

LABORATOR 8 - Implementarea Mașinilor de Stare (FSM) cu sisteme cu microprocesor

Se va implementa o parte dintr-o aplicație de tip interfon pentru 8 apartamente. Interfonul este prevăzut cu o tastatură cu 16 taste și LCD 2x16. În starea inițială interfonul permite accesul în două moduri.

1. În primul mod se apasă una din tastele 1..8 iar interfonul apelează apartamentul corespunzător cifrei apăsate. Dacă în interval de 15 secunde apartamentul apelat activează semnalul cmd, interfonul va deschide broasca electrică a ușii permițând astfel accesul. Dacă însă în intervalul de 15 secunde nu se activează **cmd**, interfonul revine în starea inițială.
2. În al doilea mod de acces, se apasă tasta ,c' și apoi se introduc trei cifre zecimale. Dacă cele 3 cifre reprezintă codul de acces predefinit, interfonul deschide broasca electrică și permite accesul iar în caz contrar revine în starea inițială.

Din descrierea modului de funcționare rezultă că interfonul se comporta diferit chiar dacă tasta apăsată este aceeași. Dacă în starea inițială se apasă de exemplu ,2', interfonul va suna la apartamentul 2, dar dacă se apasă mai întâi ,c' și apoi ,2', acest ,2' reprezintă prima cifră a codului iar interfonul o va memora, fără să sune la apartamentul 2.

Deoarece comportarea interfonului depinde nu numai de intrări ci și de contextul în care apar aceste intrări nu este posibilă o implementare de tip combinațional, precum cele din laboratoarele 3 și 4. Pentru implementările în care apare contextul, numit în literatură **stare**, implementarea este de tip automat finit determinist cu ieșiri.

Automatele finite deterministe cu ieșiri sunt mașinile secvențiale Moore și Mealy studiate la LDDC (BLPC). În engleza mașina secvențială se numește FSM – Finite State Machine. În continuare vom folosi abrevierea FSM deoarece majoritatea literaturii de specialitate este disponibilă numai în limba engleză.

Să ne aducem aminte!

- În orice moment FSM poate fi numai într-o stare. Numărul stărilor în care poate fi FSM este finit.
- FSM trece dintr-o stare în altă ca răspuns la modificarea intrărilor ce provin din exterior; trecerea dintr-o stare în alta se numește tranziție.
- FSM este definită de mulțimea stărilor în care poate fi mașina, starea sa inițială și condițiile pentru fiecare tranziție.

În continuare vom implementa aplicația interfon ca o **mașină secvențială Mealy**. Metoda este asemănătoare cu cea prezentată în laboratorul 3 pentru emularea schemelor logice combinaționale. În cazul unei mașini secvențiale Mealy ieșirile la momentul $t+1$ depind atât de intrări cât și de starea curentă de la momentul t :

$$ieșiri_{t+1} = f(intrari_t, stare_t)$$

$$stare_{t+1} = g(intrari_t, stare_t)$$

Structura hardware-software care implementează mașina secvențială se va numi în continuare SFSM (Software Finite State Machine). SFSM este prezentată în continuare:

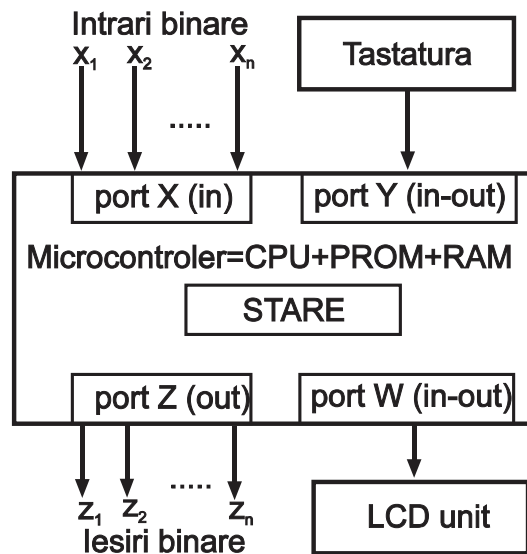


figura 1

Ansamblul hardware+software din figura 1 se comportă ca o mașină secvențială. Mașinile secvențiale învățate la cursul „Logic design” sunt sincrone, adică tranziția dintr-o stare în alta se face pe frontul activ al ceasului. În afară de mașinile secvențiale sincrone, există și mașini secvențiale asincrone, unde tranzițiile între stări au loc de îndată ce se modifică intrările, fără să existe un semnal de ceas. **SFSM de mai sus este de tip asincron.** Timpul petrecut de SFSM în fiecare stare poate fi controlat de timere sau cu contorizări ale buclei principale.

La fel ca și în cazul emulării SLC-urilor, întârzierile introduse de această metodă sunt mari. În funcție de complexitatea funcțiilor de ieșire și stare și de viteza procesorului se pot ajunge la întârzieri de ordinul milisecundelor. Acest fapt nu deranjează dacă sistemul din care provin intrările și spre care se generează ieșirile este lent. Dacă intrările provin de la contacte mecanice iar ieșirile reprezintă comenzi către motoare, timpul de răspuns al unui astfel de sistem este de ordinul zecimilor de secundă așa că întârzieri de ordinul milisecundelor nu deranjează.

Intrările SFSM pot fi:

1. **Semnale binare** citite prin intermediul porturilor de intrare. Un exemplu îl constituie intrările x_2 x_1 x_0 din laboratoarele 3 și 4. În aplicația interfon un astfel de semnal este semnalul *cmd*. Rolul acestui semnal va fi detaliat ulterior.
2. **Cuvinte** citite prin intermediul interfețelor de intrare. De exemplu, valoarea de ieșire de la un convertor analog-numeric este un astfel de cuvânt.
3. Pe baza intrărilor binare și eventual a cuvintelor citite de la interfețele IO se calculează variabilele care depind numai de intrare, ca de exemplu *kbhit* și *kbcode* pentru tastatură. Pentru interfon, *kbhit* și *kbcod* sunt intrări. Aceste variabile se vor numi **variabile de intrare**.
4. **Variabile interne** care se calculează pe baza intrărilor de tip 1-3. Un exemplu de variabilă internă îl constituie variabila/variabilele care vor memora tastele apăsate pentru codul de acces. Acestea trebuie memorate pentru a verifica dacă codul introdus este identic cu codul predefinit.
5. **Semnale de la timere** care indică scurgerea unui interval de timp. Dacă intervalele de timp nu trebuie generate cu precizie se poate folosi metoda contorizării numărului de execuții ale buclei principale în unele cazuri (rare) așteptarea în buclă *for*. În cazul aplicației interfon trebuie să se aștepte 15 de secunde activarea semnalului *cmd*.

Aplicația interfon folosește toate tipurile de intrări descrise mai sus, mai puțin cuvinte generate de interfețele de intrare

Ieșirile SFSM pot fi:

1. **Semnale binare** generate spre exterior prin intermediul porturilor de ieșire. Un exemplu îl constituie ieșirile $f_2 f_1 f_0$ din laboratorul 3 și ieșirile conectate la segmentele a-g din laboratorul 4. În aplicația interfon un astfel de semnal este semnalul care deschide ușa.
2. **Cuvinte** trimise către dispozitive de ieșire. La interfon caracterele scrise pe LCD sunt astfel de cuvinte. În cazul aplicației interfon **LCD**-ul este ieșire.
3. **Variabile interne**. La interfon în variabile interne se memorează tastele apăsate pentru cod.

La fel ca în cazul mașinile secvențiale hardware, SFSM se specifică prin intermediul diagramelor de stare. Pentru ca numele stărilor să fie cât mai scurte acestea sunt substantive în limba engleză (pentru că sunt mai scurte) sau abrevieri ale descrierii stării, tot în engleză. În figura următoare este prezentată diagrama de tranziție a stărilor pentru interfon:

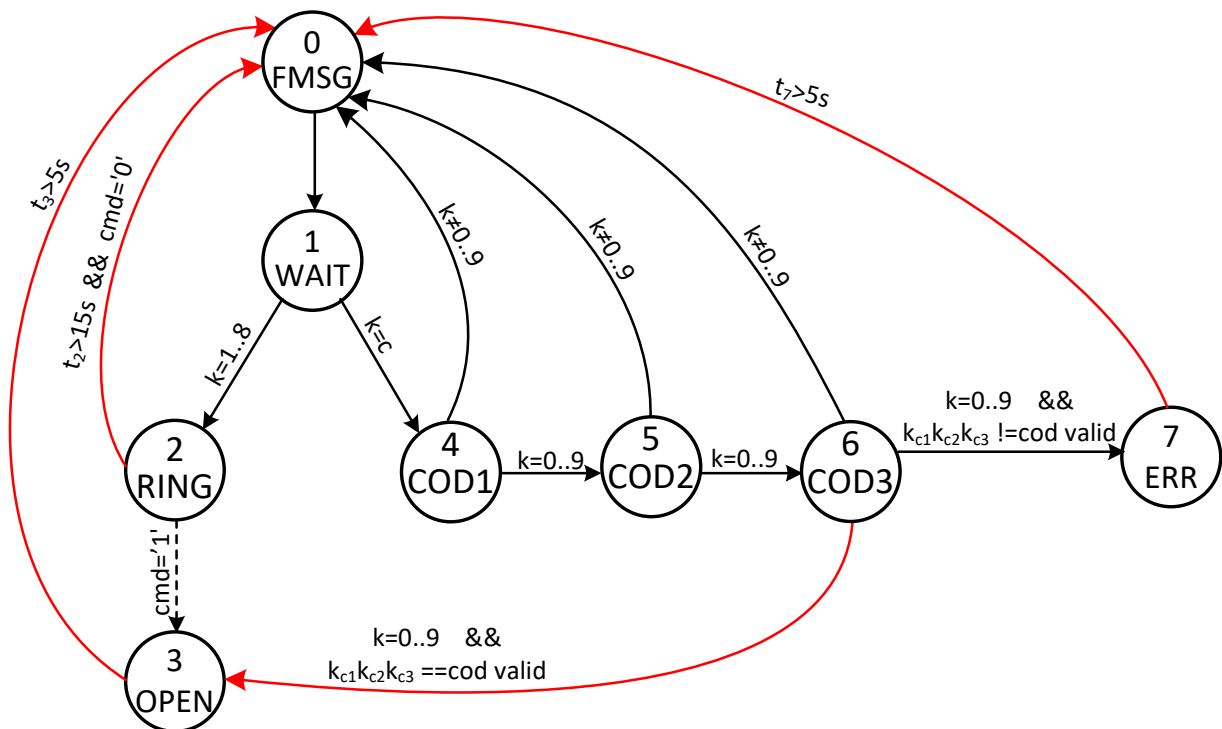


figura 2

În figura 2, k reprezintă tasta apăsată și înlocuiește condiția „kbhit == 1 && kbcode == codul_tastei_apăsate” deoarece aceasta este prea lungă și nu ar avea loc pe figură.

Timpii t_2 , t_3 și t_7 reprezintă timpul petrecut în starea RING, OPEN și ERR. În continuare se prezintă descrierea detaliată a fiecărei stări:

0. FMSG. Mesaj de întâmpinare sau primul mesaj – **F**irst **M**eSsa**G**e. Șterge LCD-ul și apoi afișează mesajul:

Pe linia 1 a LCD-ului: **Suna la 1..8,**

Pe linia 2 a LCD-ului: **C pentru cod:**

Din starea **FMSG** se trece necondiționat în starea **WAIT**. Necondiționat înseamnă că tranziția nu depinde de nici una din intrări. Într-o diagramă de stare o tranziție se reprezintă printr-o săgeată pe care se scrie condiția care trebuie îndeplinită pentru a se face respectiva tranziție. O săgeată pentru care condiția lipsește reprezintă o tranziție necondiționată.

Necesitatea stării FMSG se va explica puțin mai târziu.

1. WAIT. Interfonul **așteaptă** introducerea unui caracter de la tastatură.

- a. Dacă s-a apăsă o cifră între 1 și 8 se va șterge LCD-ul, se va afișa mesajul „**Sun la *a***”, unde *a* este numărul apartamentului introdus anterior și se va trece în starea **RING**. Tranziția în starea RING se va abrevia cu „→ RING”. În general trecerea în starea **stare** se va abrevia cu → **stare**.
- b. Dacă se apasă ,C’ se șterge LCD-ul, se afișează „**Cod=**” și → **COD1**.
- c. Dacă se apasă ceva diferit de 1..8 sau ,C’, nu se afișează nimic și se rămâne în WAIT pe durată nedeterminată; timpul nu contează în starea WAIT.

Pe diagrama de stare din figura 2 această tranziție ar trebui figurată ca o tranziție din starea WAIT în starea WAIT. Pentru a nu încălca diagrama de stare astfel de tranziții nu sunt figurate pentru nici o stare. Pe diagramă sunt figurate numai tranzițiile către o stare diferită de starea prezentă. Mai mult, nici în această descriere nu va mai apare condiția pentru a rămâne în aceeași stare. **Implicit, dacă nu este îndeplinită nici o condiție de tranziție într-o stare diferită de starea prezentă, se rămâne în starea prezentă.**

2. RING. Interfonul va **sună** la apartamentul corespunzător cifrei apăsate în starea WAIT. Ar trebui ca interfonul să trimită către apartamentul ,*a*’ semnalul care face ca postul din apartamentul ,*a*’ să sune.

- a. Dacă din apartamentul ,*a*’ se activează semnalul de acceptare *cmd* se va șterge LCD-ul, se va afișează mesajul „**Deschis!**”, iar apoi → **OPEN**. Pentru nota 5 această funcționalitate nu se va implementa.
- b. Dacă după 15 secunde *cmd*=’0’, → **FMSG**. Intervalul de 15 secunde este nerealist; 1 minut ar fi mai adecvat, dar în faza de dezvoltare a aplicației trebuie făcute multe teste și nu ne permitem să așteptăm 1 minut. Pentru nota 5 întotdeauna *cmd*=’0’

3. OPEN. Se activează semnalul care **deschide** broasca.

- a. Dacă timpul petrecut în starea OPEN $t_3 > 5s$ → **FMSG**.

4. COD1 – prima cifră a codului.

- a. Dacă tasta k este o cifră zecimală aceasta se va memora, se va afișează caracterul ,*’ și apoi → **COD2**.
- b. Dacă se apasă o tastă care nu este cifră zecimală, caracterul corespunzător nu se va afișează și → **FMSG**.

5. COD2 – a doua cifră a codului.

- a. Dacă tasta k este o cifră zecimală aceasta se va memora, se va afișează caracterul ,*’ și apoi → **COD3**.
- b. Dacă se apasă o tastă care nu este cifră zecimală, caracterul corespunzător nu se va afișează și → **FMSG**.

6. COD3 – a treia cifră a codului.

- a. Dacă tasta k este o cifră zecimală se va afișează ,*’. Apoi se va șterge LCD-ul și se va verifica dacă cele trei cifre reprezintă codul de acces.
 - i. Dacă DA, se afișează **Deschis!**, iar apoi → **OPEN**
 - ii. Dacă NU, se afișează „**Cod invalid**”, iar apoi → **ERR**

- b. Dacă se apasă o tastă care nu este cifră zecimală, caracterul corespunzător nu se va afișa și
→ **FMSG**.

7. ERR – eroare.

- a. Dacă timpul petrecut în starea ERR $t_7 > 5s$ → **FMSG**.

Implementarea tipică a SFSM Mealy se face cu switch, după cum urmează:

stare = stare₀;

while(1)

Citește intrările.

Dacă este necesar, calculează variabilele de intrare.

Switch stare

Case stare₀:

În funcție de intrări calculează ieșirile și starea următoare
Break;

Case stare₁:

În funcție de intrări calculează ieșirile și starea următoare
Break;

.
.

Case stare_n:

În funcție de intrări calculează ieșirile și starea următoare
Break;

end switch

end while(1).

SCOPUL LUCRĂRII

Se va implementa o parte dintr-o aplicație de tip interfon pentru 8 apartamente. Intrarea se primește de la tastatura implementată în laboratorul precedent iar ieșirea se afișează prin intermediul LCD. Aplicația este controlată de o SFSM care funcționează conform diagramei din figura 2 și a descrierii aferente.

Desfășurarea lucrării

Pasul 1: Crearea proiectului

- **Creați** proiectul **interfon**. La crearea proiectului **bifați** opțiunea „Create initial file”.
- Copiați **FSM.c** din arhiva acestei lucrări de laborator în folderul proiectului și apoi **adăugați-l** la proiect.
- Ștergeți din proiect fișierul **interfon.c** (care este gol).
- **Copiați** fișierele **IOfn.c** și **defs.h** din proiectul kbd în folderul proiectului **interfon**.
- **Adăugați** la proiect fișierele **IOfn.c** și **defs.h** din folderul proiectului **interfon**.

- **Completați FSM.c** pentru ca mașina de stare să funcționeze conform descrierii din figura 2. Pe moment semnalul *cmd* nu există iar arcul reprezentat cu linie întreruptă în figura 2 nu se va implementa.

Indicații de implementare:

1. Codul de acces predefinit este 123. Acest cod va fi memorat ca șir de caractere.
2. În mod normal întârzierile se implementează cu *timere*, fie prin contorizarea numărului de execuții ale buclei *while(1)*. În cazul interfonului, **pentru nota 5**, nu trebuie monitorizată și altă intrare în afară de tastatură deoarece tastatura este singura intrare a sistemului. Din acest motiv, plus faptul că întârzierile nu trebuie să fie foarte precise, se admite implementarea întârzierilor cu bucle *soft*. Folosiți funcția *wait* din *iofn.c* Apelul

```
wait (250000UL);
```

asigură o întârziere de aproximativ o secundă.

Alimentați și testați. Dacă funcționează, chemați profesorul pentru validare.

Dacă ați ajuns aici aveți nota 5!

Pasul 2: ...

Se dorește ca în starea RING să apara următoarea secvență de mesaje:

```
Sun la ,a'
Sun la ,a'.
Sun la ,a'..
Sun la ,a'...
Sun la ,a'
Sun la ,a'.
Sun la ,a'..
Sun la ,a'...
```

...

unde *,a'* este numărul apartamentului preluat în starea WAIT. Fiecare mesaj va fi afișat aproximativ o secundă. Mesajele vor fi afișate în ordinea definită mai sus, începând cu linia 1, coloana 1, până când trec cele 15 secunde.

Funcționalitatea implementată anterior trebuie să se păstreze.

Testați. Dacă funcționează, chemați profesorul pentru validare.

Dacă ați ajuns aici aveți 1 sau 2 puncte, în funcție de calitatea implementării!

Pasul 3: Semnalul *cmd*

Funcționalitatea implementată anterior trebuie să se păstreze.

Se dorește ca din starea RING să se treacă în starea OPEN, conform săgeții punctate din figura 2. Această tranziție va avea loc dacă semnalul *cmd* ce se presupune că vine de la apartamentul apelat este activ. Tranziția va avea loc **imediat** ce semnalul devine activ. Pentru tranziție imediată se recomandă așteptarea prin metoda contorizării execuțiilor buclei principale

Semnalul *cmd* se va citi prin intermediul unui bit nefolosit din portul B. Reamintim că biții 5, 6 și 7 din portul B sunt folosiți de LCD. Pentru a genera acest semnal, adăugați un comutator SPST, așa cum s-a făcut în laboratorul 3.

Testați. Dacă funcționează, chemați profesorul pentru validare.

Dacă ați ajuns aici aveți 1 sau 2 puncte, în funcție de calitatea implementării!

Pasul 4: Acționare broască

Se dorește ca în starea OPEN să se activeze către exterior un semnal. Acest semnal va fi activ numai în această stare. Acest semnal va fi folosit pentru a acționa broasca electrică și se va genera folosind un pin neutilizat din portul B. Folosiți acest semnal pentru a aprinde un LED.

Funcționalitatea implementată anterior trebuie să se păstreze.

Testați. Dacă funcționează, chemați profesorul pentru validare. Dacă ați ajuns aici aveți un punct!

Pasul 5: Deconectare

La terminare executați următoarele operații:

1. Se oprește execuția programului cu **Break**,
2. Se oprește depanatorul cu **Stop**
3. Se apasă **Con** și apoi se șterge memoria de program cu **Erase**
4. Se închide AVR Studio.
5. Se oprește alimentarea JTAG ICE.
6. Se oprește sursa Hameg
7. Se desface montajul, numai componentele adăugate în acest laborator.