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

Se va implementa o parte dintr-o aplicaţie de tip interfon pentru 8 apartamente. Interfonul este prevăzut cu o tastatură cu 16 taste şi LCD 2x16. În starea iniţială 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 uşii permiţând astfel accesul. Dacă însă în intervalul de 15 secunde nu se activează cmd, interfonul revine în starea iniţială.
2. În al doilea mod de acces, se apasă tasta ‚c' si apoi se introduc trei cifre zecimale. Dacă cele 3 cifre reprezintă codul de acces predefinit, interfonul deschide broasca electrică şi permite accesul iar în caz contrar revine în starea iniţială.

Din descrierea modului de funcţionare rezultă că interfonul se comporta diferit chiar dacă tasta apăsată este aceeaşi. Dacă în starea iniţială se apasă de exemplu ‚2', interfonul va suna la apartamentul 2, dar dacă se apasă mai întâi ‚c' şi apoi ‚2', acest ‚2' reprezintă prima cifra a codul iar interfonul o va memora, fără să sune la apartamentul 2.

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

Automatele finite deterministe cu ieşiri sunt maşinile secvenţiale Moore şi Mealy studiate la LDDC (BLPC). În engleza maşina secvenţială se numește FSM – Finite State Machine. În continuare vom folosi abrevierea FSM deoarece majoritatea literaturii de specialitate este disponibila numai în limba engleză.

Să ne aducem aminte!

- În orice moment FSM poate fi numai într-o stare. Numărul stările 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 numeşte tranziţie.
- FSM este definită de mulţimea stărilor în care poate fi maşina, starea sa iniţială şi condiţiile pentru fiecare tranziţie.

În continuare vom implementa aplicaţia interfon ca o **maşină secvenţială Mealy**. Metoda este asemănătoare cu cea prezentată în laboratorul 3 pentru emularea schemelor logice combinaţionale. În cazul unei maşini secvenţiale, ieşirile la momentul t+1 depind atât de intrări cât şi de starea curentă de la momentul t:

$$\text{ieşiri}_{t+1} = f(\text{intrari}_t, \text{stare}_t)$$

$$\text{stare}_{t+1} = g(\text{intrari}_t, \text{stare}_t)$$

Structura hardware-software care implementează maşina secvenţială se va numi în continuare SFSM (**S**oftware **F**inite **S**tate **M**achine). SFSM este prezentată în continuare:

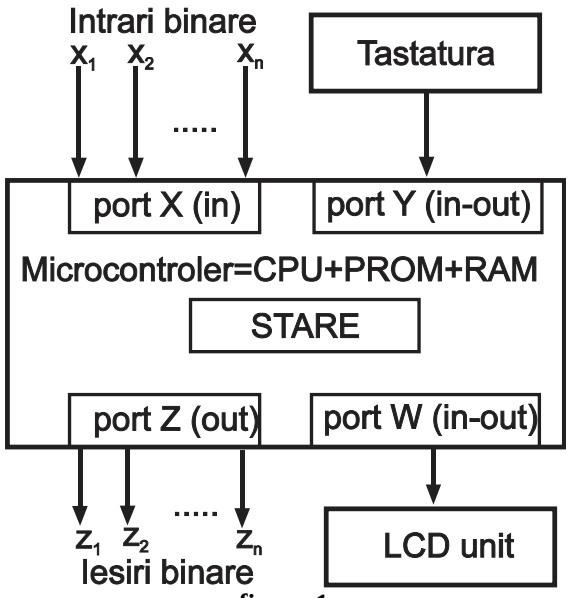

figura 1

Ansamblul hardware+software din figura 1 se comportă ca o maşină secvenţială. Maşinile secvenţiale învăţate la cursul „Logic design” sunt sincrone, adică tranziţia dint-o stare în alta se face pe frontul activ al ceasului. În afară de maşinile secvenţiale sincrone, există şi maşini secvenţiale asincrone, unde tranziţiile 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 ala buclei principale.

La fel ca şi în cazul emulării SLC-urilor, întârzierile introduse de această metodă sunt mari. În funcţie de complexitatea funcţiilor de ieşire şi stare şi de viteza procesorului se pot ajunge la întârzieri de ordinul milisecundelor. Acest fapt nu deranjează dacă sistemul din care provin intrările şi spre care se generează ieşirile este lent. Dacă intrările provin de la contacte mecanice iar ieşirile reprezintă comenzi către motoare, timpul de răspuns al unui astfel de sistem este de ordinul zecimilor de secundă aşa 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 şi 4. În aplicaţia interfon un astfel de semnal este semnalul *cmd*. Rolul acestui semnal va fi detaliat ulterior.
2. **Cuvinte** citite prin intermediul interfeţelor de intrare. De exemplu, valoarea de ieşire de la un convertor analog-numeric este un astfel de cuvânt.
3. Pe baza intrărilor binare şi eventual a cuvintelor citite de la interfeţele IO se calculează variabilele care depind numai de intrare, ca de exemplu *kbhit* şi *kbcode* pentru tastatură. Pentru interfon, *kbhit* şi *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 execuţii ale buclei principale în unele cazuri (rare) aşteptarea în buclă *for*. În cazul aplicaţiei interfon trebuie să se aştepte 15 de secunde activarea semnalului *cmd*.

Aplicaţia interfon foloseşte toate tipurile de intrări descrise mai sus, mai puţin cuvinte generate de interfețele de intrare

Ieşirile SFSM pot fi:

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

La fel ca în cazul mașinile secvenţiale 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 tranziţie a stărilor pentru interfon:

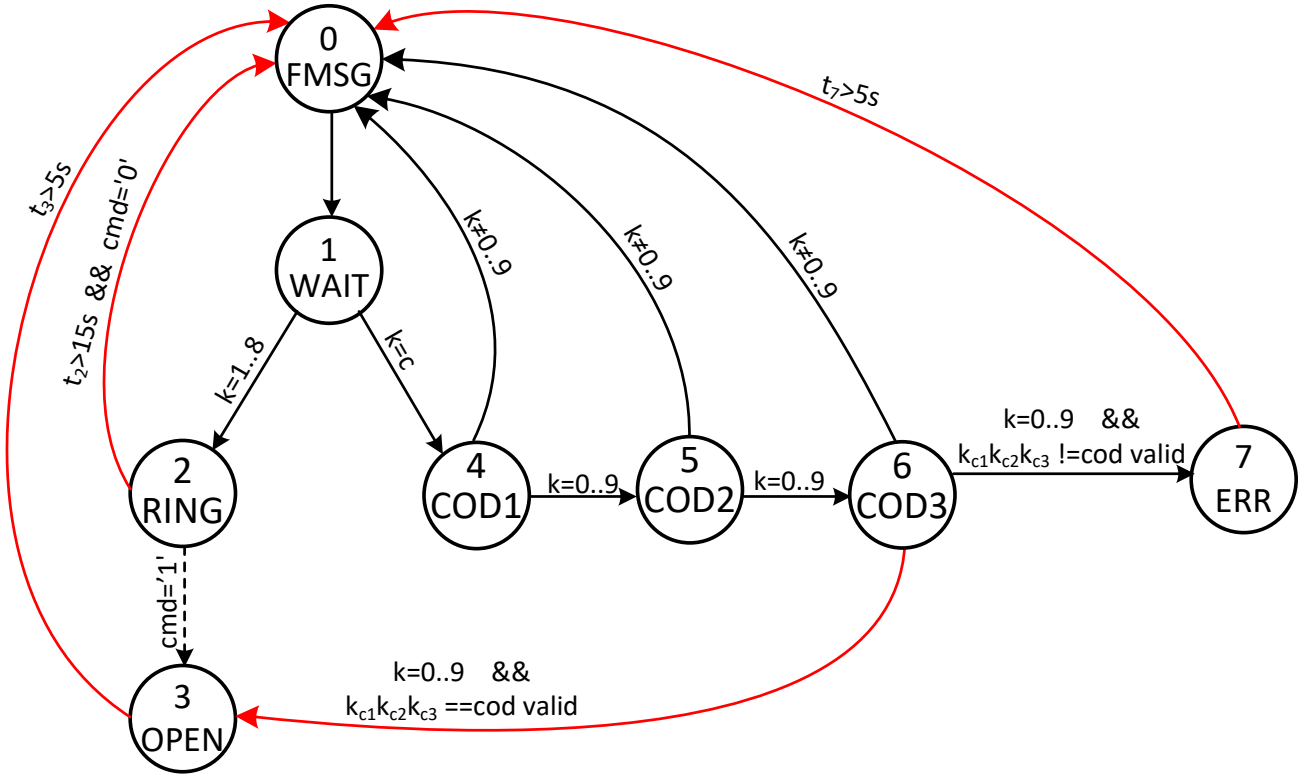

figura 2

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

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

0. FMSG. Mesaj de întâmpinare sau primul mesaj – **F**irst **M**e**S**sa**G**e. Şterge LCD-ul şi apoi afişează 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 necondiţionat în starea **WAIT**. Necondiţionat înseamnă că tranziţia nu depinde de nici una din intrări. Într-o diagramă de stare o tranziție se reprezintă printr-o săgeată pe

care se scrie condiţia care trebuie îndeplinită pentru a se face respectiva tranziție. O săgeată pentru care condiția lipseşte reprezintă o tranziție necondiționată.

Necesitatea stării FMSG se va explica puţin mai târziu.

1. **WAIT**. Interfonul **aşteaptă** introducerea unui caracter de la tastatură.
   a. Dacă s-a apăsat o cifră între 1 şi 8 se va şterge LCD-ul, se va afişa mesajul „**Sun la *a***”, unde ‚*a*’ este numărul apartamentului introdus anterior şi se va trece în starea **RING**. Tranziţia 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 şterge LCD-ul, se afișează „**Cod=**” şi **→ COD1**.
   c. Dacă se apasă ceva diferit de 1..8 sau ‚C’, nu se afişa nimic şi se rămâne în WAIT pe durată nedeterminată; timpul nu contează în starea WAIT.

      Pe diagrama de stare din figura 2 această tranziţie ar trebui figurată ca o tranziție din starea WAIT în starea WAIT. Pentru a nu încărca diagrama de stare astfel de tranziţii 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 condiţia pentru a rămâne în aceeaşi stare. **Implicit, dacă nu este îndeplinită nici o condiţie de tranziţie într-o stare diferită de starea prezentă, se rămâne în starea prezentă.**
2. **RING**. Interfonul va **suna** 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 şterge LCD-ul, se va afişă 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 şi nu ne permitem să aşteptăm 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 cifra a codului.
   a. Dacă tasta k este o cifra zecimală aceasta se va memora, se va afişa caracterul ‚*’ şi apoi **→ COD2**.
   b. Dacă se apasă o tastă care nu este cifră zecimală, caracterul corespunzător nu se va afişa şi **→ FMSG**.
5. **COD2** – a doua cifră a codului.
   a. Dacă tasta k este o cifra zecimală aceasta se va memora, se va afişa caracterul ‚*’ şi apoi **→ COD3**.
   b. Dacă se apasă o tastă care nu este cifră zecimală, caracterul corespunzător nu se va afişa şi **→ FMSG**.
6. **COD3** – a treia cifră a codului.
   a. Dacă tasta k este o cifra zecimală se va afişa ‚*’. Apoi se va şterge LCD-ul şi 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 se face cu switch, după cum urmează:

stare = starea$_0$;

while(1)

Citeşte intrările.

Dacă este necesar, calculează variabilele de intrare.

Switch stare

Case starea$_0$:

În funcție de intrări calculează ieşirile şi starea următoare

Break;

Case stare$_1$:

În funcție de intrări calculează ieşirile şi starea următoare

Break;

.

.

Case stare$_n$:

În funcție de intrări calculează ieşirile şi starea următoare

Break;

End switch

End while(1).

SCOPUL LUCRĂRII

Se va implementa o parte dintr-o aplicaţie de tip interfon pentru 8 apartamente. Intrarea se primeşte de la tastatura implementată în laboratorul precedent iar ieşirea se afişează prin intermediul LCD. Aplicaţia este controlată de o SFSM care funcţionează conform diagramei din figura 2 şi a descrierii aferente.

Desfăşurarea lucrării

Pasul 1: Crearea proiectului

- Creaţi proiectul interfon. La crearea proiectului debifaţi opțiunea „Create initial file”.
- Copiaţi FSM.c din arhiva acestei lucrării de laborator în folderul proiectului şi apoi adăugaţi-l la proiect.
- Copiaţi fişierul IOfn.c din proiectul kbd în folderul noului proiect (interfon).
- Adăugaţi la proiect fişierul IOfn.c din folderul proiectul interfon.

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

Indicaţii de implementare:

1. Codul de acces predefinit este 123. Acest cod va fi memorat ca şir de caractere.
2. În mod normal întârzierile se implementează cu timere, fie ca contorizând numărul de execuţii ale buclei while(1). În cazul interfonului, pentru nota 5, nu trebuie monitorizată şi 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. Folosiţi funcţia *wait* din iofn.c Apelul

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

   asigură o întârziere de aproximativ o secundă.

Alimentaţi şi testaţi. Dacă funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici aveţi nota 5!

Pasul 2: …

Se doreşte ca în starea RING să apăra următoarea secvenţă 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 afişat aproximativ o secundă. Mesajele vor fi afişate în ordinea definită mai sus, până când trec cele 15 secunde.

Funcţionalitatea implementată anterior trebuie să se păstreze.

Testaţi. Dacă funcţionează, chemaţi profesorul pentru validare.
Dacă aţi ajuns aici aveţi 1 sau 2 puncte, în funcţie de calitatea implementării!

Pasul 3: Semnalul cmd

Funcţionalitatea implementată anterior trebuie să se păstreze.

Se doreşte ca din starea RING să se treacă în starea OPEN, conform săgeţii punctate din figura 2. Această tranziţie 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 tranziţie imediată se recomandă aşteptarea prin metoda contorizării execuţiilor buclei principale

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

Testaţi. Dacă funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici aveţi 1 sau 2 puncte, în funcţie de calitatea implementării!

Pasul 4: Acţionare broască

Se doreşte 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 acţiona broasca electrică şi se va genera folosind un pin neutilizat din portul B. Folosiţi acest semnal pentru a aprinde un LED.

Funcţionalitatea implementată anterior trebuie să se păstreze.

Testaţi. Dacă funcţionează, chemaţi profesorul pentru validare. Dacă aţi ajuns aici aveţi un punct!

Pasul 5: Deconectare

La terminare executați următoarele operaţii:

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