|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SW7/ | SW6/ | SW5/ | SW4/ | SW3/ | SW2/ | SW1/ | SW0/ |
| Start Introducere cod PIN | Stop introducere cod PIN | Reset  Introd.  Parola  Sau  Introd ID | introd. ID | Introducere parola  sau ID  SAU  Interogare  sold | Introducere parola  sau ID  SAU  Retragere numerar | Introducere parola  sau ID  SAU  Emitere chitanta | Introducere  Parola  sau ID  SAU  Retragere  card |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 | LED0 |
| Pregatit pentru introducere pin | Eroare introd. PIN sau  Resetare  PIN sau ID | neutilizat | neutilizat | Interogare sold | Ridicare bancnote | Emitere  chitanta | Retragere card |

LED-urile se aprind pe 1(LED5,LED 4 raman neutilizate in program);

SWITCH-urile servesc ca butoane fara revenire si sunt active pe 0;

Se va considera configuratia urmatoare:

PORTD- Intrare

PORTB-Iesire

TIMER0- programat pe intreruperi periodice la 20ms

Când starea S este 0 sau 2, switch-urile funcționează precum o "tastatură", așa cum se arată în tabelul anterior.

Declarați variabile globale:

x, S, Q, in, out de tip întreg și caractere

Declarați un vector de pointeri TAB1 de 81 de elemente

Declarați tabele de semnale relevante A0 până A80

Funcția ID():

Dacă in este 0, atunci Q devine 0

Altfel, dacă in este 0x01, atunci Q devine 4

Altfel, dacă in este 0x02, atunci Q devine 8

...

Repetați pentru fiecare valoare a lui in până la 0x26

(cifrele hexazecimale corespunzătoare intrării SW7-SW0)

Funcția cls():

Inițializați variabilele i și adr cu 0

Inițializați ready cu 0

Cat timp ready este fals:

Dacă (~in) este egal cu valoarea de la adresa adr+i, atunci Q devine valoarea de la adr+i+1 și setați ready la 1(Se foloseste ~ pentru a folosi butoanele active in 0)

Altfel, dacă valoarea de la adr+i este 16, setați ready la 1

Altfel, incrementați i cu 2

Funcția de întrerupere a depășirii timerului 0 (timer0\_ovf\_isr):

Reinițializați valoarea Timerului 0

Citiți valoarea de intrare PIND

În funcție de starea S:

Dacă S este 0:

S devine 1 și apelați funcția ID()

Altfel, dacă S este 1:

Dacă in este 0x7F, atunci S devine 2 și out devine 0x80

Altfel, dacă in este 0xDF, atunci S devine 6

Altfel, dacă S este 2:

Apelați funcția cls()

Dacă in este 0xBF, atunci S devine 3

Altfel, dacă S este 3:

Dacă Q este 80:

Dacă in este 0xFB, atunci out devine 0x04 și S devine 4

Altfel, dacă in este 0xF7, atunci out devine 0x08 și S devine 5

Altfel:

Pentru x de la 30 la 0 (în ordine descrescătoare):

Aprind si sting LED-ul 6

Setează S la 6

Altfel, dacă S este 4:

Dacă in este 0xFD, atunci out devine 0x06, altfel out devine 0x04

S devine 6

Altfel, dacă S este 5:

Dacă in este 0xFD, atunci out devine 0x0A, altfel out devine 0x08

S devine 6

Altfel, dacă S este 6:

out devine 0x01 și S devine 7

Altfel, dacă S este 7:

S devine 0, out devine 0x00, Q devine 0

Setează PORTB la valoarea out

char \*TAB1[81]; //tabela de adrese

// tabelele de semnale relevante

char A0[] = {7, 1, 16, 0}; //prima cifra din PIN din ID1

char A1[] = {3, 2, 16, 1}; // a doua cifra din PIN din ID1

char A2[] = {9, 3, 16, 2}; // a treia cifra din PIN din ID1

char A3[] = {1, 80, 16, 3}; //a patra cifra din PIN din ID1

Am optat pentru un tabel de vectori care conțin detalii legate de PIN-ul respectiv fiecărui ID.

In funcție de variabila Q care se alege prin intermediul funcției ID()

void ID(void){

if (in == 0x00) {

Q = 0;

} else if (in == 0x01) {

Q = 4;

} …

In această funcție valorile în hexa reprezintă ID-ul introdus

Codificarea acestui ID s-a realizat în felul următor :

S-au folosit 6 biti pentru aceasta actiune , fiacare bit reprezentand urmatoarele:

|  |  |  |
| --- | --- | --- |
| Bitul 6 | Biții 4-5: | Biții 3-0 |
| Național – 0 | ING – 00 | 000 – 011 |
|  | BCR – 01 | 000 – 110 |
|  | BT – 10 | 000 – 010 |
| Internațional – 1 | Revolut – 00 | 000 – 111 |

void cls(void)

{

char i;

char \*adr;

char ready;

adr=TAB1[Q];

i=0;

ready=0;

while (!ready)

{

if (~in==\*(adr+i)) {Q=\*(adr+i+1); ready=1;}

else if (\*(adr+i)==16) ready=1;

else i=i+2;

}

}

Funcția cls() este folosita pentru verificarea ficarei cifre din PIN ;Terminatorul a fost ales cu valoarea 16 , valoare la care nu se poate ajunge tinand cont de faptul ca parola continue doar cifre de la 0 la 9

In functia de intrerupere sunt tratate toate starile si cazurile procesului secvential (Se aprind LED-urile cerute si sunt apelate functiile prezentate anterior)

for (x=30; x > 0; x--) {

if(x>25){

PORTB = 0x41;}

if(x>20){

PORTB = 0x01;}

if(x>15){

PORTB = 0x41;}

if(x>10){

PORTB = 0x01;}

if(x>5){

PORTB=0x41;}

if(x>0){

PORTB=0x01;}

Aceasta instructiune este folosita pentru aprinderea intermitenta a LED- ului 6, tinand cont de TIMER0- programat pe intreruperi periodice la 20ms.

INTRODUCERE :

Scopul proiectului este implementarea din punct de vedere software a unui bancomat, care are la dispozitie 8 butoane(switch-uri) si 8 leduri care reprezinta actiuni posibile pe care le poti face prin intermediul acestuia.Exemple de astfel de actiuni sunt urmatoarele: retragere numerar, interogare sold, introducere cod PIN si introducere ID.

Acest model a fost implementat in limbaj C in programul CodeVisionAVR si testat in compilatorul AVR Studio 4 pe microcontrolerul ATmega164P.

In vederea conceperii acestui proiect am folosit materialele primite la curs si laborator, dar si resurse gasite pe internet, cu precadere forumurile destinate AVR.

Implementarea functiei de adaugare si verificare a PIN ului a fost realizata printr-un circuit secvential cu 81 de subvectori (cate patru pentru fiecare dintre cele 20 de ID-uri).A fost aleasa aceasta modalitate pentru a putea corela fiecare ID cu o parola.

O alta varianta pentru aceasta implementare, care nu functioneaza intotdeauna, dar este mai scurta este aceea in care se foloseau 5 adrese lungi in loc de 81 ,iar parcurgerea starilor Q ar fi fost sepuita in felul urmator: Se testa ID-ul , iar dupa apasarea butonului responsabil pentru introducerea parolei, tabelul de vectori ar fi fost parcurs de la coada la cap pentru testarea codului PIN. Problema majora pe care o ridica aceasta implementare era ca dupa validarea ID-ului putea fi introdus orice cod PIN care ar fi fost ulterior validat.

Pentru aprinderea intermitenta a unui LED am optat pentru o bucla for() care sa ruleze pentru aproximativ 3 secunde tinand cont de TIMER-ul programat pe intreruperi periodice de 20 ms. Nu am folosit functiile predefinite din program (delay() ) pentru ca am intampinat probleme la rularea in AVR Studio 4.

TESTARE 1

Pentru testare am ales sa introduc ID-ul clientului nr. 1 de la BT, dar gresind codul PIN pentru a demonstra functionalitatea instructiunii de aprindere interminenta a LED-ului

Prima data am pus breakpoint-ul si cursorul pe functia de intrerupere

A screenshot of a computer code

Description automatically generated

Imediat dupa incarcare, programul asteapta introducerea ID-ului(00010001)

A screenshot of a computer

Description automatically generated

Starea S devine 1, iar Q isi ia valoarea specifica locului in care este salvata parola

Dupa apasarea SW7/ S devine 2, iar LED7(responsabil cu afisarea faptului ca atm-ul asteapta o parola) se aprinde

A screenshot of a computer

Description automatically generated

Dupa introducerea gresita se apasa pe SW6/ ,S devine 3, iar Q nu ajunge la 80(conditie pentru ca avea access la functionalitatea ATM-ului.

Apoi programul va trebui rulat cu autostep pentru a vizualiza LED ul aprins si stins.

S trece la valoarea 6, iar prin rularea la cursor se trece in starea 7 unde se va aprinde LED-ul responsabil pentru retragerea cardului.

TESTARE 2

Pentru testare am ales sa introduc ID-ul clientului nr. 5 de la Revolut, acesta dorind sa retraga numerar.

ID(00100100)

In mod asemanator programul asteapta un ID de utilizator.

S=1,Q=68;

Pentru introducerea parolei se va verifica tabelul de adrese astfel:

Se verifica Q si se introduc cifrele din vectorii A68,A69,A70,A71;

A screenshot of a computer screen

Description automatically generated

La sfarsitul adaugarii ultimei cifre Q devine 80 pregatit astfel sa fie testat in starea urmatoare

A screenshot of a computer

Description automatically generated

In starea 3 se apasa pe SW2/ pentru a efectua operatiunea de retragere numerar

A screenshot of a computer

Description automatically generated

LED-ul 2 se aprinde marcand faptul ca urmeaza o retragere de cash

In starea 4 avem optiunea de a cere chitanta sau a nu cere chitanta, in acest caz am optat pentru cererea chitantei apasand SW1/

A screenshot of a computer

Description automatically generated

Astfel se aprinde si LED-ul 1 corscpunzator emiterii chitantei

Din starea 6 la urmatoarea rulare se va intra in starea 7, marcandu-se cu LED-ul 0 faptul ca, cardul a fost extras din bancomat.Ulterior toate valorile revin la starea initiala printr-o ultima rulare ,iar ATM-ul asteapta introducerea unui alt ID de utilizator