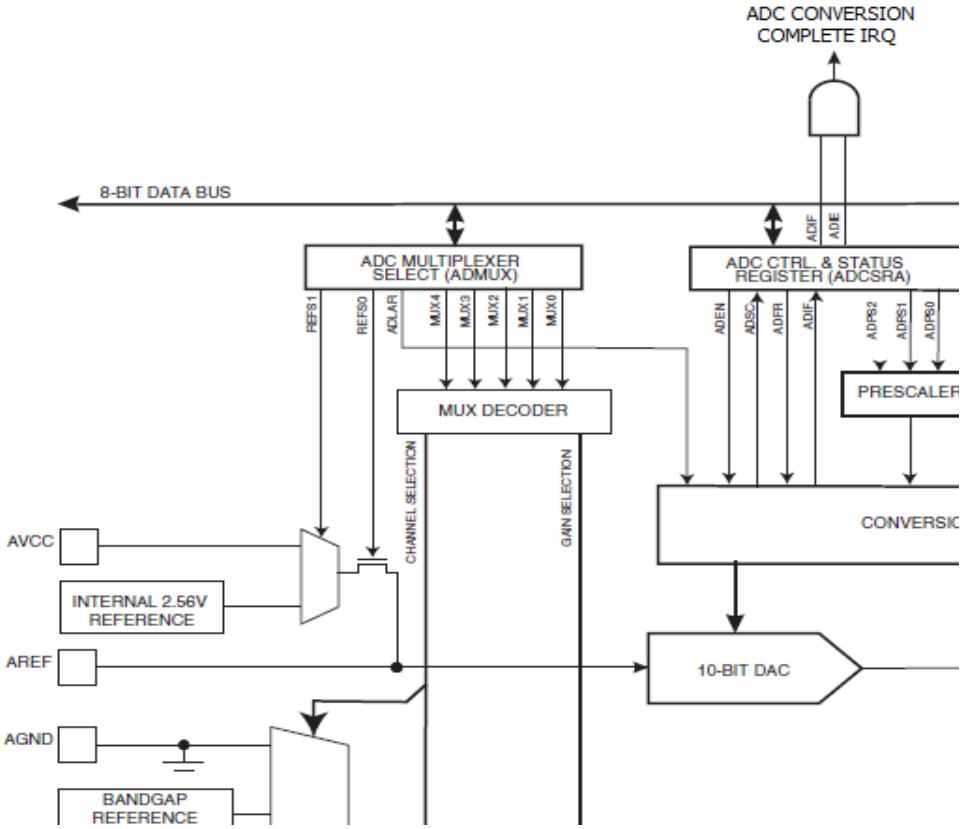
2. attēls Osciloskopa izvadīta bilde

ATmega128 analogsciparu pārveidotājs

ATmega 128 tiek izmantots 10 bitu secīga tuvinājuma ADC. ADC ir pievienots multipleksoram ar 8 kanāliem. Tas nozīmē, ka ar šādu ADC var mērīt 8 dažādus analogos signālus. Trūkums ir tāds, ka vienā laika momentā ir iespējams pārveidot tikai vienas ieejas analogo signāli. Praktiski tas nozīmē, kas vispirms ir jāizvēlas kanāls, kura analogo signālu vēlas mērīt, jāpārveido šī signāls un tad ir jāpārslēdz cita ACP ieeja ar multipleksoru. Secīga tuvinājuma ADC satur Sample and Hold shēmu, kas nodrošina, ka ADC ieejas spriegums tiek turēts nemainīgā līmenī, kamēr notiek pārveidošana. ADC ir iespējami divi barošanas avoti viens ārējs, kuru pieslēdz pie AVCC, un vien iekšējs, kurš parasti ir 2.56V un tiek nodrošināts no shēmas. To, kuru izmantot ir iepriekš, jāizvēlas.

Īsumā var uzskaitīt galvenās ATmega128 ADC īpašības:
· 10 bitu izšķirtspēja
· 0.5 LSB integrālā nelinearitātes kļūda
· ±2 LSB absolūtā precizitāte
· 13-260 µs pārveidošanas laiks
· 8 multipleksēti vienas ieejas kanāli
· Var izvēlēties rezultāta izkārtošanu – izlīdzinājums pa kreisi vai labi
· 0-VCC ADC ieejas sprieguma diapazons
· Izvēles 2.56V ADC atbalsta spriegums
· Nepārtrauktas pārveidošanas vai vienreizējas pārveidošanas režīmi
· Pārtraukums uz ADC pārveidošanas pabeigšanu
· Miega režīma trokšņu slāpētājs

ADC darbības iestādīšanai izmanto ADCSRA (ADC statusa un vadības reģistrs) un ADMUX (ADC multipleksēšanas izvēles reģistrs. Lai ieslēgtu ADC, jāuzstāda ADEN bits šajā reģistrā. Ja ADEN nav uzstādīt mikrokontrolleris nepatērē enerģiju, tāpēc ieejot miega režīmā to ir vēlams atslēgt. Pārveidošanas rezultāts tiek glabāts ADC datu reģistros – ADCH un ADCL. Pēc noklusējuma rezultāts ir izlīdzināts pa labi, bet to var izlīdzināt arī pa kreisi. ADMUX reģistrā attiecīgi izmainot bitu ADLAR. Ja rezultāts ir izlīdzināts pa kreisi un nav nepieciešama vairāk kā 8 bitu precizitāte, tad var nolasīt tikai ADCH reģistru. Pretējā gadījumā vispirms jānolasa ADCL un tikai tad ADCH, lai nodrošinātu, ka rezultāts pieder vienai un tai pašai pārveidošanai. Nolasot ADCH reģistru tiek atjaunota ADC pieeja rezultāta reģistram un tas var ierakstīt jaunu rezultātu.



ADLAR = 0:

Bit	15	14
	–	–
	ADC7	**ADC6**
	7	6
Read/Write	R	R
	R	R
Initial Value	0	0
	0	0

ADC ir savs pārtraukums, kurš tiek izsaukts tad, kad tiek pabeigta pārveidošana. Lai aprakstītu ADC pārtraukuma apstrādes funkciju:

```
ISR(ADC_vect)
{
  //Kods
}
```

Nepārtrauktas atjaunošanas režīmā ADC nepārtraukti ciparo signālu un atjauno datu reģistrus. To izvēlas, uzstādot ADFR bitu ADCSRA reģistrā. Lai uzsāktu pirmo pārveidošanu, ir jāuzstāda ADSC bits šajā pašā reģistrā. ADC tiek nodrošināts arī ar pirms dalītāju, kas nodrošina ADC ar taktsignālu. To uzstāda ar ADPS bitiem ADCSRA reģistrā. To izmanto, lai palielinātu vai samazinātu rezultāta atšķirību no oriģinālā ieejas signāla. Lai pārveidotu signālu ir nepieciešami 13 ADC takts impulsi.

Multipleksora reģistrs – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

3. attēls ADMUX reģistrs

7. un 6. bits – REFS1:0 - atbalsta sprieguma izvēles biti

Šie biti ir ADC atbildīgi par atbalsta sprieguma avota izvēli. Ja ieejošais ADC signāls ir ārpus šīm robežām, tad tas tiek pielīdzināts vai nu maksimālajam (ja spriegums lielāks par atbalsta spriegums) vai minimālajam = 0 (pretējā gadījumā).

REFS1	REFS0	Sprieguma avota izvēle
0	0	AREF, iekšējas Vref izslēgts
0	1	AVCC ar ārējo kondensatoru pie AREF pina
1	0	Rezervēts
1	1	Iekšējais 2.56V sprieguma atbalst ar ārējo kondensatoru pie AREF pina

5. bits – ADLAR – ADC rezultāts izlīdzināts pa kreisi

Šis bits atbild par to, lai izlīdzinātu ADC rezultātu ADCH un ADCL reģistros pa kreisi (vienāds ar 1) vai pa labi (vienāds ar 0).

4.-0. bits – MUX4:0 – ieejas kanāla izvēle

Ierakstot 2-0 bitā var izvēlēties kanālu, bet 4-3 var izvēlēties vai šo signālu ir nepieciešams pastiprināt.

Vadības un statusa reģistrs – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

4. attēls ADCSRA reģistrs

7. bits – ADEN – ADC atļaušana

Uzstādot šo bitu, tiek iedarbināts ADC, nodzēšot bitu, tas tiek izslēgts.

6. bits – ADSC – ADC pārveidošanas uzsākšana

Vienas pārveidošanas režīmā pirms katras pārveidošanas uzsākšanas ir jāuzstāda šis bits loģiskajā 1. Nepārtrauktas pārveidošanas režīmā to nepieciešams veikt tikai vienu reizi.

5. bits – ADFR – nepārtrauktas pārveidošanas režīma izvēle

Uzstādot šo bitu, tiek ieslēgts nepārtrauktas pārveidošanas režīms. Šajā režīmā nepārtraukti tiks atjaunoti datu rezultātu reģistri.

4. bits – ADIF – ADC pārtraukuma karodziņš

Šis bits tiek automātiski uzstādīts tad, kad tiek pabeigta pārveidošana un tiek atjaunoti datu reģistri. Ja ir atļaut pārtraukums tad tiek izsaukta pārtraukuma apstrādes funkcija. Tas tiek nodzēsts, kad pārtraukuma apstrādes funkcija tiek pabeigta. Alternatīvi to var nodzēst, ierakstot loģisko 1 karodziņā.

3. bits – ADIE – ADC pārtraukuma atļaušana

Uzstādot šo bitu, tiek atļauta ADC pārtraukuma apstrādes funkcijas izpildīšana.

2.-0. bits – ADPS2:0 – ADC pirms dalītāja izvēles biti

Ar šiem bitiem tiek uzstādīta ADC pirms dalītāja vērtība. Par pamatu tiek ņemta takts frekvence no XTAL un dalīta ar attiecīgo dalītāju.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Secinājumi

Ceturtā laboratorijas darba gaitā es iepazinos ar mikrokontrollera ATmega128 analogciparu pārveidotāju (ADC).

Šis laboratorijas darbs atšķiras no iepriekšējiem tajā ziņā, ka izpildot šo darbu mums, nebija tiešās saskarsmes ar ATmega128 mikrokontrolleri. Tomēr tas man problēmas nesagādāja, jo lekcijas slaidos bija dota gatava ATmega128 shēma, kuru varēja izmantot, lai izpildītu laboratorijas darbu.

Koda modificēšana arī nesagādāja problēmas. Vajadzēja tikai rūpīgi izlasīt teorētisko pamatojumu un atrast informāciju internetā par to, ka var iegūt pilnus 10 bitus no ADC un kā izpildīt papilduzdevumu - pareizi realizēt ADC pārtraukumu. Ar pārtraukumu realizāciju mēs jau esam saskārušies iepriekšēja laboratorijas darba ietvaros/

Uzskatu, ka laboratorijas darbs ir veiksmīgi izpildīts, jo modificētais kods strādā atbilstoši laboratorijas darba uzdevumam un tika iegūtas jaunas zināšanas par ATmega128 analogciparu pārveidotāju.