LABORATOR 6 – Convertorul analog-numeric (ADC)

Scopul lucrării

Se va conecta la microcontrolerul ATmega16 o sursă de tensiune variabilă, valoarea tensiunii se va converti numeric și se va afișa pe LCD. Revedeți principiul de funcționare a convertorului analog-numeric cu aproximări succesive din prelegere 3, capitolul „**Principiul de funcționare al ADC**”.
Schema de principiu a convertorului analog numeric este prezentată în figura următoare:

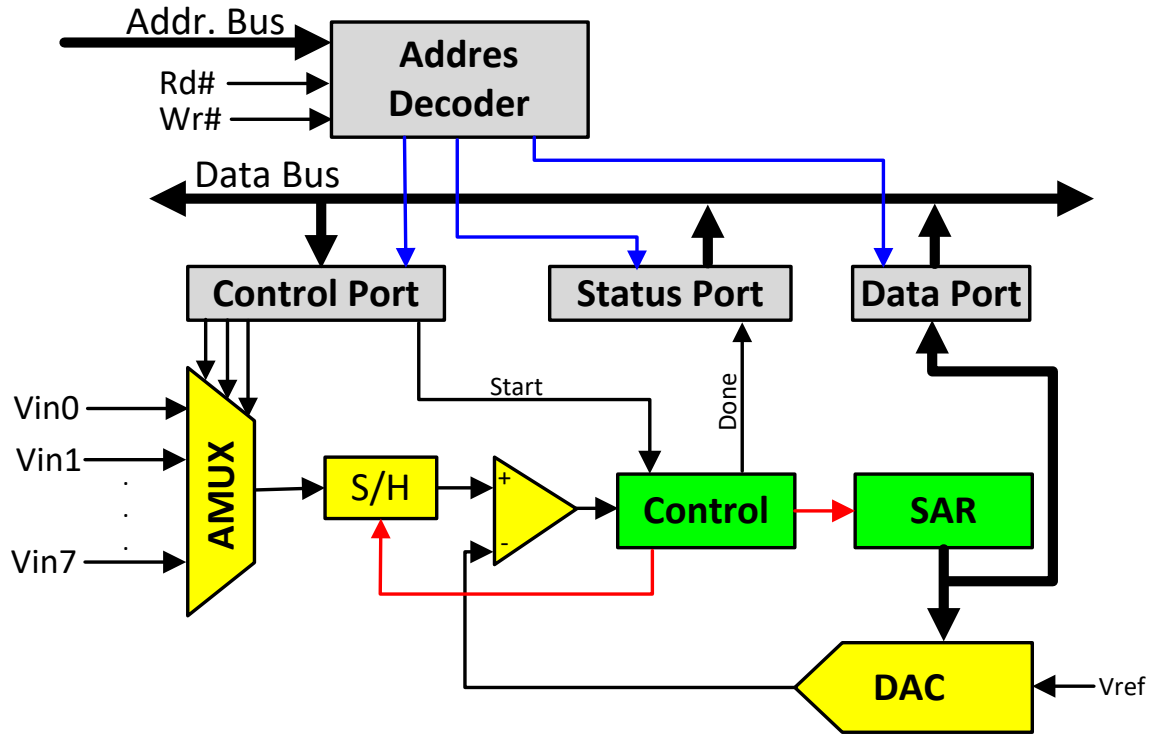
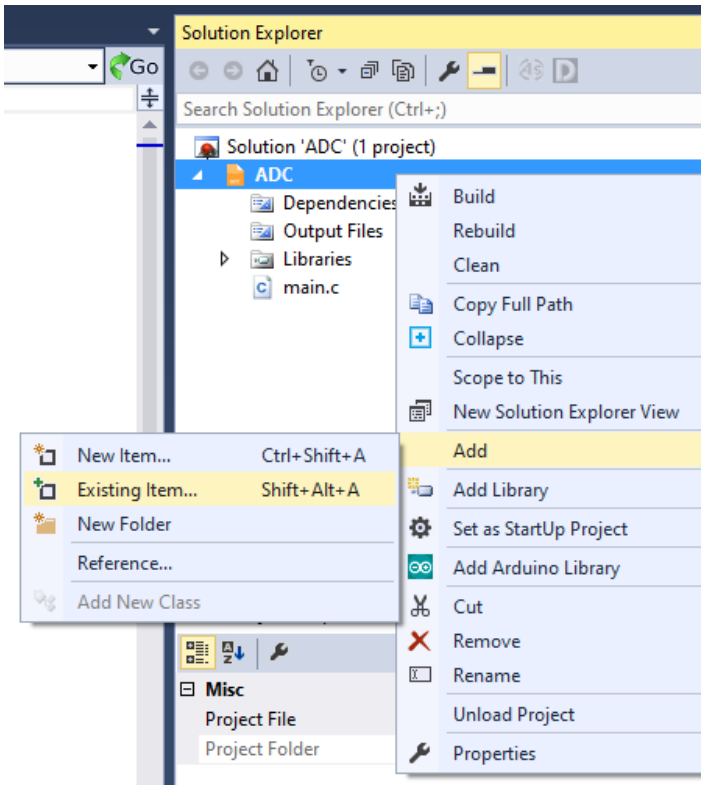


figura 1

Figura a fost preluată din prelegerea 3.

Pasul 1: Crearea proiectului ADC



- **Creaţi** un nou proiect numit **ADC**.
- Sursele LCD.h şi LCD.c se găsesc în proiectul test_LCD din platforma de laborator LCD. **Copiați aceste surse** în subfolderul ADC al folderului ADC.

ADC este folderul soluției iar subfolderul ADC este folderul proiectului.

- Adăugați aceste surse la proiect. Pentru a adăuga o sursă/surse la proiect faceți clic dreapta pe numele proiectului (ADC) în fereastra soluției și din meniul contextual care va apare selectați **Add,** iar apoi selectați **Existing Item... .** Vezi figura alăturată.
- **Redenumiți** fișierul main.c. Noul nume va fi ADC.c. **Ștergeți** conținutul acestui fișier.

Pasul 2: Crearea schemei

Faceți o copie a schemei LCD.simu din laboratorul precedent. Redenumiți copia ca ADC.simu.

Adăugați pe schema ADC.simu o sursă de tensiune ajustabilă, sursă deja folosită în laboratoarele 1 și 2. **Conectați sursa la pinul PA0 din portul A**. Sursa va furniza tensiunea ce se va converti numeric.

Pasul 3: Conversia analog-numerică

Convertorul analog-numeric este prezentat în capitolul 22 din datele de catalog ale microcontrolerului ATmega16 disponibile la https://drive.google.com/file/d/1loup5AoqPNgRn_eNs9udbxe0o9Kb8AqD/view . Vom folosi în special subcapitolul **22.9 Register Description**.

Informația din datele de catalog au fost prezentate și în capitolul 4, „**Interfețe/controlere IO – Convertorul Analog-Numeric (CAN -ADC)**”, subcapitolul „**Convertorul analog numeric din ATmega16**” din prelegerea 3. Pentru a nu fi obligați să comutați între notele de curs, platforma de laborator și proiect informația relevantă din notele de curs se va relua în continuare în această platformă de laborator.

Ca și în laboratoarele precedente, codul care se va copia din platformă în ADC.c este scris cu verde. Acest cod va trebui completat acolo unde este specificat.

```
#include "LCD.h"
#include <avr/io.h>

int main(){
   unsigned  int val;
   char buf[20];
   initLCD();
```

In secțiunea de inițializare, înainte de while(1), se vor programa registrele de control ale controlerului. Programarea inițială se va face în secțiunea de inițializare, înainte de bucla principală.

Primul registru care se programează este registrul ADMUX descris în documentație în **capitolul 22.9.1**, pagina 221, din documentație și în **prelegere 3, pagina 25 și 26**. Descrierea din curs este reluată în continuare:

ADMUX – ADC Multiplexer Selection Register.

În figura 2 este prezenta modul de conectare al registrului ADMUX:

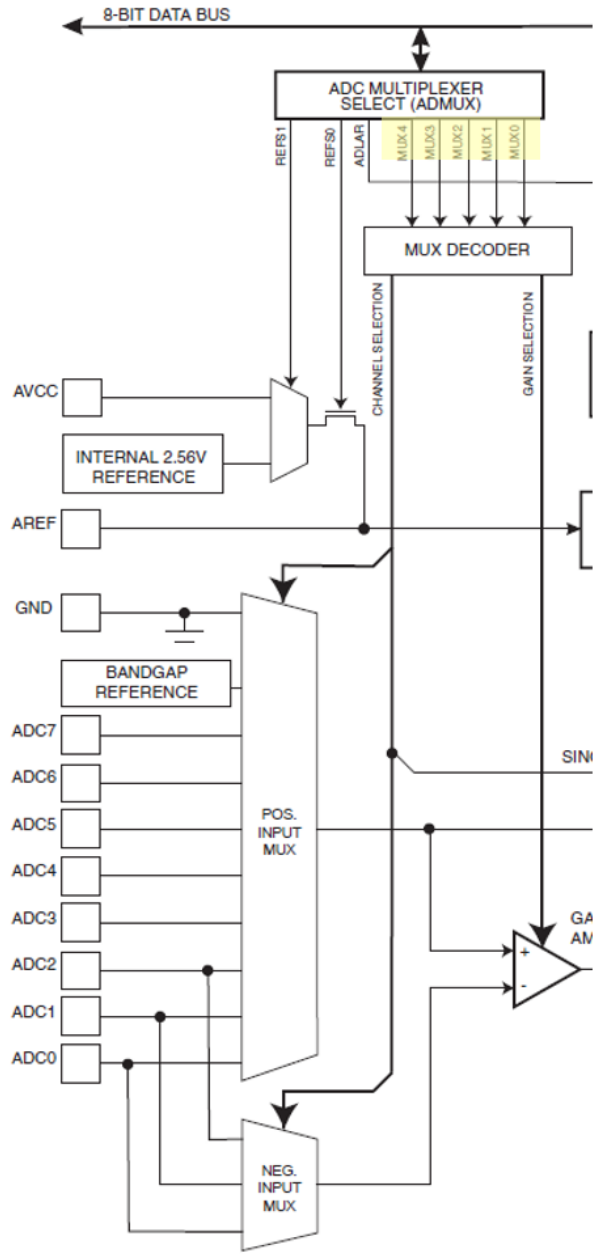


Table 22-4. Input Channel and

MUX4:0	Single Ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

figura 2

În documentație în loc de port se mai folosește termenul registru IO.

Multiplexorul analogic selectează ce intrare analogica este convertită numeric. Multiplexorul are 10 intrări și este prezentat în figura 2. Intrările conectate la pinii microcontrolerului sunt notate ADC0 – ADC7 iar pinii corespunzători sunt pinii portului A (PA0-PA7). Selecția multiplexorului analogic provine din portul de control **ADMUX**. Semnificația biților din acest port, preluată din documentație, pagina 221, este următoarea:

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	

figura 3

Selecția intrării analogice ce va deveni intrare a convertorului analog-numeric este făcută de biții **MUX4:0** marcați cu galben în figura 3. Valorile biților de selecție corespunzătoare intrărilor analogice ADC7:0 sunt prezentate în tabelul alăturat figura 2. Este posibilă și selecția diferenței a două intrări

analogice. Această opțiune este foarte utilă pentru conectarea de senzori cu ieșire diferențială. Deoarece modul diferențial este necesar mai rar, nu îl vom detalia.

În continuare vom discuta selecția tensiunii de referință VREF. Selecția se face sub controlul biților **REFS1:0** marcați cu bleu în figura 3. Rolul acestor biți este prezentat în tabelul următor:

Table 22-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

După cum s-a prezentat anterior elementul sensibil din convertorul analog-numeric este tensiunea de referință. Tensiunea cu care se compara tensiunea de intrare Vin este VDC (ieșirea DAC). VDC depinde de VREF după legea $VDC=SAR/2^{size(SAR)}$ VREF. Dacă VREF fluctuează VDC fluctuează la rândul ei și rezultatul conversiei va fi afectat.

Cea mai stabilă tensiune de referință este tensiunea internă de 2.56V. Din păcate valoarea acesteia este de doar 2.56V face ca valoarea maximă a lui Vin sa fie tot 2.56V. De multe ori acesta valoare este insuficientă. Din acest motiv există și opțiunea de a folosi un circuit extern – AREF în tabel. În acest caz precizia conversiei depinde calitatea circuitul extern; o sursa de tensiune de referința de calitate este foarte scumpă (tipic de trei ori mai scumpă decât microcontrolerul).

O treia opțiune este să se folosească drept referința tensiunea de alimentare a convertorului analog-numeric (AVCC). Acesta opțiune este cea mai puțin precisă, dar în multe aplicații sunt suficienți și 8 biți.

ADLAR. Rezultatul conversiei se poate citi prin intermediul a două porturi de intrare: ADCL și ADCH. Sunt necesare două porturi deoarece conversia se face pe 10 biți iar porturile au lățimea maximă de 8 biți.

Rezultatul conversiei este disponibil în două moduri. Aceste moduri sunt prezentate mai jos:

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Modul în care se citește rezultatul depinde de bitul ADLAR în portul ADMUX. Dacă ADLAR='1', rezultatul este aliniat la stânga iar dacă ADLAR='0' rezultatul este aliniat la dreapta. Vezi figura de mai sus.

După ce se citește ADCL, registrele de date ADC nu sunt actualizate până când nu este citit și ADCH.

În consecință, dacă rezultatul este aliniat la stânga și nu este necesară o precizie mai mare de 8 biți, este suficient să se citească numai ADCH. În caz contrar, trebuie citit mai întâi ADCL și apoi ADCH.

În continuare vom seta ADMUX. Setarea registrelor IO se comentează după cum urmează. Se va explica pentru ADMUX:

```
//1 ADMUX register
//2      REFS1 REFS0 ADLAR MUX4 MUX3 MUX2 MUX1 MUX0
//3         |     |     |     ?    ?    ?    ?    ?    Vin = ?
//4         |     |     ?  aliniere la stânga
//5         ?     ? VREF = ?
ADMUX =0b????????;
```

Pe prima linie scrieți numele registrului care se configurează – registrul ADMUX în cazul de față.

Pe a doua linie scrieți semnificația biților ce compun registrul. Copiați semnificația din documentație.

Pe a treia linie înlocuiți ‚?' cu valorile corespunzătoare pinul pe care este aplicată tensiunea de convertit, adică pinul la care ați conectat sursa de tensiune reglabilă. Aceste valori se găsesc în tabelul 22-4.

Pe a patra linie înlocuiți ‚?' cu bitul valoarea bitului ADLAR pentru care rezultatul conversiei este aliniat la stânga. Semnificația ADLAR este prezentată în documentație la pagina 224.

Pe a cincea linie alegeți valorile pentru REFS1:0 astfel încât VREF să fie AVCC.

Când ați terminat de configurat ADMUX nu mai trebuie să mai existe caractere ‚?'.

```
// ADMUX register
//REFS1 REFS0 ADLAR MUX4 MUX3 MUX2 MUX1 MUX0
//  |     |     |     ?    ?    ?    ?    ?    Vin = ?
//  |     |     ?  aliniere la stânga
//  ?     ? VREF = ?
ADMUX =0b????????;
```

ADCSRA – ADC Control and Status Register A.

Este descris în documentație în capitolul 22.9.2. Acest port este de fapt alcătuit din două porturi: un port buffer și un port de stare mapate la aceeași adresă. ADCSRA este prezentat în figura următoare:

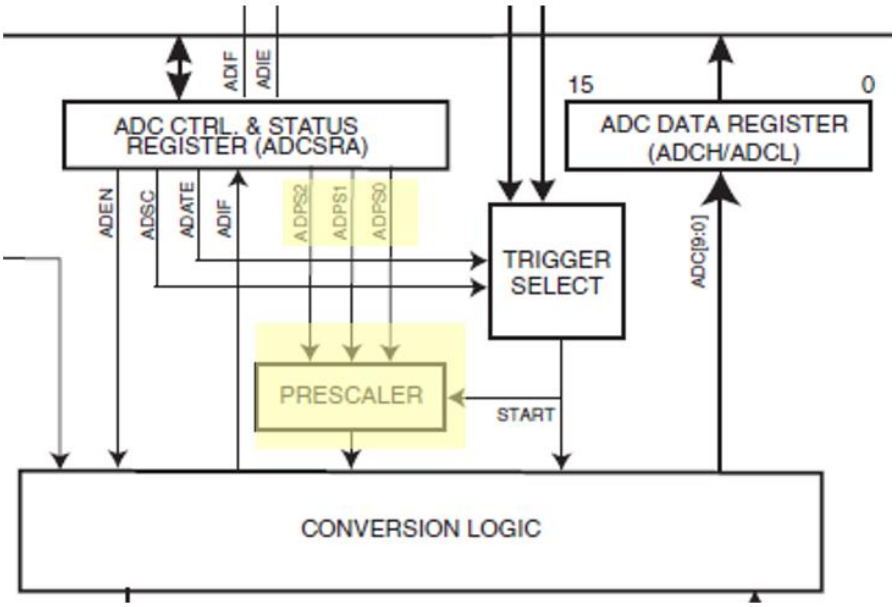


figura 4

Structura registrului IO ADCSRA (porturile se mai numesc registre IO) este următoarea:

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
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

Semnificația biților din ADCSRA este următoarea:

Bit 7 – ADEN: ADC Enable

Scrierea acestui bit cu ‚1' activează ADC iar scrierea cu ‚0' dezactivează ADC. Dezactivarea este utila pentru reducerea consumului de putere în aplicațiile care nu au nevoie de ADC. Dacă ADC este dezactivat conversia analog numerică nu este posibilă.

Bit 6 – ADSC: ADC Start Conversion

Acest bit este bitul **Start** din figura 1. Atunci când scris cu ‚1' pornește conversia analog-numerică. ADSC rămâne ‚1' atâta timp cât o conversie este în curs de desfășurare. Când conversia este finalizată, aceasta revine la zero. Scrierea cu ‚0' zero a acestui bit nu are nici un efect.

Bit 5 – ADATE: ADC Auto Trigger Enable

Este folosit pentru porni conversia în mod automat pe frontul pozitiv a unui semnal periodic. Modul Auto trigger este folosit pentru a avea permanent la dispoziție valoarea numerică a lui Vin. Modul Auto trigger nu se va detalia și nu se va folosi la curs sau laborator. Întotdeauna scrieți acest bit cu ‚0'.

Bit 4 – ADIF: ADC Interrupt Flag

ADIF este bitul **Done** din figura 1.**Error! Reference source not found.** Acest bit este scris cu ‚1' de blocul de control atunci când se termină o conversie ADC și este actualizat portul de date. ADIF este șters prin scrierea lui cu ‚1' (**nu cu ‚0'!**). ADIF poate genera întrerupere dacă sunt îndeplinite și alte condiții. Se va detalia în prelegerea despre întreruperi.

Starea ADC este caracterizată de biții ADSC și ADIF. Valorile posibile ale acestor doi biți sunt prezentate în tabelul următor:

Tabel 1

ADSC (START in figura 1)	**ADIF (DONE in figura 1)**	**Stare ADC**
0	0	ADC este în **repaus** (idle)
1	0	Conversie **în curs** de execuție.
0	1	S-a **terminat** conversia
1	1	**Imposibil**. Nu se poate ADC să execute conversie și în același timp conversia să se fi terminat.

Bit 3 – ADIE: ADC Interrupt Enable

Una din condițiile ca ADIF să genereze întrerupere este ca ADIE să fie ‚1'. Se va detalia în prelegerea despre întreruperi. Până atunci acest bit va menținut la ‚0'.

Biții 2:0 – ADPS2:0: ADC Prescaler Select Bits

La sfârșitul capitolului Error! Reference source not found. au fost enumerate caracteristicile ADC cu aproximări succesive. Una dintre caracteristici este că determinarea unui bit din SAR necesită un timp relativ mare, tipic 10 us. La fel se comporta și ADC din ATmega16.

La Atmega16 circuitul de control necesită o frecvență de ceas de intrare între 50 kHz și 200 kHz pentru a obține o rezoluție maximă. Dacă este necesară o rezoluție mai mică de 10 biți, frecvența ceasului aplicat blocului de control poate fi mai mare de 200 kHz pentru a obține o rată de eșantionare mai mare. Modulul ADC conține un divizor, care generează o frecvență de ceas acceptabilă pentru orice frecvență CPU mai mare de 100 kHz.

Factorul de divizare este setat de biții ADPS din ADCSRA. Acești biți determină factorul de divizare între frecvența ceasului microcontrolerului și ceasul blocului de control din ADC.

Factorii de divizare stabiliți prin intermediul biților ADPS sunt prezentați în tabelul următor:

Table 22-5. ADC Prescaler Selections

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

Factorul de divizare se alge din tabelul 22-5 astfel ca frecvența ADC sa fie în plaja 50-200kHz.
Frecvența ADC se obține prin divizarea ceasului procesor – 8MHz in acest laborator – cu factorul de divizare ales. Mai exact, trebuie sa găsiți factorul de divizare astfel încât:

$$\frac{8MHz}{factor\ de\ divizare} \in [50KHz, 200KHz].$$

```
// ADCSRA register.
// ADEN ADSC ADATE ADIF ADIE ADPS2 ADPS1 ADPS0
//  |    |     |     |    |     ?     ?     ?    factor divizare=?
//  |    |     0     |    0 ADIE și ADATE se setează la ‚0’ și nu se mai modifica
//  |    |           ?
//  |    ?                                  ?
//  ?
ADCSRA = 0b??0?0???;
```

În bucla principală trebuie să pornim conversia, trebuie sa determinăm când conversia s-a terminat și după terminare să afișam rezultatul.

Atenție: pentru setare, resetare, testare bit folosiți macro definițiile `setbit`, `clrbit` și `testbit`. Nu folosiți numărul bitului, folosiți numele acestuia. De exemplu, nu folosiți

```
clrbit(ADSCRA, 4);
```

folosiți

```
clrbit(ADSCRA, ADIF);
```

Definiția `#define ADIF 4` este deja scrisă în `io.h`

Modifică codul următor:

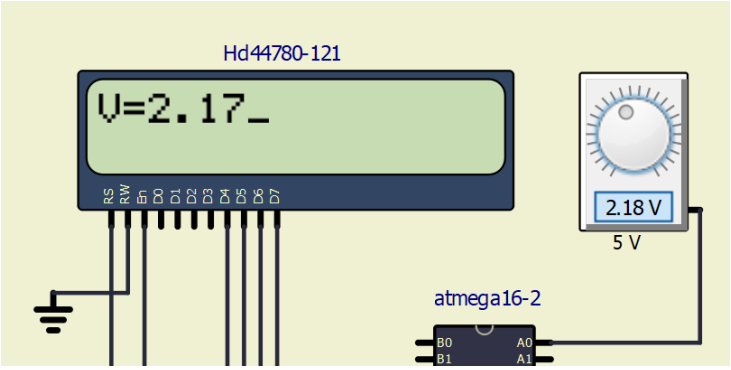
```
while(1){
   // dacă ADC este în repaus adică nu există conversie în curs. Folosiți Tabel 1
   // pentru detectarea stării de repaus.
   if( ??????? ) {
      setbit(?, ?); start conversie
   }

   // Folosiți Tabel 1 pentru detecția terminării conversiei
   //dacă s-a terminat conversia:
   if( testbit( ?, ?)) {
      //sterge bitul care indică sfârșitul conversiei
      clrbit( ?, ?);

      //citește rezultatul conversiei din portul de date conform setării ADLAR
      val= din ce port?

      // convertește val în string pentru a-l afișa. Rezultatul conversiei
      // se va afla în buf.

      // Pentru conversie se recomanda metoda câtului și a restului.
      // Faceti conversia pe trei cifre.

      // Daca nu știți metoda câtului si arestului puteți folosi sprintf(...) dar
      // codul rezultat va fi mult mai mare.
      // Cine folosește sprintf nu primește bonus.
      gotoLC(1, 1);
      putsLCD(buf);
   }
}//end while(1)
}//end main()
```

Testați! Pe afișor trebuie să valori între 0 și 255 când tensiunea de la sursa variază între 0V și 5V. Dacă nu funcționează modificați, testați, modificați, până funcționează.

Verificare intermediară: Când funcționează chemați profesorul.

Pasul 4: Afișarea tensiunii

Se cere să se afișeze pe LCD tensiunea oferită de sursa programabilă, ca în figura următoare:



Tensiunea de intrare se va afișa cu două zecimale. Valoarea afișată pe LCD trebuie să fie valoarea tensiunii generată de sursă ± 0.1V.

Anterior s-a afișat rezultatul conversiei, rezultatul fiind un număr între 0 și 255. Calcularea valorii tensiunii din codul generat de CAN se bazează pe dependența tensiune - cod conversie.

În cazul în care codul ADC este zero pentru tensiune de intrare 0V calculul tensiunii se poate face cu regula de 3 simplă:

Cod ADC tensiune
255 .....5V
val_ADC.....x

$$x = \frac{5V \ \ val\_ADC}{255} = \frac{val\_ADC}{255} 5V$$

De exemplu, dacă codul oferit de ADC este 128 → $Vin = \frac{128}{255} 5V = 2.5V$

Atenție: dacă variabila alocată codului ADC este de tip `int`, rezultatul este 2.

Atenție: **simulatorul nu suportă** variabile de tip `float` sau `double`. Dacă folosiți asemenea variabile, rezultatul va fi greșit. Și dacă nu ar fi greșit folosirea tipurilor `float` sau `double` conduce la un cod mașină mare deoarece microcontrolerele nu au procesor de virgulă mobilă. Toate operațiile cu numere în virgulă mobilă se implementează cu ajutorul unei biblioteci. Asta duce la cod mare și timp de execuție mare.

Soluția este să scalați calculul, adică să înmulțiți cu 100 și apoi sa afișați punctul zecimal.

Pentru exemplul anterior efectuăm: $Vin = \frac{100 \ 128 \ 5}{255} = 250$

Toate calculele anterioare se fac cu constante și variabile de tip întreg. **Atenție**: dacă în Studio folosiți numai variabile și constante de tip `int` **rezultatul va fi greșit! De ce?** Nu uitați! În Microchip Studio `int` se reprezintă pe 16 biți. Există două rezolvări! Găsiți una din ele!

Rezultatul este 250 dar este de 100 de ori mai mare decât rezultatul real. Tot ce rămâne de făcut este să afișam cifra sutelor, adică 2, să afișam punctul zecimal și apoi să afișăm restul de cifre, adică 50.

Verificare finală: Când funcționează chemați profesorul.

Această secțiune poate fi necesară pentru bonus. Poate apare și la examenul final.

În aplicațiile reale afișarea tensiunii de intrare în ADC nu este de mare folos. Să presupunem că tensiunea de intrare este generată de un senzor de temperatură. Atunci informația de interes, care trebuie afișată, este temperatura.

În general datele de catalog ale senzorului specifică dependența între mărimea măsurată și **valoarea** tensiunii generată de senzor. Această dependență este în 95% din cazuri liniară, adică este de forma:

$$Val = mVin + n$$

Tot în datele de catalog ale senzorului se prezintă direct sau indirect două perechi (*Val*, *Vin*). De exemplu, pentru un senzor de temperatură aceste două perechi sunt perechea (-40^0C, 0V) și perechea (100^0C, 5V). Aceste două perechi sunt suficiente pentru a afla parametrii *m* și *n*. Puțină matematică!

În concluzie afișarea unei mărimi fizice (temperatură, presiune, etc.) se face în 3 pași:

1. Se calculează valoarea tensiunii de intrare în funcție de codul citit de la ADC.
2. Se folosesc datele de catalog ale senzorului pentru a calcula pe *m* și *n*.
3. Se afișează *Val*.