# ADC

| Status | Done |
| --- | --- |

## Šta je ADC?

**Analogno Digitalni Konvertor** - je elektronski sklop koji prevodi - elektricni napon u binarni broj.

## Naponska rezolucija

Rezolucija izrazena u broju bitova.

**Relativna rezolucija** zavisi od opsega mernog napona
Broj bitova je povezan sa kvalitetom ADC, sto je
**veci broj bitova, ADC je bolji.**

## Vremenska rezolucija

Vremenska rezolucija je povezana sa **brzinom rada ADC, sto je vreme konverzije krace to je ADC bolji.**

## ADC kanali

ADC kanali jesu broj nazavisnih merenja.
ADC nezimenicno vrsi merenje na razlicitim izvorima napona ili kanalima.
Za izvodjenje takvih merenja koristi se sklop pod nazivom
**multiplekser koji ima N kanala koji su povezani sa razlicitim izvorima napona i samo jedan izlaz koji je povezan sa ADC.** Kanali se mogu softverski spojiti sa izlazom cime se omoguvaca programiranje merenja na razlicitim kanalima.
Ako se merenje vrsi dovoljn brzo, to meze biti dobra aproksimacija kada se jos i mereni naponski signali dovoljno sporo menjaju.

## S/H

Za vreme trajanja konverzije, napon koji se meri treba da bude konstantan, sto se postize pomocu elektronskog sklopa Samle and Hold. Sample and Hold je vrsta analogne memorije, jer pamnti vrednosti analogog napona.

## Band Gap internal reference

Referentni naponski razmak je temperaturno nezavisan referentni napon koji se koristi u integrisanim koloma. **Proizvodi fiksni napon bez ozbira na vibracije napajanja, promene temperature ili opterecenja kola.**

## ADMUX registar

**ADC Multiplexer Selection Register** - Moguce je izabrati bilo koji od pinova ADC7 - 0 kako bi zamenili negativni ulaz analognog komparatora.
ADCMUX se koristi za odabir ovog ulaza i prema tome mora biti iskljucen kako bi se ova funkcija koristila. Ako analogni komparator omoguci bit ACME in ADCSRB onda je on podesen i ADC iskljucen tada je ADEN u ADCSRA je nula.

**MUX2..0 u ADMUX bira ulazni pin da zameni negativni ulaz analognog komparatora. Ako je ACME obrisan ili ADEN podesen, AIN1 se primenjuje na negativni ulaz analognog komparatora.**

## ADCSRA registar - Control and Status Register A

ADC modul sadrzi **predskaler koji generise frekvenciju takta ADC-a** sa bilo koje CPU frekvencije iznad 100kHZ.

**Preskaler pocinje da broji od trenutka kada je ADC ukljucen, postavljenjem ADEN bita u ADCSRA. Preskaler nastavlja da radi sve dok je ADEN bit podesen i resetuje se kada je ADEN nizak.**

## ADCH / ADCL registri - The ADC Data Register

Kada se **ADC koverzija zavrsi, rezultat se nalazi u ova dva registra**.
Kada se
**cita ADCL** registar, registar podataka **ADC se ne azurira dok se ADCX ne procita.** Shodno tome, ako se rezultat ostavi prilagodjen i nije potrebno vise od 8-bitne preciznosti, dovoljno je procitati ADCX. U Suprotnom se prvo mora procitati **ADCL**, a zatim **ADCH**.

**Bit ADLAR** u ADMUX, MUXn bitova u ADMUX-u uticu na nacin na koji se rezultat cita iz

registara.
**Ako je ADLAR podesen, rezultat je ostaljen prilagodjen. Ako je ADLAR obrisan, rezultat se podesava udesno.**

## ADCSRB registar - ADC control and Status Register B

Kada je bit postavljen na **logicki jedan i ADC je iskljucen** ( ADEN u ADCSRA je nula). **ADCMUX bira negativni ulaz u analogni komparator.** Kada je ovaj bit upisan kao **logicka nula**, **AIN1** se primenjuje na negativni ulaz analognog komparatora.

---

## Analog comparator

**Analogni komparator uporedjuje ulazne vrednosti na pozitivnom pinu AIN0 i negativnom pinu AIN1.** Kada **napon na pozitivnom pinu AIN0 je veci on napona na negativnom pinu AIN1**, **izlaz analognog komparatora ACO je podesen**.

**Izlaz komparatora** se moze podesiti da aktivira funkciju hvatanja ulaza tajmer/brojac.
**Korisnim moze da izabere okidanje prekida ulaza na uzlaznu, silaznu ili promenu ivice.**

## Instrukcije za sabiranje

LDI R16, 123

LDI R17, 64

**ADD** R16, R17

**ADD Rd, Rr** - saberi dva registra i smesti u Rd ← Rd + Rr

**ADC Rd, Rr** - saberi dva registra sa prenosom C, Rd ← Rd + Rr + C

## Instrukcije za oduzimanje

LDI R16, 123

LDI R17, 64

**SUB** R16, R17

**SUB Rd, Rr** - oduzmi dva registra i postavi u Rd ← Rd - Rr

**SBC Rd, Rr** - Oduzmi sadrzaj dva regustra sa prenosom C, Rd ← Rd - Rr - C

## Množenje 8x8

**[rM1H][rM1L] * [RM2] = [rR1H][rR1L]**

Potrebno je da **RM1L** i **RM2** budu ucitani.

**rM1H** treba da bude cleard kao i registri **rR1H:rR1L** koji sluze za rezultat.
U prvom koraku potrebno je da
**rM2 bude siftovan desno (LSR rM2)**.

Pomera nulu u svoj bit 7, pomera svoje bitove 7 na 1 za jednu poziciju udesno i svoj prethodni bit **0 kao fleg**.
Ako je
**oznaka za prenos jedan, dodaje rM1H:rM1L u registre rezultata, ako nije, ne vrši se dodavanje i preskače se.**

**TST rM2**, ako je **multiplicator ociscen**, tada je mnozenje gotovo.
Ako nije,
**rM1H:rM1L je pomnozeno sa 2, sada se siftuje rM1L za jedan levo i rotira se tako da se carry flag smesta u rM1H. (LSL rM1L, ROL rM1H)**

**Ponavljanje ovih koraka se izvrsava sve dok svi bitovi rM2 su testirani i rM2 nije bude prazan.**

## Binary to decimal

Algoritam:

Oduzmite binarni **100 (0x64)** dok se ne dogodi prenos. Dodajte 100 da poništite poslednje oduzimanje. Zatim oduzimajte binarni broj **10 (0x0A) dok se ne dogodi prenos**. Dodajte 10 da poništite poslednje oduzimanje. **Ostatak broja je poslednja cifra. Ako je potrebno dodavanje nule na praznine, postavite flag (T u SREG) i razmenite nule po praznim mestima sve dok je flag postavljen. Ako se pojavi cifra različita od nule, uklonite flag. Ne koristite flag sa poslednjom cifrom.** Proširivanje konverzije na 16-bitne brojeve: počnite sa binarnim 10000, zatim sa binarnim 1000 i nižim i koristite dva bajta za oduzimanje.

## Kako se kontrolišu 4 7seg LED display - a sa 4 x 8 = 32 izvoda sa 12 pinova, kako se vrši upis znakova na 4 7seg LED display - u na Multifunctionshield - u sa svega 3 pina?

**ddb0 - 5 → izlazni pinovi za g, dp i kontolu anoda**

**pb0 - 1 → izlazi za cifre na 1**

**pb2 - 5 → izlazi na anode na 0**

**dd2 - 7 → postavljanje izlaznih bitova registra D**

**Vrsi se upisivanje brojeva u SRAM → jedinice, desetice, stotine, hiljade**

**Led display se sastoji od 4 crvena 7-segmetna indikatora kojima se mogu generisati slova i cifre.**

Za jednostavno kontolisanje LED display-a korise se **2 serijsko paralelna pretvaraca. Uloga serisko-paralelnog pretvaraca je da omoguci slanje 8 signala za upravljanje svakim segmentom preko jedne serijske linije. Za svako od 4 inidkatora je potbreno 8 linija signala ⇒ 4 * 8 = 32 linije za definisanje 4 znaka. Serijski ulaz konvertora SDI 14 je vezan za digitalni serijski izlaz D8** na Arduino kartici. **Jedan serijsko-paralelni pretvarac se koristi za selekciju jednog od 4 7segmeta, a drugi za kontrolu svih 8 segmenata selektovanog indikatora. 7 segmenta se koristi za generisanje znaka, dok se 8 bit koristi za kontrolu decimalne tacke.** Svaki znak - cifra ili slovo koji je potrebno prikazati je potrebno **kodirati pre slanja preko digitalno serijskog izlaza** Arduino kartice. **Nule ukljucuju segmente, jer je drugi kraj segmenta fisno vezan za pozitivan napon.**

## Šta je EEPROM?

**Electricaly Erasable Programmable Read Only Memory** je treca vrsta memorije atm328 MCU od **1kB**, pred 32kB flesh memorije za smestaj programa i 2kB SRAM radne memorije za podatke.
EPROM memorije nije neophodna za funkcionisanje MCU i ne mora da se koristi. Namena ove

memorije je da se
**koristi prema potrebi za trajno belezenje podataka iz SRAM memorije koja se brise iskljucivanjem napajanja.** Podaci koji su ubelezeni u **EEPROM memoriju koja se ponasa kao ROM, sluze samo za citanje i ostaju trajno sacuvani i posle iskljucivanja napajanja. Posebnim podesavanjima EEPROM podaci se mogu menjati.**

## EEARH i EEARL registri

**EEARH - EEPROM Address Register High,**
**EEARL - EEPROM Address Register Low**

Predstavljaju registere za
**adresiranje EEARH, EEARL - visi i nizi byte.**
EEPROM memorija je
**linearno organizovana sa sopstvenim memorijskim prostorom koji je nezavissan od ostalih vrsta memorije. Pristupom EEPROM memoriji je preko odgovarajucih registara, tako da se razlikuje u odnosu na adresiranje sram i flash memorije.**

## EECR registar postupak čitanja EEPROM

```
EEPROM_read:
; Wait for completion of previous write
SBIC EECR, EEPE
RJMP EEPROM_read
; Set up address (r18:r17) in address register
out EEARH, r18
out EEARL, r17
; Start eeprom read by writing EERE
sbi EECR, EERE
; Read data from Data Register
in r16,EEDR
ret
```

**EECR - Control Register**
**EEPE - Write Enabled**

**SBIC** - skip If bit in I/O register cleared

**OUT** - Out port

**RET** - subroutine return

**SBI** - Set bit in I/O register

**IN** - in port

## EECR - EEPROM Control Register -

Postavka bita - rezima programiranje EEPROM se definise radnja koja ce se izvrsiti kada se vrsi pisanje EEPEn:

0 - 0 - Brisanje i pisanje u jednoj operaciji

0 - 1 - Brisanje

1 - 0 - Pisanje

1 - 1 - Rezervisan prostor

**Citanje - Signal za omogucavanje citanja EEPROM-a EERE - EEPROM READ ENABLED**.
Kada je tacna adresa postavljena u
**EEAR registar**, **EERE bit mora biti upisan u logicki 1 da bi se pokrenulo citanje.**

**Pristup za citanje EEPREM-a traje jedan ciklus - trazeni podaci su odmah dostupni.**

**Korisnik treba da ispita EEPE bit pre pocetka operacije citanje. Ako je opreacija pisanja u toku, nije moguce citanje EEPROM-a, niti za promenu EEAR registra.**

## EECR registar postupak upisa u EEPROM

in R16, SREG ; store SREG value
CLI ; disable intrerrupts during timed srquence
SBI EECR, EEMPE ; start EEPROM write
SBI EECR, EEPE
OUT SREG, R16 ; restore SREG value

**EEMPE** - Master Write enable

Kada su adresa i podaci ispravno podeseni, **EEPE bit biti postavljen na logicku 1** kako bi se **vrednost zapisala u EEPROM**.

Potrebno je da se saceka da
**EEPE ne postane nula, SELFPREGEN u SPMCSR ne postane nula, i opciono je da se upise nova adresa/podaci u EEPROM** → EEAR/EEDR.

**Logicka 1 se upisuje u EEMPE bit dok se nula upisuje u EEPE u EECR.** U roku od 4 ciklusa nakon podesavanja EEMPE, potrebno je upisati logicki ciklus u EEPE. **Softver mora da se proveri da li je programiranje zavrseno pre nego sto je pocelo novo upisivanje.**

---

## EEDR registar - EEPROM Data Register

**Registar za podatke EEDR koji se citaju/upisuju.**

Za operaciju upisivanja u EEPROM, **EEDR registar sadrzi podatke koji se upisuju u EEPROM na adresi koju daje registar EEAR**.
Za operaciju citanja EEPROM-a,
**EEDR sadrzi podatke ocitane iz EEPROM-a na adresi koju je dao EEAR.**

## .ESEG direktiva

**Predstavlja asemblersku direktivu za prebacivanje asemblera u EEPROM segment.**

## Zbog čega je upis niza podataka u EEPROM pogodno vršiti korišćenjem interrupt - a?

Ako su interapti omoguceni to nam omogucava da iskljucimo trenutno CLI kako bi smo se osigurali da vreme upisa nije corrupted.

## Šta je USART?

**Universal Synchronous Asynchronous Receive Transmit** - predstavlja **serijski prenos podataka, bit po bit izmadju dva uredjaja za obradu podataka.** Serijski prenos podataka je standardizovan, postoje stroge specifikacije koje svaki uredjaj mora da ispunjava da bi mogao da razmenjuje podatke sa nekim drugim uredjajem. Specifikacije USART se nazivaju i serijski interface jer implementacija moze da se razlikuje na svakom uredjaju, ali treba da omogucava razmenu podataka.

**USART - sadrzi oba vida serijske komunikacije S - sinhrono, A - asinhrona**

## Osnovna razlika sinhrone i asinhrone komunikacije?

Za **sinhronu komunikaciju se koristi poseban CLK signal koji se prenosi posebnim provodnikom zajedno sa bitovima OUT/IN izmedju uredjaja koji razmenjuju podatke. Bitovi koji se prenose kroz poseban provodnik u odnosu na clock signal su sinhronizovani sa clock signalom.**

**Asinhrona serijska komunikcija takodje koristi clock signal odredjene frekvencije, ali se taj clock signal ne prenosi izmedju uredjaja koji razmedjuju podatke, vec se koristi za definisanje brzine prenosa bitova.**

## Kako je ATM328p mikrokontroler povezan sa USB komunikacijom koja se razlikuje od UART?

UART koristi **3 provodnika za vezu TX (Transmit), RX (Recive) i GND.**
TX i RX su ukrsteni, dok GND definise zajendicki potencijal.

**TX → RX ; RX → TX ; GND → GND ⇒ UART - - - UART**
Pinovi TX - D1 i RX - D0 su preko 16u2 MCU vezani na USB i omogucava resriju

komunikaciju sa PC-em.

**Tx → D1 ; RX → D0 ; GND ; 5.5 ⇒ USB → PC**

## Koji su osnovni parametri serijske asinhrone komunikacije - inicijalizacija UART?

USRAT mora biti **inicijalizovan pre bilo kakve komunikacije.**
Pod inicijalizaciju se podrazumeva podesavanje
**bound rate, frame fomrat i omogucavanje transmitter ili reciver u zavisnosti od namene koriscenja.** Globalni interapt flag treba biti cleared i global interapti disabled.

**TXCn flag se moze koristiti da se proveri da li je prenos zavrsen, a RXC flag se moze koristiti za proveru da li su podaci pristigli u bafer. TXCn mora biti cleared pre svakog prenosa.**

## Šta je parnost, za šta se koristi?

**Bit parnosti se koristi za otkrivanje gresaka. Bit parnosti se nalazi izmedju poslednjeg bita podataka i prvog stop bita serijskog okvira. Postoje dve varijante bita parnisti, paran i neparan paritetni bit.**

**Bit parnosti se izracunava operacijom exclusive-or na sve bitove podataka. Ako se koristi neparan paritet, rezultat je invertovan.**

Za dati skup bitova, broje se bitovi cija je **vrednost 1**. Ako je taj broj **neparan**, vrednost bita **parsnosti se postavlja na 1**, cineci ukupan broj pojavljivanja 1 u celom skupu. Ako je broj 1 u datom skupu bitova vec paran, vrednostt bita parnosti je 0. U slucaju neparnog pariteta, situacija je obrnuta.

## UDR0 registar

**Registar bafera podataka za prenos podataka USART i registri bafera podataka za prijem dele istu I/O adresu koje se referencira na UDRn.**

Registar za prenos podataka
**TXB bice odrediste za podatke upisane u UDRn. Citanje lokacije registra UDRn ce vratiti sadrzaj registra bafera primljenih podataka RKSB.** Za 5,6,7 bitne znakove, **predajnik ce ignosristati gornje nekoriscene bitove i prijemnik ih postavlja na nula.**

**Bafer za prenos moze upisati samo kada je postavljena UDREn fleg u UCSRnA registru.** Podaci upisani kada fleg nije postavljen bice ignorisani od strane predajnika.

**Kada se podaci upisuju u bafer za prenos i predajnik je ukljucen on ce ucitati podatke u registar pomeranja prenosa kada je pomeracki registar prazan.**

**Bafer za prijem se sastoji od dva nivoa FIFO. FIFO ce promeniti svoje stanje kad god se pristupi baferu za prijem.** Ne sme se koristiti SBI CBI na ovoj logaciji, potrebno je koriscenje instrikcija SBIC i SBIS, ali i one ce promeniti stanje u FIFO.

## UBRR0H i UBRR0L registri

**UBRR0H - USART baud rate register high**
**UBRR0L - USART baud rate register low**

**Bit 15 : 12 - rezervisani bitovi za buducu upotrebu. Za kompatibilnost sa buducim uredjajima ovi bitovi moraju biti upisani na nulu kada se UBRRnH je napisan.**

**Bit 11 : 0 - USART Baud Rate Register - registar brizine prenosa, 4 najznacijija bita UBRRnH, i osam manje znacajnih bitova UBRRnL.**

**Tekuci prenosi od predajnika i prijemnika ce biti osteceni ako se promeni brzina prenosa. Pisanje UBRRnL ce pokrenuti trenutno azuriranje brzine prenosa preskalera.**

## UART postupak predaje podataka

Prvi korak se sastoji u **inicijalizaciji USART →**

**postavljanje baud rate zatim se omogucava receiver i transmitter** (1 << RXEN0) | (1 << TXEN0), **postavlja se frame format i ucitavaju se podaci zl, low(2 * podaci) zh, high(2 * podaci)**.

**Nakon cega se definise poruka koja se smesta u registar R16 i podaci se salju. Cekaju se podaci da pristignu u bafer, zatim ide citanje podataka (lds ulaz, udr0) i slanje podataka (sts udr0, ulaz).**

## UART postupak prijema podataka

Prvi korak se sastoji u **inicijalizaciji USART →**

**postavljanje baud rate zatim se omogucava receiver i transmitter** (1 << RXEN0) | (1 << TXEN0), **postavlja se frame format i ucitavaju se podaci zl, low(2 * podaci) zh, high(2 * podaci)**.

**UART_ Transmit → Cekamo da se isprazni bafer za prijem podataka, nakon toga radimo upis podataka u registar R16 i saljemo podatke.**

---

EXAMPLE 1 :: LED

```
.include "m328pdef.inc"
ldi r16, (1 << PINB0)
out DDRB, R16
OUT PORTB, R16
loop:
rjmp loop
```

EXAMPLE 2 :: ADDING 16-bit NUMBERS

```
.include "m328pdef.inc"
.def num1L = R16
.def num1H = R17
.def num2L = R18
.def num2H = R19

ldi num1L, 0x34
ldi num1H, 0x12
ldi num2L, 0xCD
ldi num2H, 0xAB

add num1L, num2L
adc num2H, num2H
```

```
loop:
rjmp loop
```

EXAMPLE 3 :: BLINK LED 1Hz

```
.include "m328pdef.inc"
.def mask = R16
.def ledR = R17
.def oLoopR = R18 ; inner loop register
.def iLoopRl = R24 ; inner loop register low
.def iLoopRh = R25 ; inner loop register high

.equ oVal = 71 ; outer loop
.equ iVal = 28168 ; inner loop

clr ledR
ldi mask, (1<<PINB0)
out DDRB, mask
start:
eor ledR, mask
out PORTB, ledR
ldi oLoopR, oVal

oLoop:
ldi iLoopRl, LOW(iVal)
ldi iLoopRh, HIGH(iVal)

iLoop:
sbiw iLoopRl, 1
brne iLoop
dec oLoopR
brne oLoop
rjmp start
```

EXAMPLE 4 :: 8x8

```
Main:
.ifdef SPH ; if SPH is defined
  ldi rmp,High(RAMEND)
  out SPH,rmp ; Init MSB stack pointer
  .endif
    ldi rmp,Low(RAMEND)
    out SPL,rmp ; Init LSB stack pointer
; ...
    sei ; Enable interrupts

  clr m1h
  ldi m1l, 5
  ldi m2, 7
  clr rl
  clr rh

  mul m1l, m2

mult:
  lsr m2
  brcc test
  add rl, m1l
  adc rh, m1h
test:
  tst m2
  breq kraj
  lsl m1l
  rol m1h
  rjmp mult
kraj:
    rjmp kraj
```

EXAMPLE 5 :: TASTER

```
.include "m328Pdef.inc"
sbi ddrd, ddd5; arduino digital pin 5 led as output
cbi portd, portd5; arduino digital 5 led off
cbi ddrd, ddd2; arduino digital 2 as input
sbi portd, portd2; 2 pull up
loop:
sbis pind, pind2; jump if taster 2 not pressed - 1
sbi portd, portd5; led 5 on
sbic pind, pind2; jump if taster 2 pressed - 0
cbi portd, portd5; led 5 off
rjmp loop
```

EXAMPLE 6 :: Toggle

```
.include "m328Pdef.inc"
SBI DDRB, DDB5; Set the direction bit of PB5 high - output
Toggle:
SBI PORTB, PORTB5 ; Set the output port pin PB5 to HIGH
CBI PORTB, PORTB5 ; Clear the output port pin PB5 to LOW
RJMP Toggle ; Jump back to the label Toggle
```