

LABORATOR 2 – Prima aplicație

Implementările practice la laboratorul disciplinei „Proiectarea sistemelor cu microprocesor” se fac folosind placa de dezvoltare și microcontrolerul **ATmega16A** din gama **AVR**, produs de firma **Atmel**. Datele de catalog pentru acesta se afla pe pagina cursului, în secțiunea **ATmega16A Data Sheet**.

Chestiuni teoretice: Dezvoltarea aplicațiilor pentru microcontrolere.

Dezvoltarea aplicațiilor pe PC presupune o succesiune de faze. Acestea sunt:

1. Editarea surselor.
2. Compilarea acestora
3. Editarea legăturilor
4. Depanarea
5. Obținerea fișierului executabil (care funcționează corect).

Toate aceste programe au o interfață grafică comună care se numește **IDE** (integrated development environment - *mediu integrat de dezvoltare*). Un exemplu de IDE este **Visual Studio** de la Microsoft. Toate programele care constituie IDE rulează pe aceeași mașină (calculator).

Dezvoltarea aplicațiilor pentru microcontrolere sau microprocesoare cu forță redusă de calcul nu se poate face în acest fel deoarece forța de calcul a acestor circuite este mică sau foarte mică. De aceea dezvoltarea aplicațiilor care vor rula pe microcontrolere presupune folosirea a trei componente:

- un calculator puternic de tip **PC**,
- un montaj hardware – **target board** (montajul țintă) – ce conține microcontrolerului (microcontroler se va abrevia μC) plus hardware-ul necesar aplicației ce se dezvoltă: LED-uri, afișoare 7 segmente, LCD, minitastaturi, etc.
- un pachet de programe tip **IDE** care rulează distribuit: o parte pe PC și o altă parte pe μC .

Ansamblul celor trei componente funcționează după cum urmează:

1. **Se obține codul executabil pentru μC folosind un calculator puternic (tip PC).** Calculatorul pe care se face dezvoltarea aplicației se numește calculator gazdă sau mașina gazdă (host machine). Acesta este un calculator puternic, de obicei un PC. Partea de IDE care rulează pe mașina gazdă conține editor de texte, asamblor, compilator (de obicei de C), linker și interfață grafică pentru rularea comodă a acestor programe. În urma compilării și linkerării se obține **cod mașină** pentru μC . Evident, acest cod ce nu poate fi executat pe mașina gazdă. Modul de operare prezentat mai sus se numește cross-compilare.
2. **Codul mașină rezultat la pasul anterior se transferă pe microcontroler.** În funcție de tipul memoriei în care va fi rezident codului mașină există două variante majore: ROM sau RAM. În continuare se va prezenta numai varianta ROM deoarece μC de tip **ATmega** și implicit montajele construite pe baza acestora aparțin acestei categorii. În varianta ROM μC dispune de resurse hardware dedicate pentru:
 - comunicația cu mașina gazdă.
 - înscrisirea codului mașină generat la punctul 1.
 - depanare (debug). Depanarea înseamnă breakpoint, watches, step-by-step, etc.

Comunicația între mașina gazdă și μC se face prin intermediul unui hardware special. În cazul anumitor μC AVR o variantă a acestui hardware se numește **JTAG ICE**. JTAG ICE se conectează cu gazda prin intermediul unei interfețe USB și cu microcontrolerul AVR prin interfața JTAG, ca în figura 1:



figura 1

3. **Se depanează programul de aplicație.** Pentru depanare mediul de dezvoltare este prevăzut cu depanator simbolic.
4. După ce programul de aplicație funcționează corect pe montajul ce conține μC , acesta **se deconectează** de mașina gazdă și va funcționa de sine stătător, independent.

Scopul lucrării.

- **Se va realiza un montaj** bazat pe ATmega16 la care se va conecta un LED. LED-ul se va conecta la bitul 0 al portului A.
- **Se va scrie programul C care face să clipească** acest LED aproximativ o dată pe secundă.

Pasul 1: Plasarea microcontrolerului.

Microcontrolerul se plasează pe placa de dezvoltare ca în figura 2:

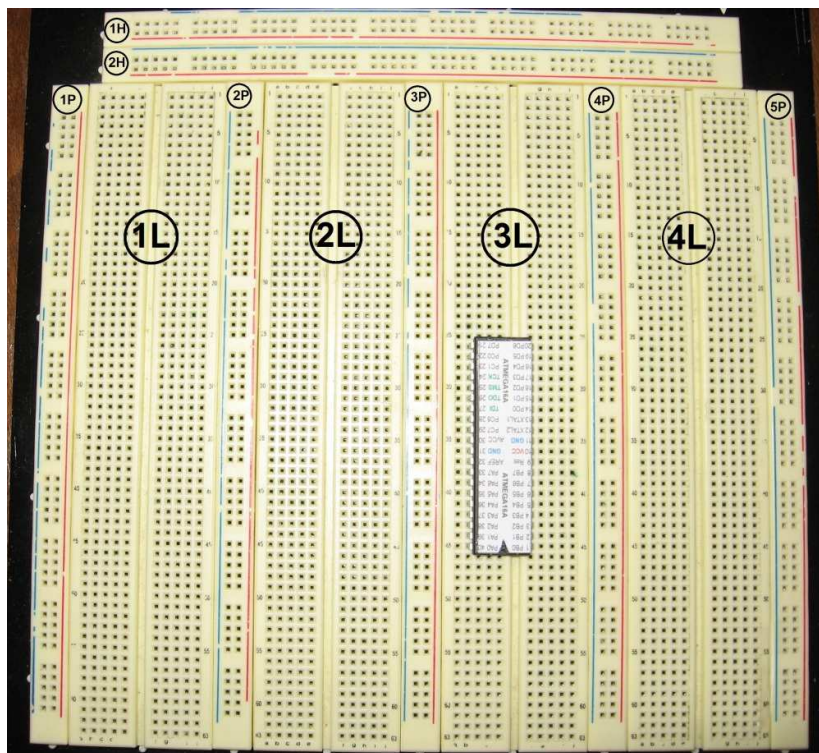


figura 2

Respectați următoarele cerințe **OBLIGATORII**:

- Plasarea μC se va face în secțiunea 3L (L-Logică).
- Pinul 1 trebuie să intre în rândul 45. Identificați acest marcaj pe placa de dezvoltare.
- Pinul 20 trebuie să intre în rândul 26.
- Se lasă patru găuri libere spre coloana 4P (P-Power) și o singură gaură spre 3P.

Pasul 2: Realizarea conexiunilor de alimentare pe placa de dezvoltare.

Pentru a alimenta μC mai întâi se conectează coloanele 3P și 4P la linia 2H prin intermediul firelor verzi și albastre din figura 3b.

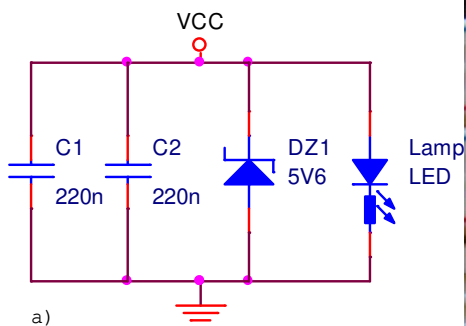
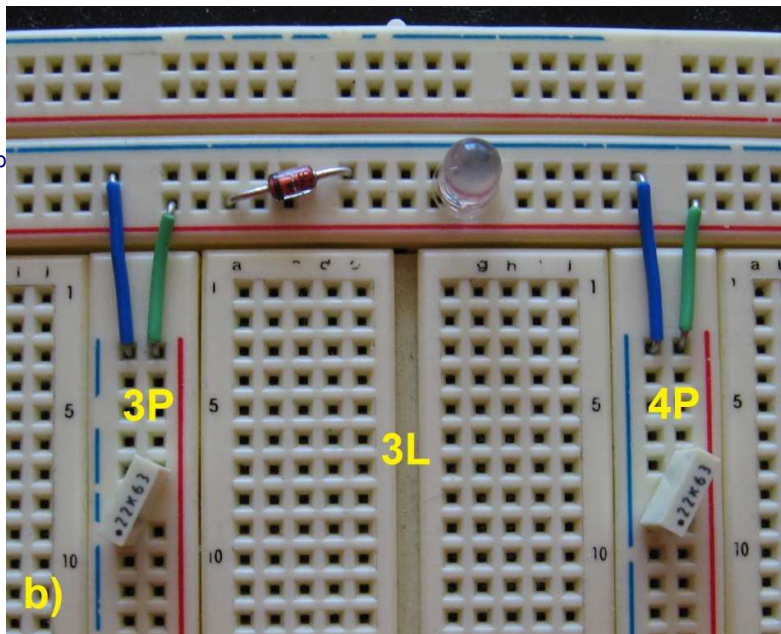


figura 3



Apoi se plantează piesele și se fac conexiunile din figura 3a:

1. Condensatoarele de decuplare C1 și C2 se plantează pe coloanele 3P și 4P. Aceste condensatoare micșorează variațiile de tensiune rezultate în urma fluctuațiilor de consum din montaj.
2. Dioda Zenner DZ1 are rol de protecție la supratensiune sau alimentare inversă. Montați DZ1 ca în figura 3b.
3. LED-ul cu rezistență internă, numit în continuare lampă LED, luminează când montajul este alimentat. **De fiecare dată când veți modifica montajul acest LED trebuie să fie stins!** Modificarea montajului sub tensiune poate duce la distrugerea componentelor.

Pentru realizarea corectă a montajului trebuie identificate anodul și catodul lămpii LED. Deoarece lampa LED și LED-ul se aseamănă foarte mult și pentru că în scurt timp va trebui să conectați în montaj și un LED obișnuit, fără rezistență internă, cele două componente vor fi tratate împreună.

Pentru a identifica anodul și catodul unui LED obișnuit, fără rezistență, priviți figura 4a:

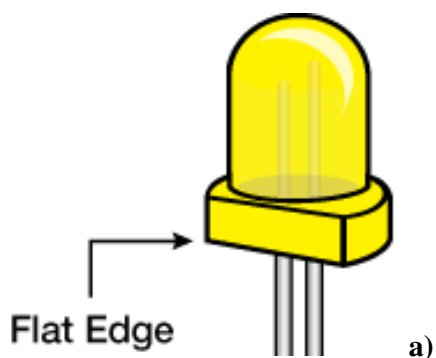


figura 4



Există o singură modalitate sigură de identificare ce funcționează întotdeauna:

Capsula de plastic a LED-ului are o teșitură în dreptul catodului. Folosiți numai această metodă!

Pentru a diferenția LED-ul de lampa LED priviți figura 4b. În cazul lămpii LED de care dispunem **cele două terminale din interiorul capsulei de plastic** arată la fel. Există însă lămpi LED care arată ca un LED obișnuit.

Atenție! Nu montați un LED obișnuit în locul unei lămpi LED deoarece veți distruge LED-ul!

Lampa LED cu care veți lucra are terminalele scurte pentru a putea fi montată la nivelul plăcii de dezvoltare. **Montați lampa LED ca în figura 3a și b.**

Pasul 3: Conectarea microcontrolerului.

În orice sistem cu microprocesor sau μC primele trei aspecte care trebuie tratate cu maximă atenție sunt **alimentarea, semnalul de reset și semnalul de ceas**. Semnalul de reset și cel de ceas sunt necesare deoarece $\mu P/\mu C$ conține blocuri logice secvențiale cum ar fi bistabili, registre, numărătoare și mașini de stare.

Dacă tensiunea de alimentare nu are valoarea specificată în datele de catalog sau prezintă variații, în cel mai bun caz sistemul nu va funcționa iar în cel mai rău caz va fi instabil (instabil = un timp funcționează corect iar apoi, aleatoriu, se blochează). Cum instabilitatea poate avea atât cauze hardware cât și software, determinarea cauzei reale a instabilității este foarte dificilă în practică. De aceea este de dorit ca alimentarea să nu fie printre cauzele posibile ale instabilității.

La fel ca alimentarea, frecvență incorectă a ceasului, valorile incorecte de tensiune ale ceasului sau resetului duc la nefuncționare sau instabilitate.

În orice aplicație cu micro, acordați maximă atenție alimentării, resetului și ceasului!

În continuare se va conecta alimentarea μC și LED-ul.

Semnalul de ceas lipsește deoarece în această primă aplicație se va folosi ceasul intern din ATmega16. Problema semnalului de ceas va fi tratată ulterior.

Semnalul pentru RESET lipsește deoarece resetul intern din ATmega16 este suficient. Resetul extern (pinul \overline{RESET}) se conectează numai atunci când sistemul se poate bloca. Resetul extern este asemănător cu semnalul generat la apăsarea butonului reset la un calculator compatibil IBM PC. Dacă sistemul nu se blochează niciodată, resetul extern este inutil.

Modul de conectare al microcontrolerului ATmega16A este prezentat în figura 5:

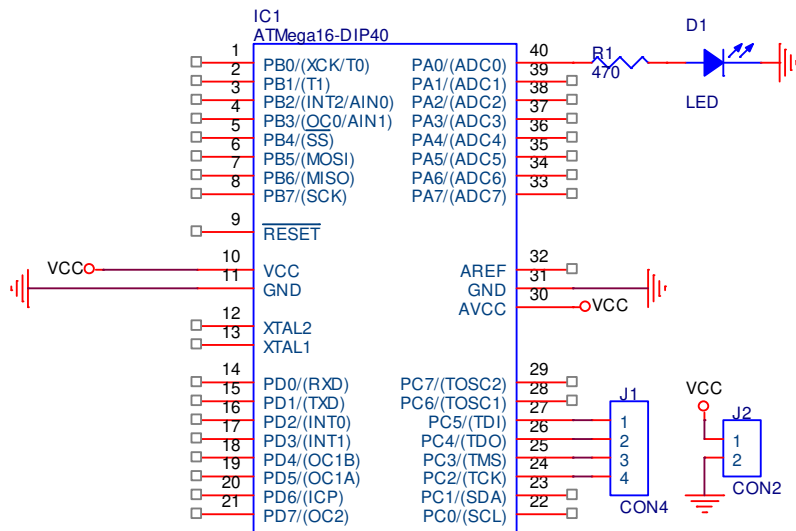


figura 5

Realizați următoarele conexiuni:

1. Conectați pinii de alimentare 10, 11, 30 și 31 ca în figură. Folosiți fire scurte portocalii și galbene, ca în figura 6.
2. Conectați LED-ul și rezistența R1 la pinul 40. Verificați valoarea rezistenței. Pentru a afla valoarea rezistorilor trebuie să știți codul culorilor. Dacă nu îl știți încă, acesta este disponibil la <http://www.leonaudio.com.au/res-code.gif> (și în multe alte locuri).
3. **NU conectați JTAG ICE la cablul USB.**
4. Conectați ATmega16A la JTAG ICE ca în figura 6. Conectorul cu 4 fire se conectează astfel încât firul alb să fie în dreptul pinului TCK. Conectorul cu 2 fire se conectează astfel încât firul alb să fie conectat la VCC. **Multa atenție la conectarea JTAG ICE!**

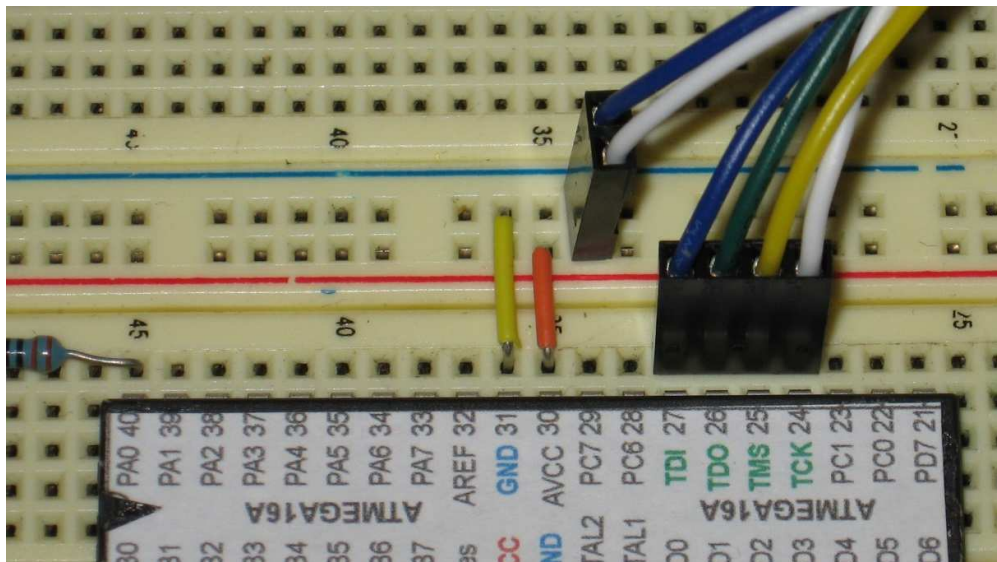


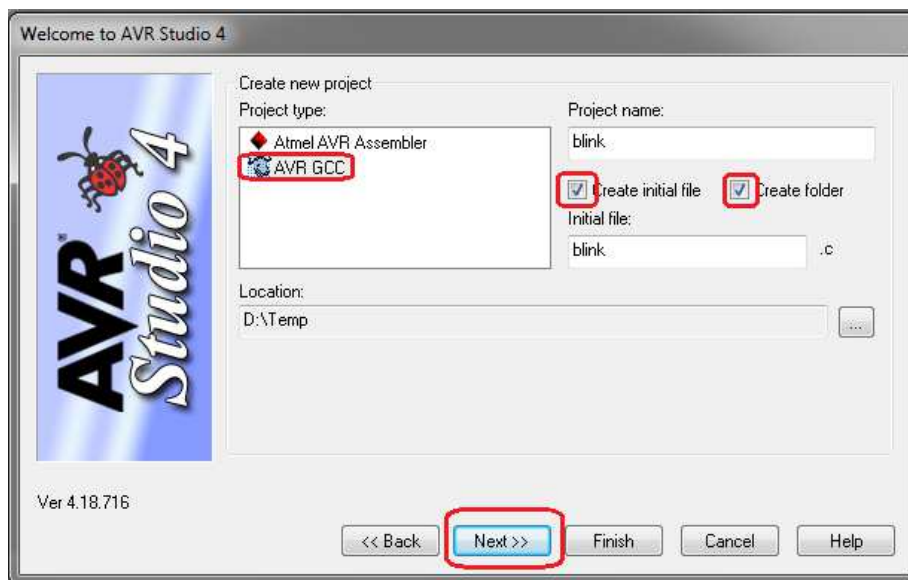
figura 6

5. Conectați firele sursei de alimentare HAMEG, secțiunea de 5V, la rândul 2H al plăcii de dezvoltare. **NU porniți sursa de alimentare! În caz contrar veți suporta eventualele pagube!**

Pasul 4: Crearea proiectului

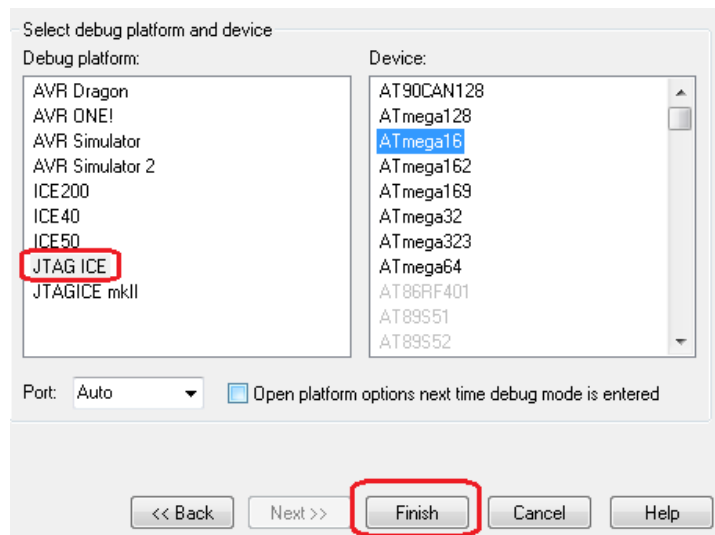


1. Se lansează în execuție **AVR Studio4**:
2. Din fereastra **Welcome to AVR Studio 4** se alege opțiunea **New Project**.
3. În următoarea fereastră,



alegeți proiect de tip AVR GCC, bifați cele două checkbox-uri ca în figura de mai sus, stabiliți locația proiectului (evident pe D:\micro\xx) și numele proiectului. În această lucrare de laborator numele proiectului va fi **blink**. În final apăsați butonul **Next**. **Fiți atenți să nu apăsați din greșeală Finish.**

4. În următoarea fereastră



alegeți drept platformă de debug **JTAG ICE**, tipul microcontrolerului **ATmega16**, lăsați opțiunea port pe **Auto** și apoi apăsați **Finish**. **Atenție:** deși opțiunea JTAG ICE este selectată, marcajul de selecție este un gri care abia se observă.

ATENȚIE:

Procedura de la subpașii 3 și 4 se urmează atât la crearea unui nou proiect cât și la deschiderea unui proiect. Întotdeauna folosiți **Next, **selecție**, **Finish**!**

Pasul 5: Crearea sursei blink.c

Optional: Pentru o editare mai ușoară a sursei, în fereastra **Tools** → **Options** → **Editor** se setează **Tab width** la 3 și se validează **Repace tab with spaces**. Deoarece editorul din AVR Studio este foarte „spartan” se recomandă editare cu **Notepad++**.

Atenție: codul sursă se scrie cu indentare. Profesorul verifică indentarea și în caz că acesta este incorectă, va trebui refăcută. Refacerea consumă timp!

Verificați că fișierul blink.c aparține proiectului. Acesta se află în **Source Files**. Dacă **Source Files** este gol, probabil ca ați uitat să bifați **Create initial file**. Adăugați acum la proiect acest fișier. Pentru aceasta, în fereastra **AVR GCC** faceți clic dreapta pe **Source Files** și din meniul contextual care va apărea selectați **Create New Source File...**

Atenție: în C numărul de biți pe care se reprezintă o variabilă **int** depinde de compilator. În cazul compilatorului din AVR Studio **variabilele de tip int se reprezintă pe 16 biți** iar cele de tip **long int** pe 32 de biți.

Structura unei aplicații micro este prezentată în prelegerea 2, capitolul 3. Orice astfel de aplicație conține o buclă infinită. Execuția codului din această buclă necesită un anumit timp. Acest fapt sugerează o metodă **aproximativă** de măsurare a trecerii timpului. De exemplu, dacă o execuție a buclei necesită 10 μs, atunci pentru 1s este nevoie 100000 de execuții. LED-ul va fi aprins la iterația 0 și va fi stins la iterația 50000 și astfel se obține un factor de umplere de 50%. Placând se la această idee, codul care face să clipească LED-ul conectat pe bitul 0 al portului A este:

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

#define P 125000L
#define DF 50L
#define TH (P*DF/100)
//125 = 1ms

int main(){
    volatile long i;

    DDRA=0xff;
    i=0;

    while(1){
        if(i==0)
            PORTA=1; //aprinde LED-ul

        if(i==TH)
            PORTA=0; //stinge LED-ul


        i++;
        if(i==P)
            i=0; //a trecut o secunda
    }
}
```

Adăugați codul de mai sus în fișierul **blink.c** creat anterior.

PORTA și DDRA sunt porturi ale procesorului din microcontroler. DDRA=0xFF; face ca portul A să fie un port de tip buffer (vezi prelegere 2, figura1). Adresele porturilor PORTA și DDRA se găsesc în documentația ATmega16 la pagina 334. Porturile vor fi tratate pe larg la curs.

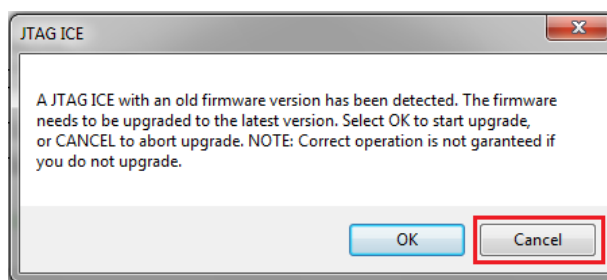
Pentru a crește claritatea programelor, adresele porturilor au primit nume simbolice. Astfel, în loc de 0x1B putem folosi PORTA. Atribuirea de nume simbolice este făcută în fișierul **io.h**, motiv pentru care acest fișier a fost inclus în codul sursă.

În rest ar trebui să fie clar ce face programul blink. Dacă există ceva ce nu ați înțeles sau ceva neclar, **chemați profesorul!**

În final compilați și linkeditați apăsând butonul **Build** .

ATENȚIE!

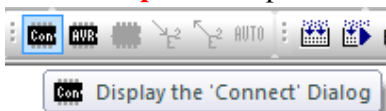
Dacă vreodată apare fereastra de mai sus, întotdeauna selectați Cancel!



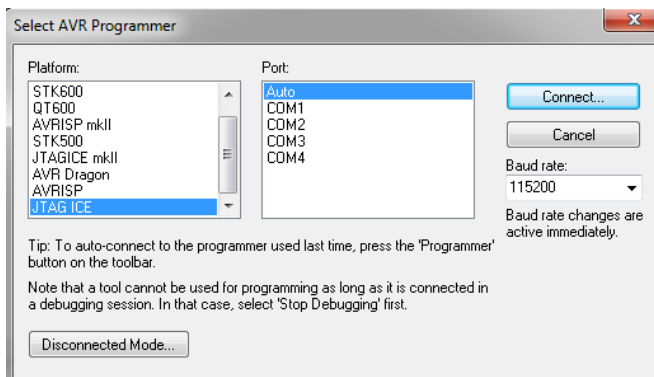
În caz contrar veți distruge codul mașină din JTAG ICE. În acest caz JTAG ICE va trebui reprogramat, operație care necesită un programator extern plus modificări hard. Consecințele vor fi din cele mai neplăcute!

Pasul 6: Execuția

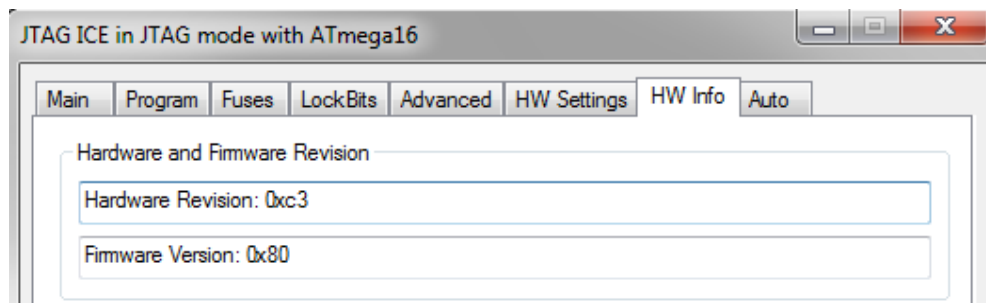
1. **Pentru validarea și alimentarea montajului, chemați profesorul! Țineți minte! Secvența de pornire este sursa HAMEG și apoi JTAG ICE.**
2. Pentru a alimenta JTAG ICE acționați comutatorul montat pe cablul USB. După ce ați alimentat JTAG ICE veți observa cum un LED roșu de pe această placă clipește. **Așteptați ca acest LED să nu mai clipească.** Apoi vom conecta AVR Studio cu μ C. Pentru aceasta apăsați butonul **Con** :



3. Faceți selecțiile din figura de mai jos dacă nu sunt deja făcute și apăsați butonul **Connect...**



4. Dacă se stabilește conexiunea cu μC , va apare fereastra prezentată mai jos:



Nu este obligatoriu să fie selectat câmpul „HW Info”, selecția inițială poate fi diferită. Valoarea câmpului „Hardware Revision” poate fi diferită față de figură.

În cazul în care conexiunea nu se stabilește închideți proiectul cu **Project → Close Project**, redeschideți-l cu **Project → Open Project** și apoi încercați să va conectați cu **Con**.

Dacă nici așa nu reușiți să vă conectați, închideți AVR Studio și apoi redeschideți-l. Încercați să vă conectați. **Dacă nici acum nu funcționează, chemați profesorul!**

5. În interiorul μC există câteva locații din memoria ROM care controlează funcționarea anumitor blocuri hardware. Cele mai importante setări sunt controlate de două locații cunoscute sub numele de **Fuses**. În fereastra JTAG ICE... apărută la pasul anterior selectați câmpul tab **Fuses**:

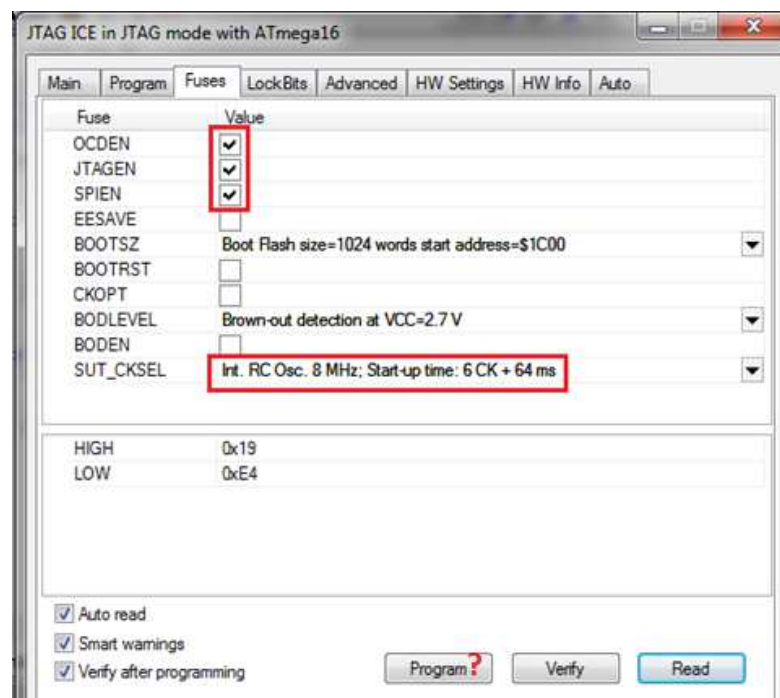
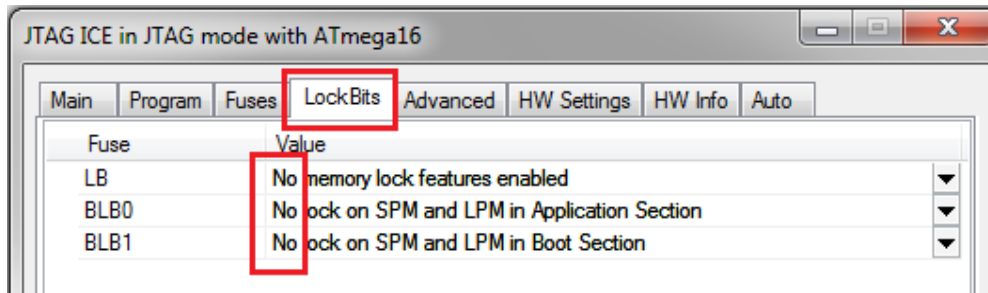


figura 7

- Opțiunea **OCDEN** – On Chip Debug ENable – trebuie să fie bifată. În caz contrar operația de debug nu este posibilă.
- Opțiunea **JTAGEN** – JTAG ENable trebuie să fie validată. În caz contrar blocul JTAG din interiorul μC este inactiv. Dacă această opțiune ar fi fost inactivă, conexiunea cu μC nu s-ar fi stabilit și fereastra de mai sus nu ar mai fi apărut. Din fabricație aceasta opțiune este activată. **Fiiți atenți sa nu o invalidați, altfel nu se va mai putea stabili conexiunea cu μC .** Reprogramarea va trebui efectuată cu un dispozitiv specializat numit programator și consumă timp. Acest timp se va scade din timpul dumneavoastră de lucru!

- uC ATmega16 dispune de circuite interne de reset și ceas. Pentru a simplifica implementarea hardware vom alege ca sursă de ceas oscilatorul intern de 8MHz. De asemenea, la punerea sub tensiune se va activa resetul intern pe durata a 64ms. Pentru a efectua aceste setări faceți selecția din figura 7 pentru opțiunea **SUT_CKSEL** - Start-Up Time & Clock SElect.
 - **Dacă** a fost necesar să schimbați vreuna din setările din figura 7, apăsați butonul **Program!**
În caz contrar NU apăsați butonul **Program**.
6. În fereastra JTAG ICE... apărută la pasul anterior selectați câmpul tab tab **LockBits**. Verificați că sunt făcute setările din figura următoare.




Dacă a fost necesar să schimbați vreuna din setările din figura de mai sus, apăsați butonul **Program!**


7. Puteți explora și celelalte câmpuri tab dar nu schimbați nici o setare.

8. **Închideți fereastra JTAG ICE....**

Apoi apăsați butonul **Start debugging** .

Ca urmare codul mașină obținut după Build va fi programat prin intermediul JTAG ICE în memoria de cod a uC. De asemenea programul se va opri la începutul lui main. Locul în care este oprit programul este marcat de o săgeată galbenă.

Pentru a executa codul mașină înscris anterior în flash apăsați butonul **Run** .

Pentru a opri execuția programului apăsați butonul **Break** . Puteți relua execuția apăsând din nou **Run**. Puteți opri și relua execuția de câte ori doriți.

Pentru a ieși din modul debug apăsați butonul **Stop Debugging** .

Pasul 7: Decuplarea JTAG ICE

În acest moment codul este în memoria flash a microcontrolerului. Aplicația trebuie să funcționeze și fără JTAG ICE. Verificați acest fapt executând următoarea secvență:

1. Dacă mai sunteți în modul Debug, părăsiți acest mod așa cum s-a explicat la pasul anterior.
2. Închideți AVR Studio
3. Opriți alimentarea JTAG ICE prin acționarea comutatorului montat pe cablul USB.
4. Opriți alimentarea montajului de la sursa HAMEG.
5. Decuplați conectori JTAG ICE (CON4 și CON2 din figura 5 și figura 6).

6. **Reporniți** alimentarea montajului de la sursa HAMEG.

Pașii 1-4 reprezintă secvența de oprire! **Respectați cu strictețe ordinea de mai sus.**

Țineți minte! Secvența de oprire este: AVR Studio, JTAG ICE și sursa HAMEG.

Secvența de oprire este inversă secvenței de pornire!

Clipește LED-ul cu conectorii (CON4 și CON2 deconectați? De ce? Indiferent dacă LED-ul clipește sau nu, **chemați profesorul!**

Reconectăm JTAG ICE:

1. Opriți alimentarea montajului de la sursa HAMEG.
2. Conectați JTAG CON4 și CON2 ca în figura 6.
3. Porniți alimentarea montajului de la sursa HAMEG.
4. Porniți alimentarea JTAG ICE prin acționarea comutatorului montat pe cablul USB.
5. Lansați în execuție AVR Studio
6. Deschideți proiectul blink.

Executați 4 experimente:

1. Modificați codul pentru ca LED-ul să clipească cu un factor de umplere **DF=10**. Observați efectul.
2. Modificați codul pentru ca LED-ul să clipească cu un factor de umplere **DF=90**. Observați efectul.
3. Modificați codul pentru a implementa clipirea cu frecvență mare. În programul blink.c micșorați perioada **P la 1250**. Acum un ciclu aprins-stins va dura 10 ms. Rulați programul pentru **DF=10** (DF=Duty Factor).
4. La fel ca la pasul 3, perioada trebuie să fie 1250. Rulați programul pentru **DF=90** (DF=Duty Factor).

Care este efectul P=1250 și DF=10? Dar P=1250 și DF=90? Mai clipește LED-ul? Când puteți răspunde, chemați profesorul.

Pasul 8: Clipire semi aprins - semi stins (simulare pentru notă mare)

Pentru $P=125000$ și $DF=50$ programul anterior face ca LED-ul să clipească la nesfârșit, o dată pe secundă. Din perioada de o secundă jumătate din timp LED-ul este aprins și jumătate de secundă stins.

În continuare se dorește ca LED-ul să clipească la fel ca mai înainte, o dată pe secundă, dar în loc să fie stins va fi aprins cu intensitate mică și în loc să fie aprins la maxim va fi aprins cu intensitate mare. Pentru a obține intensitate mică și intensitate mare vom seta $P=1250$. Valoarea lui P nu se va modifica. Pentru a obține cele două intensități se va modifica DF . LED-ul va lumina astfel:

- 0.5 secunde cu DF=10, adică va lumina slab.
- 0.5 secunde cu DF=90, adică va lumina puternic.

Modificați codul pentru ca LED-ul să lumineze la nesfârșit după cum s-a precizat mai sus.

Indicație de implementare: Așa cum s-a precizat la pasul 5 execuția codului din buclă `while(1)` necesită un timp determinat. Modificarea conținutului acestei bucle modifică automat timpul de execuție. Pentru a evita modificarea majoră a timpului de execuție nu efectuați calcule cu variabile `long int` în bucla `while(1)`. Definiți pragurile de comutare cu `define`:

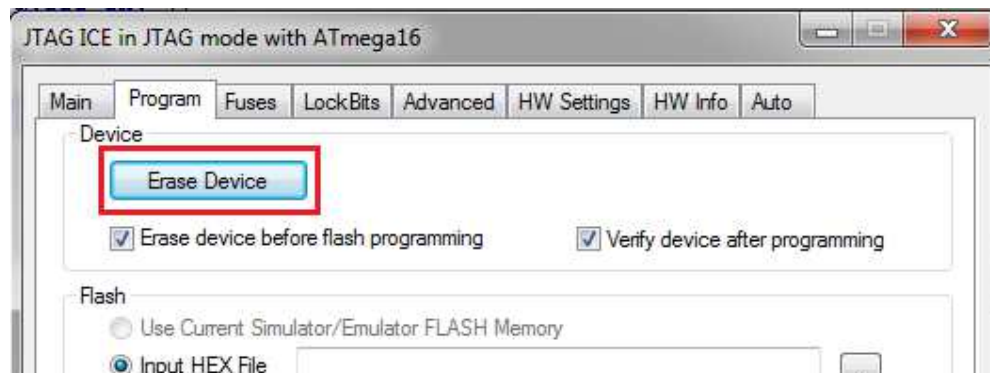
```
#define P 1250L
#define TH1 (P*10/100)
#define TH2 (P*90/100)
```

În `main` folosiți o variabilă de tip `long int` numită `th` care va lua fie valoarea lui `TH1` fie pe cea a lui `TH2`. Astfel la execuție se face doar o atribuire la o variabilă `long int`. Această atribuire modifică puțin durata de execuție a buclei.

Când funcționează, chemați profesorul!

Pasul 9: Terminarea activității de laborator

1. Mai întâi **stergeți memoria ROM**. Echipa care urmează după dumneavoastră nu trebuie să aibă un montaj funcțional; trebuie să pornească de la zero!
Dacă mai sunteți în modul Debug, părăsiți acest mod. Pentru aceasta apăsați butonul **Stop Debugging**.
2. Apăsați butonul **Con** și apoi **Connect**.
3. În fereastra **JTAG ICE in JTAG mode...** selectați câmpul tab **Program** și apoi apăsați butonul **Erase Device**, ca în figura următoare:



4. Apoi închideți AVR Studio
5. Opriți alimentarea JTAG ICE prin acționarea comutatorului montat pe cablul USB.
6. Opriți alimentarea montajului de la sursa HAMEG