Stéphane ARCHER

Fire Alarm System

March 19, 2014

1. Introduction
2. Visual Studio Source Code
3. Microcontroller Source Code
4. Structured Design Approach
5. Testing
6. Conclusion

**Introduction:**

The system is to monitor 9 separate alarm circuits via the 68HC11 parallel ports that can be split into three fire zones (3 trips per zone). Each zone must be capable of being enabled or disabled when the alarm is set via a menu driven interface on the system terminal. If the alarm is activated then the system should activate a single bit of a port output and display the alarm status on the system terminal, until the password is entered. The program should log a limited number of set/alarm events (100 MAX) in memory and print these to the screen when required by the user.

**Visual Studio Source Code:**

Main.c:

#include "Init.h"

#include "Display.h"

#include "Options\_menu.h"

/\* Main.c \*/

int main()

{

int stop\_main = 0;

init\_all(); /\* initialize everything \*/

display(); /\* display informations about the alarms and zones \*/

while(!stop\_main) /\* never stop \*/

{

options\_menu(); /\* launch the menu \*/

}

return 0;

}

Options\_menu.h:

/\* Options\_menu.h \*/

int options\_menu(); /\* launch the menu \*/

Options\_menu.c:

#include <stdio.h>

#include "Password.h"

#include "Shell.h"

#include "Display.h"

#include "Automatic\_loop.h"

#include "Init.h"

#include "Mylog.h"

#include "Help.h"

/\* Options\_menu.c \*/

int options\_menu() /\* launch the menu \*/

{

int commande\_utilisateur;

printf("\n\r");

printf("Welcome to the options menu\n\r");

printf("Press 1 to change the password\n\r");

printf("Press 2 to activate zones\n\r");

printf("Press 3 to set alarms\n\r");

printf("Press 4 to display\n\r");

printf("Press 5 to launch the alarm system\n\r");

printf("Press 6 to print the log\n\r");

printf("Press 7 to reset all\n\r");

printf("Press 8 to have help\n\r");

scanf("%d",&commande\_utilisateur);

if(commande\_utilisateur == 1)

{

/\* change the password \*/

change\_password();

return 0;

}

if(commande\_utilisateur == 2)

{

/\* activate zones \*/

activate\_zones();

return 0;

}

if(commande\_utilisateur == 3)

{

/\* set alarms \*/

set\_alarms();

return 0;

}

if(commande\_utilisateur == 4)

{

/\* display \*/

display();

return 0;

}

if(commande\_utilisateur == 5)

{

/\* launch the alarm system \*/

automatic\_loop();

return 0;

}

if(commande\_utilisateur == 6)

{

/\* print the log \*/

print\_mylog();

return 0;

}

if(commande\_utilisateur == 7)

{

/\* reset \*/

init\_all();

return 0;

}

if(commande\_utilisateur == 8)

{

/\* help \*/

print\_help();

return 0;

}

return 1;

}

Help.h:

/\* Help.h \*/

int print\_help(); /\* print help message \*/

Help.c:

#include<stdio.h>

/\* Help.c \*/

int print\_help()

{

printf("Alarms are safe when alarm1 = 1 and on fire when alarm1 = 0 for example.\n\r");

printf("Zones are activated when zone1 = 1 and deactivated when zone1 = 0 for example.\n\r");

printf("In port a you have the eight alarms, the last alarm is on port e and the light on port g.\n\r");

return 0;

}

Automatic\_loop.h:

/\* Automatic\_loop.h \*/

int automatic\_loop(); /\* launch the automatic loop (everything is automatic you should press a key to return to the menu) \*/

int automatic\_loop\_microprocessor(); /\* for the microprocessor you should replace automatic\_loop() by automatic\_loop\_microprocessor() \*/

Automatic\_loop.c:

#include <stdio.h>

#include"Shell.h"

#include"Display.h"

#include"Input.h"

#include"Alarm.h"

#include"Output.h"

#include"Sleep.h"

#include"Mygetchar.h"

/\* Automatic\_loop.c \*/

int do\_you\_want\_return\_to\_the\_menu()

{

int commande\_utilisateur;

printf("do you want return to the menu?(yes = 1 / no = 0)\n\r");

scanf("%d",&commande\_utilisateur);

return commande\_utilisateur; /\* 1 the user want to stop the loop and 0 the user don t want to stop the loop \*/

}

int automatic\_loop()

{

int stop\_automatic\_loop = 0;

while(!stop\_automatic\_loop)

{

/\* if you want activate zones in this mod

activate\_zones();

\*/

set\_alarms();

display();

stop\_automatic\_loop = do\_you\_want\_return\_to\_the\_menu(); /\* 1 stop the loop and 0 don t stop the loop \*/

}

return 0;

}

int automatic\_loop\_microprocessor()

{

int stop\_automatic\_loop = 0;

unsigned char key;

while(!stop\_automatic\_loop)

{

port\_a(); /\* take the information of port a, extract it and change values of alarms \*/

port\_e(); /\* take the information of port e, extract it and change values of alarms \*/

update\_zones(); /\* update zones values \*/

port\_g(); /\* change the value of port g for turn on or turn off the light (depending to the alarms value) \*/

display(); /\* display informations about the alarms and zones \*/

sleep(1000); /\* a little break \*/

key = mygetchar(); /\* take a key if the user press the keyboard \*/

if(key != '\0') /\* if the keyboard is press then stop the loop \*/

{

stop\_automatic\_loop = 1;

}

}

return 0;

}

Init.h:

/\* Init.h \*/

int init\_all(); /\* initialize everything (or reset everything) \*/

Init.c:

#include<stdio.h>

#include"Alarm.h"

#include"Password.h"

#include"Mylog.h"

/\* Init.c \*/

int init\_all()

{

printf("initialization...\n\r");

init\_alarms();

init\_zone();

init\_zone\_active();

init\_password();

init\_mylog();

printf("finish\n\r");

return 0;

}

Display.h:

/\* Display.h \*/

int display(); /\* display informations about the alarms and zones \*/

Display.c:

#include <stdio.h>

#include "Alarm.h"

/\* Display.c \*/

int display\_zones\_active()

{

if(get\_zones\_active1())

{

printf("zones 1 active\n\r");

}

else

{

printf("zones 1 desactive\n\r");

}

if(get\_zones\_active2())

{

printf("zones 2 active\n\r");

}

else

{

printf("zones 2 desactive\n\r");

}

if(get\_zones\_active3())

{

printf("zones 3 active\n\r");

}

else

{

printf("zones 3 desactive\n\r");

}

return 0;

}

int display\_alarms\_in\_zone(int zone)

{

if(zone==1)

{

printf(" alarm 1 = %d\n\r",get\_alarm1());

printf(" alarm 2 = %d\n\r",get\_alarm2());

printf(" alarm 3 = %d\n\r",get\_alarm3());

}

if(zone==2)

{

printf(" alarm 4 = %d\n\r",get\_alarm4());

printf(" alarm 5 = %d\n\r",get\_alarm5());

printf(" alarm 6 = %d\n\r",get\_alarm6());

}

if(zone==3)

{

printf(" alarm 7 = %d\n\r",get\_alarm7());

printf(" alarm 8 = %d\n\r",get\_alarm8());

printf(" alarm 9 = %d\n\r",get\_alarm9());

}

return 0;

}

int display\_the\_situation()

{

if(get\_zones\_active1())

{

if(get\_zone1()) /\* if zone n safe then print "zone n safe" else print zone n on fire and print all the alarms in the zone n \*/

{

printf("zone 1 safe\n\r");

}

else

{

printf("zone 1 on fire\n\r");

display\_alarms\_in\_zone(1);

}

}

if(get\_zones\_active2())

{

if(get\_zone2())

{

printf("zone 2 safe\n\r");

}

else

{

printf("zone 2 on fire\n\r");

display\_alarms\_in\_zone(2);

}

}

if(get\_zones\_active3())

{

if(get\_zone3())

{

printf("zone 3 safe\n\r");

}

else

{

printf("zone 3 on fire\n\r");

display\_alarms\_in\_zone(3);

}

}

return 0;

}

int display\_zone1\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone1\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone2\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone2\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone3\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone3\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_the\_house()

{

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\r");

if(get\_zones\_active1())

{

if(get\_zone1())

{

display\_zone1\_safe();

}

else

{

display\_zone1\_fire();

}

}

if(get\_zones\_active2())

{

if(get\_zone2())

{

display\_zone2\_safe();

}

else

{

display\_zone2\_fire();

}

}

if(get\_zones\_active3())

{

if(get\_zone3())

{

display\_zone3\_safe();

}

else

{

display\_zone3\_fire();

}

}

return 0;

}

int display()

{

printf("\n\r");

display\_zones\_active();

printf("\n\r");

display\_the\_situation();

printf("\n\r");

display\_the\_house();

printf("\n\r");

return 0;

}

Input.h:

/\* Input.h \*/

int port\_a(); /\* take the information of port a and give it to extraction\_port\_a(information\_of\_port\_a) in Extraction.h \*/

int port\_e(); /\* take the information of port e and give it to extraction\_port\_e(information\_of\_port\_e) in Extraction.h \*/

Input.c:

#include <stdio.h>

#include "Extraction.h"

/\* Input.c \*/

int port\_a()

{

unsigned char \*porta,\*ddra;

porta=(unsigned char \*)0x00;

/\*Cast values into pointers\*/

ddra=(unsigned char \*)0x01;

\*ddra=0x00;

/\*Set DDRA to all inputs\*/

extraction\_port\_a(\*porta); /\* extract all informations of port a and change values of alarms \*/

return 0;

}

int port\_e()

{

unsigned char \*porte;

porte=(unsigned char \*)0x0a;

/\*Cast values into pointers\*/

extraction\_port\_e(\*porte); /\* extract all informations of port e and change values of alarms \*/

return 0;

}

Mygetchar.h:

/\* Mygetchar.h \*/

unsigned char mygetchar(); /\* a version of getchar but no wait the user, if the user press a key mygetchar return the key else mygetchar return \0 \*/

Mygetchar.c:

/\* Mygetchar.c \*/

unsigned char mygetchar()

{

unsigned char key;

unsigned char \*scsr, \*scdr;

scsr =(unsigned char \*)0x2e;

scdr = (unsigned char \*)0x2f;

if(\*scsr & 0x20) /\* masking, if != 0 then return \*scdr else return \0 \*/

{

key = \*scdr;

}

else

{

key = '\0';

}

return key; /\* if a key is press then return the key press else return \0 \*/

}

Sleep.h:

/\* Sleep.h \*/

int sleep(int limit); /\* a function to do a little break \*/

Sleep.c:

/\* Sleep.c \*/

int sleep(int limit)

{

int compteur = 0;

while(compteur < limit)

{

compteur++;

}

return 0;

}

Shell.h:

/\* Shell.h \*/

int activate\_zones(); /\* a function ask to the user which zones are activated \*/

int set\_alarms (); /\* a function which simulate sensor for the visual studio program and ask to the user what are the values for the alarms \*/

Shell.c:

#include <stdio.h>

#include "Alarm.h"

#include "Mylog.h"

/\* Shell.c \*/

int activate\_zones()

{

int commande\_utilisateur;

do

{

printf("zone 1 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1)); /\* check the user command and ask again if it is not correct \*/

set\_zones\_active1(commande\_utilisateur);

do

{

printf("zone 2 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_zones\_active2(commande\_utilisateur);

do

{

printf("zone 3 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_zones\_active3(commande\_utilisateur);

update\_mylog\_zones\_active(); /\* add a line to the log with the time about which zones are active \*/

return 0;

}

int set\_alarms ()

{

int commande\_utilisateur;

do

{

printf("alarm 1 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1)); /\* check the user command and ask again if it is not correct \*/

set\_alarm1(commande\_utilisateur);

do

{

printf("alarm 2 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm2(commande\_utilisateur);

do

{

printf("alarm 3 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm3(commande\_utilisateur);

do

{

printf("alarm 4 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm4(commande\_utilisateur);

do

{

printf("alarm 5 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm5(commande\_utilisateur);

do

{

printf("alarm 6 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm6(commande\_utilisateur);

do

{

printf("alarm 7 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm7(commande\_utilisateur);

printf("alarm 8 = ");

scanf("%d", &commande\_utilisateur);

set\_alarm8(commande\_utilisateur);

do

{

printf("alarm 9 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm9(commande\_utilisateur);

update\_zones(); /\* values of zones are changed automatically with alarms values \*/

update\_mylog\_zones\_on\_fire(); /\* add a line to the log with the time about which zones are on fire \*/

update\_mylog\_alarms\_on\_fire(); /\* add a line to the log with the time about which alarms are on fire \*/

return 0;

}

Password.h:

/\* Password.h \*/

int init\_password(); /\* initialize the password \*/

int change\_password(); /\* launch the procedure for changing the password \*/

int request\_password(); /\* ask the password to the user, if bad password return 1 else return 0 \*/

Password.c:

#include <stdio.h>

#include <string.h>

/\* Password.c \*/

char password[200]; /\* the password \*/

int init\_password() /\* initialize the password \*/

{

char new\_password[200];

new\_password[0]='\0';

printf("press the password what you want\n\r");

scanf("%s",new\_password); /\* ask what is the password for the initialization \*/

strcpy(password,new\_password);

return 0;

}

int set\_password(char \*new\_password) /\* change the password \*/

{

strcpy(password,new\_password);

return 0;

}

int test\_password(char \*try\_password) /\* test if the parameter is the same of the password \*/

{

return strcmp(password,try\_password); /\* return 0 if it's ok \*/

}

int change\_password() /\* launch the procedure for changing the password \*/

{

char old\_password[200];

char new\_password[200];

printf("change password...\n\r");

printf("press the old password\n\r");

scanf("%s",old\_password);

if(test\_password(old\_password)) /\* if old password bab then \*/

{

printf("bad password\n\r");

return 1; /\* stop \*/

}

printf("press the new password\n\r");

scanf("%s",new\_password);

printf("press again the new password\n\r");

scanf("%s",old\_password);

if(strcmp(new\_password,old\_password)) /\* strcmp return 0 if strins are the same \*/

{

printf("it is not the same password\n\r");

return 1; /\* stop \*/

}

set\_password(old\_password); /\* change the password \*/

printf("password change with success\n\r");

return 0;

}

int request\_password() /\* ask the password to the user, if bad password return 1 else return 0 \*/

{

char old\_password[200];

printf("request password...\n\r");

printf("press the password\n\r");

scanf("%s",old\_password);

if(test\_password(old\_password)) /\* test if the parameter is the same of the password, return 0 if it's good \*/

{

printf("bad password\n\r");

return 1; /\* stop \*/

}

printf("good password\n\r");

return 0;

}

Extraction.h:

/\* Extraction.h \*/

int extraction\_port\_a(unsigned char interupteurs); /\* extract all informations of port a and change values of alarms \*/

int extraction\_port\_e(unsigned char interupteurs); /\* extract all informations of port e and change values of alarms \*/

Extraction.c:

#include "Alarm.h"

/\* Extraction.c \*/

int extraction\_port\_a(unsigned char interupteurs)

{

unsigned char tmp = interupteurs;

tmp = tmp & 0x01; /\* extract the n bit with a mask \*/

set\_alarm1(tmp); /\* change the value of alarm n \*/

tmp = interupteurs;

tmp = tmp & 0x02;

tmp = tmp >> 1; /\* after extract the n bit with a mask you should shift the bit to the fist place \*/

set\_alarm2(tmp);

tmp = interupteurs;

tmp = tmp & 0x04;

tmp = tmp >> 2;

set\_alarm3(tmp);

tmp = interupteurs;

tmp = tmp & 0x08;

tmp = tmp >> 3;

set\_alarm4(tmp);

tmp = interupteurs;

tmp = tmp & 0x10;

tmp = tmp >> 4;

set\_alarm5(tmp);

tmp = interupteurs;

tmp = tmp & 0x20;

tmp = tmp >> 5;

set\_alarm6(tmp);

tmp = interupteurs;

tmp = tmp & 0x40;

tmp = tmp >> 6;

set\_alarm7(tmp);

tmp = interupteurs;

tmp = tmp & 0x80;

tmp = tmp >> 7;

set\_alarm8(tmp);

return 0;

}

int extraction\_port\_e(unsigned char interupteurs)

{

unsigned char tmp = interupteurs;

tmp = tmp & 0x01; /\* extract the n bit with a mask \*/

set\_alarm9(tmp); /\* change the value of alarm n \*/

/\* if you want stop automatic\_loop\_microprocessor() with E2 take this code

tmp = interupteurs;

tmp = tmp & 0x02;

tmp = tmp >> 1;

set\_stop\_automatic\_loop(tmp);

\*/

/\* if you want change the zones active with switch E3 E4 E5 take this code

tmp = interupteurs;

tmp = tmp & 0x04;

tmp = tmp >> 2;

set\_zones\_active1(tmp);

tmp = interupteurs;

tmp = tmp & 0x08;

tmp = tmp >> 3;

set\_zones\_active2(tmp);

tmp = interupteurs;

tmp = tmp & 0x10;

tmp = tmp >> 4;

set\_zones\_active3(tmp);

\*/

return 0;

}

Output.h:

/\* Output.h \*/

int port\_g(); /\* change the value of port g automatically with alarms values to turn on or off the light \*/

Output.c:

#include "Alarm.h"

/\* Output.c \*/

int port\_g()

{

unsigned char \*portg,\*ddrg;

portg=(unsigned char \*)0x02;

/\*Cast values into pointers\*/

ddrg=(unsigned char \*)0x03;

\*ddrg=0xff;

/\*Set DDRA to all outputs\*/

/\* treatment \*/

if(everything\_is\_safe()) /\* everything is safe \*/

{

\*portg = 0x00; /\* no light \*/

}

else

{

\*portg = 0x01; /\* light \*/

}

return 0;

}

Mylog.h:

/\* Mylog.h \*/

int init\_mylog(); /\* initialization of the log \*/

int update\_mylog\_zones\_active(); /\* add a line to the log with the time about which zones are active \*/

int update\_mylog\_zones\_on\_fire(); /\* add a line to the log with the time about which zones are on fire \*/

int update\_mylog\_alarms\_on\_fire(); /\* add a line to the log with the time about which alarms are on fire \*/

int print\_mylog(); /\* print the log \*/

Mylog.c:

#include <stdio.h>

#include <string.h>

#include "MyTime.h"

#include "Alarm.h"

/\* Mylog.c \*/

char mylog[2000]; /\* the log \*/

int init\_mylog()

{

mylog[0]='\0';

return 0;

}

int add\_line\_to\_mylog(char \*new\_line) /\* add the time and the string in parameter in the log \*/

{

char new\_line\_final[1000]; /\* the goal is to create the new line \*/

char string\_time[1000];

new\_line\_final[0]='\0';

string\_time[0]='\0';

return\_time(string\_time); /\* put the current time on the string \*/

strcpy(new\_line\_final,string\_time);

strcat(new\_line\_final,"-> ");

strcat(new\_line\_final,new\_line);

strcat(new\_line\_final,"\n\r");

strcat(new\_line\_final,"\n\r");

strcat(mylog,new\_line\_final); /\* add the new line created to the log \*/

return 0;

}

int update\_mylog\_zones\_active() /\* add a line to the log with the time about which zones are active \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_zones\_active1() && get\_zones\_active2() && get\_zones\_active3())

{

strcpy(new\_line,"every zones are active");

}

if(!get\_zones\_active1())

{

strcat(new\_line,"zone 1 inactive, ");

}

if(!get\_zones\_active2())

{

strcat(new\_line,"zone 2 inactive, ");

}

if(!get\_zones\_active3())

{

strcat(new\_line,"zone 3 inactive, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int update\_mylog\_zones\_on\_fire() /\* add a line to the log with the time about which zones are on fire \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_zone1() && get\_zone2() && get\_zone3())

{

strcpy(new\_line,"every zones are safe");

}

if(!get\_zone1())

{

strcat(new\_line,"zone 1 on fire, ");

}

if(!get\_zone2())

{

strcat(new\_line,"zone 2 on fire, ");

}

if(!get\_zone3())

{

strcat(new\_line,"zone 3 on fire, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int update\_mylog\_alarms\_on\_fire() /\* add a line to the log with the time about which alarms are on fire \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_alarm1() && get\_alarm2() && get\_alarm3() && get\_alarm4() && get\_alarm5() && get\_alarm6() && get\_alarm7() && get\_alarm8() && get\_alarm9())

{

strcpy(new\_line,"no fire");

}

if(!get\_alarm1())

{

strcat(new\_line,"alarm1 fire, ");

}

if(!get\_alarm2())

{

strcat(new\_line,"alarm2 fire, ");

}

if(!get\_alarm3())

{

strcat(new\_line,"alarm3 fire, ");

}

if(!get\_alarm4())

{

strcat(new\_line,"alarm4 fire, ");

}

if(!get\_alarm5())

{

strcat(new\_line,"alarm5 fire, ");

}

if(!get\_alarm6())

{

strcat(new\_line,"alarm6 fire, ");

}

if(!get\_alarm7())

{

strcat(new\_line,"alarm7 fire, ");

}

if(!get\_alarm8())

{

strcat(new\_line,"alarm8 fire, ");

}

if(!get\_alarm9())

{

strcat(new\_line,"alarm9 fire, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int print\_mylog() /\* print the log \*/

{

if(mylog[0]=='\0') /\* print "no log for the moment" if no log \*/

{

printf("no log for the moment\n\r");

}

else

{

printf("%s",mylog);

}

return 0;

}

Alarm.h:

/\* Alarm.h \*/

int init\_alarms(); /\* initialization of alarms (all are safe) \*/

int init\_zone(); /\* initialization of zone (all are safe) \*/

int init\_zone\_active(); /\* initialization of which zones are active (all are deactivated) \*/

int update\_zones(); /\* values of zones are changed automatically with alarms values \*/

int get\_alarm1(); /\* have the value of alarm1 \*/

int set\_alarm1(int new\_alarm1); /\* change the value of alarm1 \*/

int get\_alarm2(); /\* have the value of alarm2 \*/

int set\_alarm2(int new\_alarm2); /\* change the value of alarm2 \*/

int get\_alarm3(); /\* have the value of alarm3 \*/

int set\_alarm3(int new\_alarm3); /\* change the value of alarm3 \*/

int get\_alarm4(); /\* have the value of alarm4 \*/

int set\_alarm4(int new\_alarm4); /\* change the value of alarm4 \*/

int get\_alarm5(); /\* have the value of alarm5 \*/

int set\_alarm5(int new\_alarm5); /\* change the value of alarm5 \*/

int get\_alarm6(); /\* have the value of alarm6 \*/

int set\_alarm6(int new\_alarm6); /\* change the value of alarm6 \*/

int get\_alarm7(); /\* have the value of alarm7 \*/

int set\_alarm7(int new\_alarm7); /\* change the value of alarm7 \*/

int get\_alarm8(); /\* have the value of alarm8 \*/

int set\_alarm8(int new\_alarm8); /\* change the value of alarm8 \*/

int get\_alarm9(); /\* have the value of alarm9 \*/

int set\_alarm9(int new\_alarm9); /\* change the value of alarm9 \*/

int get\_zone1(); /\* have the value of zone1 \*/

int get\_zone2(); /\* have the value of zone2 \*/

int get\_zone3(); /\* have the value of zone3 \*/

int get\_zones\_active1(); /\* have the value of zones active1 \*/

int set\_zones\_active1(int new\_zones\_active1); /\* change the value of zones active1 \*/

int get\_zones\_active2(); /\* have the value of zones active2 \*/

int set\_zones\_active2(int new\_zones\_active2); /\* change the value of zones active2 \*/

int get\_zones\_active3(); /\* have the value of zones active3 \*/

int set\_zones\_active3(int new\_zones\_active3); /\* change the value of zones active3 \*/

int everything\_is\_safe(); /\* return 1 if everything is safe (no fire) else 0 \*/

Alarm.c:

/\* Alarm.c \*/

struct Alarms

{

int alarm1;

int alarm2;

int alarm3;

int alarm4;

int alarm5;

int alarm6;

int alarm7;

int alarm8;

int alarm9;

};

struct Zones

{

int zone1;

int zone2;

int zone3;

};

struct Alarms alarms;

struct Zones zones;

struct Zones zones\_active;

int init\_alarms()

{

alarms.alarm1 = 1;

alarms.alarm2 = 1;

alarms.alarm3 = 1;

alarms.alarm4 = 1;

alarms.alarm5 = 1;

alarms.alarm6 = 1;

alarms.alarm7 = 1;

alarms.alarm8 = 1;

alarms.alarm9 = 1;

return 0;

}

int init\_zone()

{

zones.zone1 = 1;

zones.zone2 = 1;

zones.zone3 = 1;

return 0;

}

int init\_zone\_active()

{

zones\_active.zone1 = 0;

zones\_active.zone2 = 0;

zones\_active.zone3 = 0;

return 0;

}

int update\_zones()

{

if (alarms.alarm1 == 1 && alarms.alarm2 == 1 && alarms.alarm3 == 1)

{

zones.zone1 = 1;

}

else

{

zones.zone1 = 0;

}

if (alarms.alarm4 == 1 && alarms.alarm5 == 1 && alarms.alarm6 == 1)

{

zones.zone2 = 1;

}

else

{

zones.zone2 = 0;

}

if (alarms.alarm7 == 1 && alarms.alarm8 == 1 && alarms.alarm9 == 1)

{

zones.zone3 = 1;

}

else

{

zones.zone3 = 0;

}

return 0;

}

int get\_alarm1()

{

return alarms.alarm1;

}

int set\_alarm1(int new\_alarm1)

{

alarms.alarm1=new\_alarm1;

return 0;

}

int get\_alarm2()

{

return alarms.alarm2;

}

int set\_alarm2(int new\_alarm2)

{

alarms.alarm2=new\_alarm2;

return 0;

}

int get\_alarm3()

{

return alarms.alarm3;

}

int set\_alarm3(int new\_alarm3)

{

alarms.alarm3=new\_alarm3;

return 0;

}

int get\_alarm4()

{

return alarms.alarm4;

}

int set\_alarm4(int new\_alarm4)

{

alarms.alarm4=new\_alarm4;

return 0;

}

int get\_alarm5()

{

return alarms.alarm5;

}

int set\_alarm5(int new\_alarm5)

{

alarms.alarm5=new\_alarm5;

return 0;

}

int get\_alarm6()

{

return alarms.alarm6;

}

int set\_alarm6(int new\_alarm6)

{

alarms.alarm6=new\_alarm6;

return 0;

}

int get\_alarm7()

{

return alarms.alarm7;

}

int set\_alarm7(int new\_alarm7)

{

alarms.alarm7=new\_alarm7;

return 0;

}

int get\_alarm8()

{

return alarms.alarm8;

}

int set\_alarm8(int new\_alarm8)

{

alarms.alarm8=new\_alarm8;

return 0;

}

int get\_alarm9()

{

return alarms.alarm9;

}

int set\_alarm9(int new\_alarm9)

{

alarms.alarm9=new\_alarm9;

return 0;

}

int get\_zone1()

{

return zones.zone1;

}

int get\_zone2()

{

return zones.zone2;

}

int get\_zone3()

{

return zones.zone3;

}

int get\_zones\_active1()

{

return zones\_active.zone1;

}

int set\_zones\_active1(int new\_zones\_active1)

{

zones\_active.zone1=new\_zones\_active1;

return 0;

}

int get\_zones\_active2()

{

return zones\_active.zone2;

}

int set\_zones\_active2(int new\_zones\_active2)

{

zones\_active.zone2=new\_zones\_active2;

return 0;

}

int get\_zones\_active3()

{

return zones\_active.zone3;

}

int set\_zones\_active3(int new\_zones\_active3)

{

zones\_active.zone3=new\_zones\_active3;

return 0;

}

int everything\_is\_safe()

{

int safe = 1;

if(zones\_active.zone1)

{

if(!(alarms.alarm1 && alarms.alarm2 && alarms.alarm3))

{

safe = 0;

}

}

if(zones\_active.zone2)

{

if(!(alarms.alarm4 && alarms.alarm5 && alarms.alarm6))

{

safe = 0;

}

}

if(zones\_active.zone3)

{

if(!(alarms.alarm7 && alarms.alarm8 && alarms.alarm9))

{

safe = 0;

}

}

return safe;

}

MyTime.h:

/\* MyTime.h \*/

int return\_time(char \*string\_time); /\* put the current time on the string \*/

int return\_time\_microprocessor(char \*string\_time); /\* for the microprocessor you should replace return\_time(char \*string\_time) by return\_time\_microprocessor(char \*string\_time) \*/

MyTime.c:

#include <stdio.h>

#include <string.h>

#include <time.h>

/\* MyTime.c \*/

int return\_time(char \*string\_time)

{

time\_t t = time(NULL);

strcpy(string\_time,asctime(localtime(&t)));

return 0;

}

int return\_time\_microprocessor(char \*string\_time)

{

sprintf(string\_time,"Time");

return 0;

}

**Microcontroller Source Code:**

#include <stdio.h>

#include <string.h>

/\* Options\_menu.h \*/

int options\_menu(); /\* launch the menu \*/

/\* Help.h \*/

int print\_help(); /\* print help message \*/

/\* Automatic\_loop.h \*/

int automatic\_loop(); /\* launch the automatic loop (everything is automatic you should press a key to return to the menu) \*/

int automatic\_loop\_microprocessor(); /\* for the microprocessor you should replace automatic\_loop() by automatic\_loop\_microprocessor() \*/

/\* Init.h \*/

int init\_all(); /\* initialize everything (or reset everything) \*/

/\* Display.h \*/

int display(); /\* display informations about the alarms and zones \*/

/\* Input.h \*/

int port\_a(); /\* take the information of port a and give it to extraction\_port\_a(information\_of\_port\_a) in Extraction.h \*/

int port\_e(); /\* take the information of port e and give it to extraction\_port\_e(information\_of\_port\_e) in Extraction.h \*/

/\* Mygetchar.h \*/

unsigned char mygetchar(); /\* a version of getchar but no wait the user, if the user press a key mygetchar return the key else mygetchar return \0 \*/

/\* Sleep.h \*/

int sleep(int limit); /\* a function to do a little break \*/

/\* Shell.h \*/

int activate\_zones(); /\* a function ask to the user which zones are activated \*/

int set\_alarms (); /\* a function which simulate sensor for the visual studio program and ask to the user what are the values for the alarms \*/

/\* Password.h \*/

int init\_password(); /\* initialize the password \*/

int change\_password(); /\* launch the procedure for changing the password \*/

int request\_password(); /\* ask the password to the user, if bad password return 1 else return 0 \*/

/\* Extraction.h \*/

int extraction\_port\_a(unsigned char interupteurs); /\* extract all informations of port a and change values of alarms \*/

int extraction\_port\_e(unsigned char interupteurs); /\* extract all informations of port e and change values of alarms \*/

/\* Output.h \*/

int port\_g(); /\* change the value of port g automatically with alarms values to turn on or off the light \*/

/\* Mylog.h \*/

int init\_mylog(); /\* initialization of the log \*/

int update\_mylog\_zones\_active(); /\* add a line to the log with the time about which zones are active \*/

int update\_mylog\_zones\_on\_fire(); /\* add a line to the log with the time about which zones are on fire \*/

int update\_mylog\_alarms\_on\_fire(); /\* add a line to the log with the time about which alarms are on fire \*/

int print\_mylog(); /\* print the log \*/

/\* Alarm.h \*/

int init\_alarms(); /\* initialization of alarms (all are safe) \*/

int init\_zone(); /\* initialization of zone (all are safe) \*/

int init\_zone\_active(); /\* initialization of which zones are active (all are deactivated) \*/

int update\_zones(); /\* values of zones are changed automatically with alarms values \*/

int get\_alarm1(); /\* have the value of alarm1 \*/

int set\_alarm1(int new\_alarm1); /\* change the value of alarm1 \*/

int get\_alarm2(); /\* have the value of alarm2 \*/

int set\_alarm2(int new\_alarm2); /\* change the value of alarm2 \*/

int get\_alarm3(); /\* have the value of alarm3 \*/

int set\_alarm3(int new\_alarm3); /\* change the value of alarm3 \*/

int get\_alarm4(); /\* have the value of alarm4 \*/

int set\_alarm4(int new\_alarm4); /\* change the value of alarm4 \*/

int get\_alarm5(); /\* have the value of alarm5 \*/

int set\_alarm5(int new\_alarm5); /\* change the value of alarm5 \*/

int get\_alarm6(); /\* have the value of alarm6 \*/

int set\_alarm6(int new\_alarm6); /\* change the value of alarm6 \*/

int get\_alarm7(); /\* have the value of alarm7 \*/

int set\_alarm7(int new\_alarm7); /\* change the value of alarm7 \*/

int get\_alarm8(); /\* have the value of alarm8 \*/

int set\_alarm8(int new\_alarm8); /\* change the value of alarm8 \*/

int get\_alarm9(); /\* have the value of alarm9 \*/

int set\_alarm9(int new\_alarm9); /\* change the value of alarm9 \*/

int get\_zone1(); /\* have the value of zone1 \*/

int get\_zone2(); /\* have the value of zone2 \*/

int get\_zone3(); /\* have the value of zone3 \*/

int get\_zones\_active1(); /\* have the value of zones active1 \*/

int set\_zones\_active1(int new\_zones\_active1); /\* change the value of zones active1 \*/

int get\_zones\_active2(); /\* have the value of zones active2 \*/

int set\_zones\_active2(int new\_zones\_active2); /\* change the value of zones active2 \*/

int get\_zones\_active3(); /\* have the value of zones active3 \*/

int set\_zones\_active3(int new\_zones\_active3); /\* change the value of zones active3 \*/

int everything\_is\_safe(); /\* return 1 if everything is safe (no fire) else 0 \*/

/\* MyTime.h \*/

int return\_time\_microprocessor(char \*string\_time); /\* for the microprocessor you should replace return\_time(char \*string\_time) by return\_time\_microprocessor(char \*string\_time) \*/

/\* Main.c \*/

int hours,mins,secs,ticks;

int main()

{

unsigned char \*pactl,\*tmsk2;

int stop\_main = 0;

tmsk2=(unsigned char\*)0x24;

pactl=(unsigned char\*)0x26;

\*pactl=0x03; /\*Set prescaler to maximum\*/

\*tmsk2=0x40; /\*Enable RTI interrupt\*/

init\_all(); /\* initialize everything \*/

display(); /\* display informations about the alarms and zones \*/

while(!stop\_main) /\* never stop \*/

{

options\_menu(); /\* launch the menu \*/

}

return 0;

}

@interrupt void timer(void)

{

unsigned char \*tflg2;

tflg2=(unsigned char\*)0x25;

ticks++;

if (ticks==30)

{

ticks=0;

secs++;

}

if (secs==60)

{

secs=0;

mins++;

}

if (mins==60)

{

mins=0;

hours++;

}

if (hours==24)

{

hours=0;

}

\*tflg2=0x40; /\*Reset RTI flag\*/

}

/\* Options\_menu.c \*/

int options\_menu() /\* launch the menu \*/

{

int commande\_utilisateur;

printf("\n\r");

printf("Welcome to the options menu\n\r");

printf("Press 1 to change the password\n\r");

printf("Press 2 to activate zones\n\r");

printf("Press 3 to set alarms\n\r");

printf("Press 4 to display\n\r");

printf("Press 5 to launch the alarm system\n\r");

printf("Press 6 to print the log\n\r");

printf("Press 7 to reset all\n\r");

printf("Press 8 to have help\n\r");

scanf("%d",&commande\_utilisateur);

if(commande\_utilisateur == 1)

{

/\* change the password \*/

change\_password();

return 0;

}

if(commande\_utilisateur == 2)

{

/\* activate zones \*/

activate\_zones();

return 0;

}

if(commande\_utilisateur == 3)

{

/\* set alarms \*/

set\_alarms();

return 0;

}

if(commande\_utilisateur == 4)

{

/\* display \*/

display();

return 0;

}

if(commande\_utilisateur == 5)

{

/\* launch the alarm system \*/

automatic\_loop\_microprocessor();

return 0;

}

if(commande\_utilisateur == 6)

{

/\* print the log \*/

print\_mylog();

return 0;

}

if(commande\_utilisateur == 7)

{

/\* reset \*/

init\_all();

return 0;

}

if(commande\_utilisateur == 8)

{

/\* help \*/

print\_help();

return 0;

}

return 1;

}

/\* Help.c \*/

int print\_help()

{

printf("Alarms are safe when alarm1 = 1 and on fire when alarm1 = 0 for example.\n\r");

printf("Zones are activated when zone1 = 1 and deactivated when zone1 = 0 for example.\n\r");

printf("In port a you have the eight alarms, the last alarm is on port e and the light on port g.\n\r");

return 0;

}

/\* Automatic\_loop.c \*/

int do\_you\_want\_return\_to\_the\_menu()

{

int commande\_utilisateur;

printf("do you want return to the menu?(yes = 1 / no = 0)\n\r");

scanf("%d",&commande\_utilisateur);

return commande\_utilisateur; /\* 1 the user want to stop the loop and 0 the user don t want to stop the loop \*/

}

int automatic\_loop()

{

int stop\_automatic\_loop = 0;

while(!stop\_automatic\_loop)

{

/\* if you want activate zones in this mod

activate\_zones();

\*/

set\_alarms();

display();

stop\_automatic\_loop = do\_you\_want\_return\_to\_the\_menu(); /\* 1 stop the loop and 0 don t stop the loop \*/

}

return 0;

}

int automatic\_loop\_microprocessor()

{

int stop\_automatic\_loop = 0;

unsigned char key;

while(!stop\_automatic\_loop)

{

port\_a(); /\* take the information of port a, extract it and change values of alarms \*/

port\_e(); /\* take the information of port e, extract it and change values of alarms \*/

update\_zones(); /\* update zones values \*/

port\_g(); /\* change the value of port g for turn on or turn off the light (depending to the alarms value) \*/

display(); /\* display informations about the alarms and zones \*/

sleep(1000); /\* a little break \*/

key = mygetchar(); /\* take a key if the user press the keyboard \*/

if(key != '\0') /\* if the keyboard is press then stop the loop \*/

{

stop\_automatic\_loop = 1;

}

}

return 0;

}

/\* Init.c \*/

int init\_all()

{

printf("initialization...\n\r");

init\_alarms();

init\_zone();

init\_zone\_active();

init\_password();

init\_mylog();

printf("finish\n\r");

return 0;

}

/\* Display.c \*/

int display\_zones\_active()

{

if(get\_zones\_active1())

{

printf("zones 1 active\n\r");

}

else

{

printf("zones 1 desactive\n\r");

}

if(get\_zones\_active2())

{

printf("zones 2 active\n\r");

}

else

{

printf("zones 2 desactive\n\r");

}

if(get\_zones\_active3())

{

printf("zones 3 active\n\r");

}

else

{

printf("zones 3 desactive\n\r");

}

return 0;

}

int display\_alarms\_in\_zone(int zone)

{

if(zone==1)

{

printf(" alarm 1 = %d\n\r",get\_alarm1());

printf(" alarm 2 = %d\n\r",get\_alarm2());

printf(" alarm 3 = %d\n\r",get\_alarm3());

}

if(zone==2)

{

printf(" alarm 4 = %d\n\r",get\_alarm4());

printf(" alarm 5 = %d\n\r",get\_alarm5());

printf(" alarm 6 = %d\n\r",get\_alarm6());

}

if(zone==3)

{

printf(" alarm 7 = %d\n\r",get\_alarm7());

printf(" alarm 8 = %d\n\r",get\_alarm8());

printf(" alarm 9 = %d\n\r",get\_alarm9());

}

return 0;

}

int display\_the\_situation()

{

if(get\_zones\_active1())

{

if(get\_zone1()) /\* if zone n safe then print "zone n safe" else print zone n on fire and print all the alarms in the zone n \*/

{

printf("zone 1 safe\n\r");

}

else

{

printf("zone 1 on fire\n\r");

display\_alarms\_in\_zone(1);

}

}

if(get\_zones\_active2())

{

if(get\_zone2())

{

printf("zone 2 safe\n\r");

}

else

{

printf("zone 2 on fire\n\r");

display\_alarms\_in\_zone(2);

}

}

if(get\_zones\_active3())

{

if(get\_zone3())

{

printf("zone 3 safe\n\r");

}

else

{

printf("zone 3 on fire\n\r");

display\_alarms\_in\_zone(3);

}

}

return 0;

}

int display\_zone1\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone1\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone2\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone2\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone3\_safe()

{

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("| |\n\r");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_zone3\_fire()

{

printf("| |\n\r");

printf("| ) |\n\r");

printf("| ) | |\n\r");

printf("| / ) ( |\n\r");

printf("|\_\_\_\_\_\_\_\(\_)/\_\_\_\_\_\_\_\_|\n\r");

return 0;

}

int display\_the\_house()

{

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\r");

if(get\_zones\_active1())

{

if(get\_zone1())

{

display\_zone1\_safe();

}

else

{

display\_zone1\_fire();

}

}

if(get\_zones\_active2())

{

if(get\_zone2())

{

display\_zone2\_safe();

}

else

{

display\_zone2\_fire();

}

}

if(get\_zones\_active3())

{

if(get\_zone3())

{

display\_zone3\_safe();

}

else

{

display\_zone3\_fire();

}

}

return 0;

}

int display()

{

printf("\n\r");

display\_zones\_active();

printf("\n\r");

display\_the\_situation();

printf("\n\r");

display\_the\_house();

printf("\n\r");

return 0;

}

/\* Input.c \*/

int port\_a()

{

unsigned char \*porta,\*ddra;

porta=(unsigned char \*)0x00;

/\*Cast values into pointers\*/

ddra=(unsigned char \*)0x01;

\*ddra=0x00;

/\*Set DDRA to all inputs\*/

extraction\_port\_a(\*porta); /\* extract all informations of port a and change values of alarms \*/

return 0;

}

int port\_e()

{

unsigned char \*porte;

porte=(unsigned char \*)0x0a;

/\*Cast values into pointers\*/

extraction\_port\_e(\*porte); /\* extract all informations of port e and change values of alarms \*/

return 0;

}

/\* Mygetchar.c \*/

unsigned char mygetchar()

{

unsigned char key;

unsigned char \*scsr, \*scdr;

scsr =(unsigned char \*)0x2e;

scdr = (unsigned char \*)0x2f;

if(\*scsr & 0x20) /\* masking, if != 0 then return \*scdr else return \0 \*/

{

key = \*scdr;

}

else

{

key = '\0';

}

return key; /\* if a key is press then return the key press else return \0 \*/

}

/\* Sleep.c \*/

int sleep(int limit)

{

int compteur = 0;

while(compteur < limit)

{

compteur++;

}

return 0;

}

/\* Shell.c \*/

int activate\_zones()

{

int commande\_utilisateur;

do

{

printf("zone 1 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1)); /\* check the user command and ask again if it is not correct \*/

set\_zones\_active1(commande\_utilisateur);

do

{

printf("zone 2 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_zones\_active2(commande\_utilisateur);

do

{

printf("zone 3 active = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_zones\_active3(commande\_utilisateur);

update\_mylog\_zones\_active(); /\* add a line to the log with the time about which zones are active \*/

return 0;

}

int set\_alarms ()

{

int commande\_utilisateur;

do

{

printf("alarm 1 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1)); /\* check the user command and ask again if it is not correct \*/

set\_alarm1(commande\_utilisateur);

do

{

printf("alarm 2 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm2(commande\_utilisateur);

do

{

printf("alarm 3 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm3(commande\_utilisateur);

do

{

printf("alarm 4 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm4(commande\_utilisateur);

do

{

printf("alarm 5 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm5(commande\_utilisateur);

do

{

printf("alarm 6 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm6(commande\_utilisateur);

do

{

printf("alarm 7 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm7(commande\_utilisateur);

printf("alarm 8 = ");

scanf("%d", &commande\_utilisateur);

set\_alarm8(commande\_utilisateur);

do

{

printf("alarm 9 = ");

scanf("%d", &commande\_utilisateur);

} while(!(commande\_utilisateur==0 || commande\_utilisateur==1));

set\_alarm9(commande\_utilisateur);

update\_zones(); /\* values of zones are changed automatically with alarms values \*/

update\_mylog\_zones\_on\_fire(); /\* add a line to the log with the time about which zones are on fire \*/

update\_mylog\_alarms\_on\_fire(); /\* add a line to the log with the time about which alarms are on fire \*/

return 0;

}

/\* Password.c \*/

char password[200]; /\* the password \*/

int init\_password() /\* initialize the password \*/

{

char new\_password[200];

new\_password[0]='\0';

printf("press the password what you want\n\r");

scanf("%s",new\_password); /\* ask what is the password for the initialization \*/

strcpy(password,new\_password);

return 0;

}

int set\_password(char \*new\_password) /\* change the password \*/

{

strcpy(password,new\_password);

return 0;

}

int test\_password(char \*try\_password) /\* test if the parameter is the same of the password \*/

{

return strcmp(password,try\_password); /\* return 0 if it's ok \*/

}

int change\_password() /\* launch the procedure for changing the password \*/

{

char old\_password[200];

char new\_password[200];

printf("change password...\n\r");

printf("press the old password\n\r");

scanf("%s",old\_password);

if(test\_password(old\_password)) /\* if old password bab then \*/

{

printf("bad password\n\r");

return 1; /\* stop \*/

}

printf("press the new password\n\r");

scanf("%s",new\_password);

printf("press again the new password\n\r");

scanf("%s",old\_password);

if(strcmp(new\_password,old\_password)) /\* strcmp return 0 if strins are the same \*/

{

printf("it is not the same password\n\r");

return 1; /\* stop \*/

}

set\_password(old\_password); /\* change the password \*/

printf("password change with success\n\r");

return 0;

}

int request\_password() /\* ask the password to the user, if bad password return 1 else return 0 \*/

{

char old\_password[200];

printf("request password...\n\r");

printf("press the password\n\r");

scanf("%s",old\_password);

if(test\_password(old\_password)) /\* test if the parameter is the same of the password, return 0 if it's good \*/

{

printf("bad password\n\r");

return 1; /\* stop \*/

}

printf("good password\n\r");

return 0;

}

/\* Extraction.c \*/

int extraction\_port\_a(unsigned char interupteurs)

{

unsigned char tmp = interupteurs;

tmp = tmp & 0x01; /\* extract the n bit with a mask \*/

set\_alarm1(tmp); /\* change the value of alarm n \*/

tmp = interupteurs;

tmp = tmp & 0x02;

tmp = tmp >> 1; /\* after extract the n bit with a mask you should shift the bit to the fist place \*/

set\_alarm2(tmp);

tmp = interupteurs;

tmp = tmp & 0x04;

tmp = tmp >> 2;

set\_alarm3(tmp);

tmp = interupteurs;

tmp = tmp & 0x08;

tmp = tmp >> 3;

set\_alarm4(tmp);

tmp = interupteurs;

tmp = tmp & 0x10;

tmp = tmp >> 4;

set\_alarm5(tmp);

tmp = interupteurs;

tmp = tmp & 0x20;

tmp = tmp >> 5;

set\_alarm6(tmp);

tmp = interupteurs;

tmp = tmp & 0x40;

tmp = tmp >> 6;

set\_alarm7(tmp);

tmp = interupteurs;

tmp = tmp & 0x80;

tmp = tmp >> 7;

set\_alarm8(tmp);

return 0;

}

int extraction\_port\_e(unsigned char interupteurs)

{

unsigned char tmp = interupteurs;

tmp = tmp & 0x01; /\* extract the n bit with a mask \*/

set\_alarm9(tmp); /\* change the value of alarm n \*/

/\* if you want stop automatic\_loop\_microprocessor() with E2 take this code

tmp = interupteurs;

tmp = tmp & 0x02;

tmp = tmp >> 1;

set\_stop\_automatic\_loop(tmp);

\*/

/\* if you want change the zones active with switch E3 E4 E5 take this code

tmp = interupteurs;

tmp = tmp & 0x04;

tmp = tmp >> 2;

set\_zones\_active1(tmp);

tmp = interupteurs;

tmp = tmp & 0x08;

tmp = tmp >> 3;

set\_zones\_active2(tmp);

tmp = interupteurs;

tmp = tmp & 0x10;

tmp = tmp >> 4;

set\_zones\_active3(tmp);

\*/

return 0;

}

/\* Output.c \*/

int port\_g()

{

unsigned char \*portg,\*ddrg;

portg=(unsigned char \*)0x02;

/\*Cast values into pointers\*/

ddrg=(unsigned char \*)0x03;

\*ddrg=0xff;

/\*Set DDRA to all outputs\*/

/\* treatment \*/

if(everything\_is\_safe()) /\* everything is safe \*/

{

\*portg = 0x00; /\* no light \*/

}

else

{

\*portg = 0x01; /\* light \*/

}

return 0;

}

/\* Mylog.c \*/

char mylog[2000]; /\* the log \*/

int init\_mylog()

{

mylog[0]='\0';

return 0;

}

int add\_line\_to\_mylog(char \*new\_line) /\* add the time and the string in parameter in the log \*/

{

char new\_line\_final[1000]; /\* the goal is to create the new line \*/

char string\_time[1000];

new\_line\_final[0]='\0';

string\_time[0]='\0';

return\_time\_microprocessor(string\_time); /\* put the current time on the string \*/

strcpy(new\_line\_final,string\_time);

strcat(new\_line\_final,"-> ");

strcat(new\_line\_final,new\_line);

strcat(new\_line\_final,"\n\r");

strcat(new\_line\_final,"\n\r");

strcat(mylog,new\_line\_final); /\* add the new line created to the log \*/

return 0;

}

int update\_mylog\_zones\_active() /\* add a line to the log with the time about which zones are active \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_zones\_active1() && get\_zones\_active2() && get\_zones\_active3())

{

strcpy(new\_line,"every zones are active");

}

if(!get\_zones\_active1())

{

strcat(new\_line,"zone 1 inactive, ");

}

if(!get\_zones\_active2())

{

strcat(new\_line,"zone 2 inactive, ");

}

if(!get\_zones\_active3())

{

strcat(new\_line,"zone 3 inactive, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int update\_mylog\_zones\_on\_fire() /\* add a line to the log with the time about which zones are on fire \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_zone1() && get\_zone2() && get\_zone3())

{

strcpy(new\_line,"every zones are safe");

}

if(!get\_zone1())

{

strcat(new\_line,"zone 1 on fire, ");

}

if(!get\_zone2())

{

strcat(new\_line,"zone 2 on fire, ");

}

if(!get\_zone3())

{

strcat(new\_line,"zone 3 on fire, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int update\_mylog\_alarms\_on\_fire() /\* add a line to the log with the time about which alarms are on fire \*/

{

char new\_line[1000]; /\* the goal is to create the new line \*/

new\_line[0]='\0';

if(get\_alarm1() && get\_alarm2() && get\_alarm3() && get\_alarm4() && get\_alarm5() && get\_alarm6() && get\_alarm7() && get\_alarm8() && get\_alarm9())

{

strcpy(new\_line,"no fire");

}

if(!get\_alarm1())

{

strcat(new\_line,"alarm1 fire, ");

}

if(!get\_alarm2())

{

strcat(new\_line,"alarm2 fire, ");

}

if(!get\_alarm3())

{

strcat(new\_line,"alarm3 fire, ");

}

if(!get\_alarm4())

{

strcat(new\_line,"alarm4 fire, ");

}

if(!get\_alarm5())

{

strcat(new\_line,"alarm5 fire, ");

}

if(!get\_alarm6())

{

strcat(new\_line,"alarm6 fire, ");

}

if(!get\_alarm7())

{

strcat(new\_line,"alarm7 fire, ");

}

if(!get\_alarm8())

{

strcat(new\_line,"alarm8 fire, ");

}

if(!get\_alarm9())

{

strcat(new\_line,"alarm9 fire, ");

}

add\_line\_to\_mylog(new\_line); /\* this function add the time and the string in parameter in the log \*/

return 0;

}

int print\_mylog() /\* print the log \*/

{

if(mylog[0]=='\0') /\* print "no log for the moment" if no log \*/

{

printf("no log for the moment\n\r");

}

else

{

printf("%s",mylog);

}

return 0;

}

/\* Alarm.c \*/

struct Alarms

{

int alarm1;

int alarm2;

int alarm3;

int alarm4;

int alarm5;

int alarm6;

int alarm7;

int alarm8;

int alarm9;

};

struct Zones

{

int zone1;

int zone2;

int zone3;

};

struct Alarms alarms;

struct Zones zones;

struct Zones zones\_active;

int init\_alarms()

{

alarms.alarm1 = 1;

alarms.alarm2 = 1;

alarms.alarm3 = 1;

alarms.alarm4 = 1;

alarms.alarm5 = 1;

alarms.alarm6 = 1;

alarms.alarm7 = 1;

alarms.alarm8 = 1;

alarms.alarm9 = 1;

return 0;

}

int init\_zone()

{

zones.zone1 = 1;

zones.zone2 = 1;

zones.zone3 = 1;

return 0;

}

int init\_zone\_active()

{

zones\_active.zone1 = 0;

zones\_active.zone2 = 0;

zones\_active.zone3 = 0;

return 0;

}

int update\_zones()

{

if (alarms.alarm1 == 1 && alarms.alarm2 == 1 && alarms.alarm3 == 1)

{

zones.zone1 = 1;

}

else

{

zones.zone1 = 0;

}

if (alarms.alarm4 == 1 && alarms.alarm5 == 1 && alarms.alarm6 == 1)

{

zones.zone2 = 1;

}

else

{

zones.zone2 = 0;

}

if (alarms.alarm7 == 1 && alarms.alarm8 == 1 && alarms.alarm9 == 1)

{

zones.zone3 = 1;

}

else

{

zones.zone3 = 0;

}

return 0;

}

int get\_alarm1()

{

return alarms.alarm1;

}

int set\_alarm1(int new\_alarm1)

{

alarms.alarm1=new\_alarm1;

return 0;

}

int get\_alarm2()

{

return alarms.alarm2;

}

int set\_alarm2(int new\_alarm2)

{

alarms.alarm2=new\_alarm2;

return 0;

}

int get\_alarm3()

{

return alarms.alarm3;

}

int set\_alarm3(int new\_alarm3)

{

alarms.alarm3=new\_alarm3;

return 0;

}

int get\_alarm4()

{

return alarms.alarm4;

}

int set\_alarm4(int new\_alarm4)

{

alarms.alarm4=new\_alarm4;

return 0;

}

int get\_alarm5()

{

return alarms.alarm5;

}

int set\_alarm5(int new\_alarm5)

{

alarms.alarm5=new\_alarm5;

return 0;

}

int get\_alarm6()

{

return alarms.alarm6;

}

int set\_alarm6(int new\_alarm6)

{

alarms.alarm6=new\_alarm6;

return 0;

}

int get\_alarm7()

{

return alarms.alarm7;

}

int set\_alarm7(int new\_alarm7)

{

alarms.alarm7=new\_alarm7;

return 0;

}

int get\_alarm8()

{

return alarms.alarm8;

}

int set\_alarm8(int new\_alarm8)

{

alarms.alarm8=new\_alarm8;

return 0;

}

int get\_alarm9()

{

return alarms.alarm9;

}

int set\_alarm9(int new\_alarm9)

{

alarms.alarm9=new\_alarm9;

return 0;

}

int get\_zone1()

{

return zones.zone1;

}

int get\_zone2()

{

return zones.zone2;

}

int get\_zone3()

{

return zones.zone3;

}

int get\_zones\_active1()

{

return zones\_active.zone1;

}

int set\_zones\_active1(int new\_zones\_active1)

{

zones\_active.zone1=new\_zones\_active1;

return 0;

}

int get\_zones\_active2()

{

return zones\_active.zone2;

}

int set\_zones\_active2(int new\_zones\_active2)

{

zones\_active.zone2=new\_zones\_active2;

return 0;

}

int get\_zones\_active3()

{

return zones\_active.zone3;

}

int set\_zones\_active3(int new\_zones\_active3)

{

zones\_active.zone3=new\_zones\_active3;

return 0;

}

int everything\_is\_safe()

{

int safe = 1;

if(zones\_active.zone1)

{

if(!(alarms.alarm1 && alarms.alarm2 && alarms.alarm3))

{

safe = 0;

}

}

if(zones\_active.zone2)

{

if(!(alarms.alarm4 && alarms.alarm5 && alarms.alarm6))

{

safe = 0;

}

}

if(zones\_active.zone3)

{

if(!(alarms.alarm7 && alarms.alarm8 && alarms.alarm9))

{

safe = 0;

}

}

return safe;

}

/\* MyTime.c \*/

int return\_time\_microprocessor(char \*string\_time)

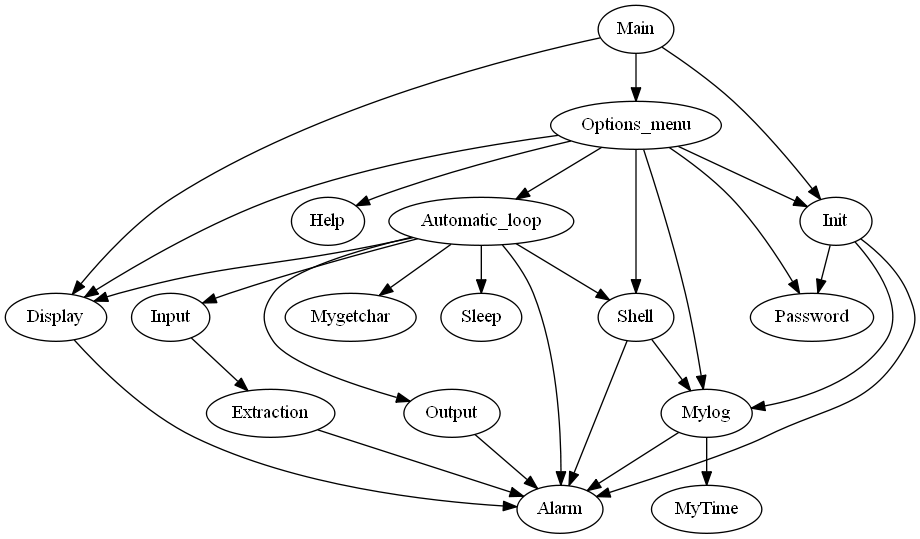
{

sprintf(string\_time,"%dh %dm %ds ",hours,mins,secs);

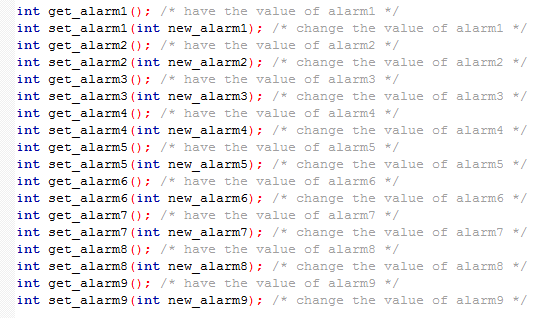
return 0;

}

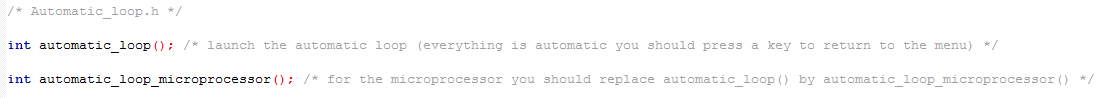
**Structured Design Approach:**



As we can see from this dependency graph, the code is structured in full of small file.

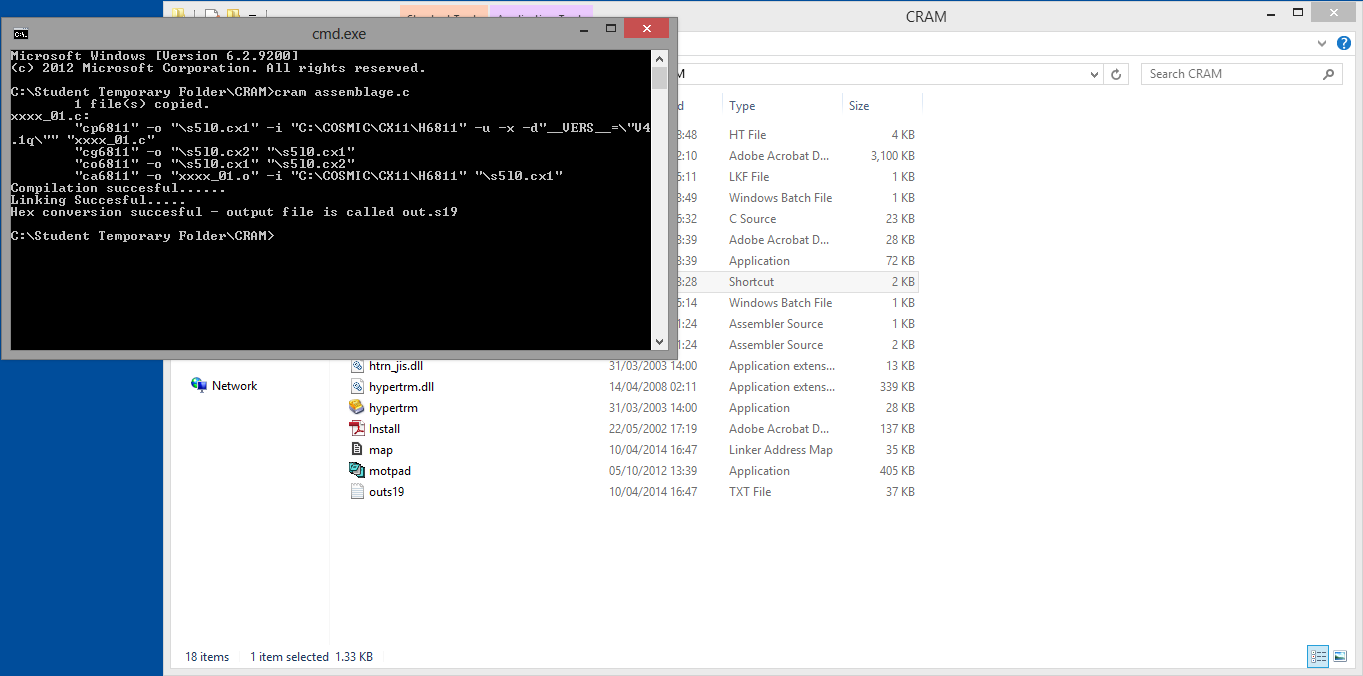


They are independent and interact with each other thanks to the header.

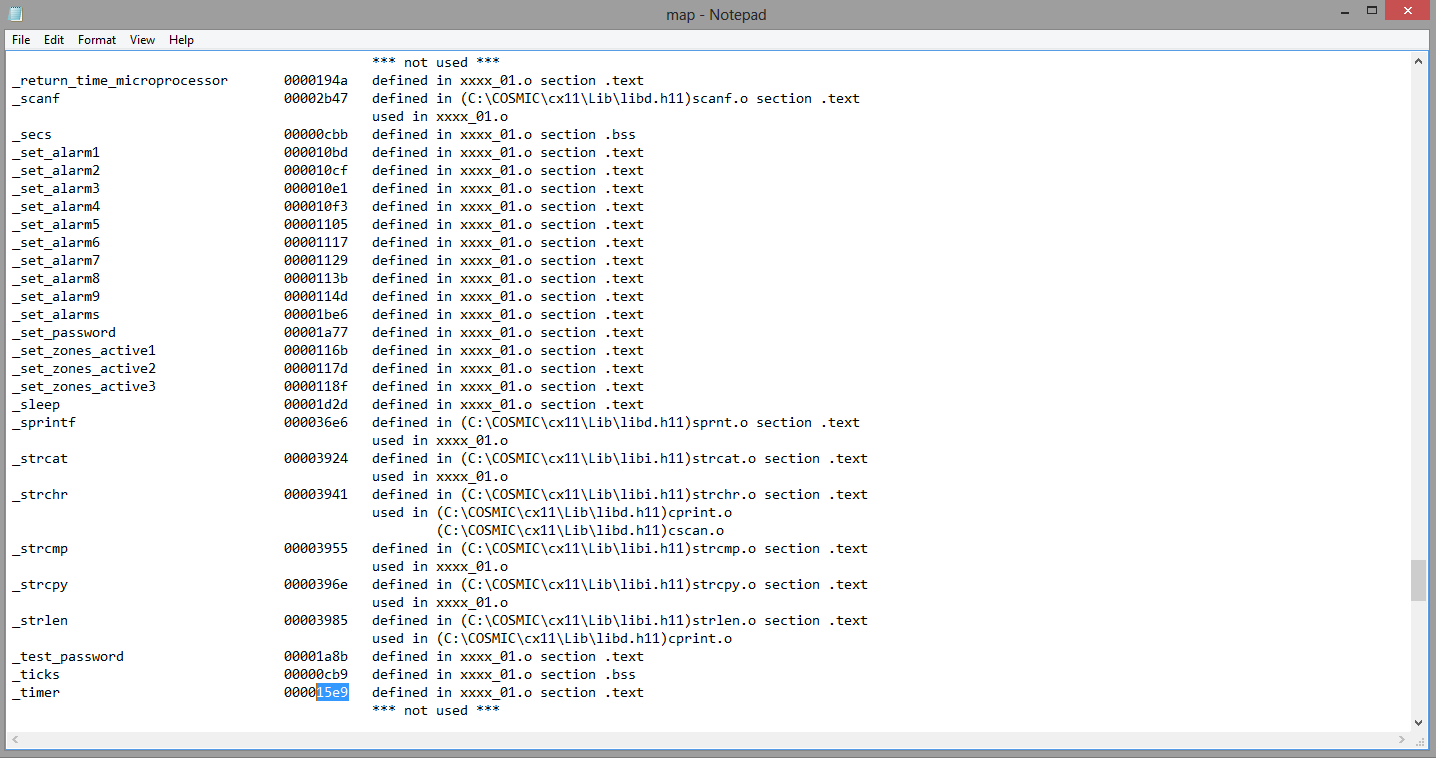


For porting, only few lines should be change for automatically check port a, port e and port g.

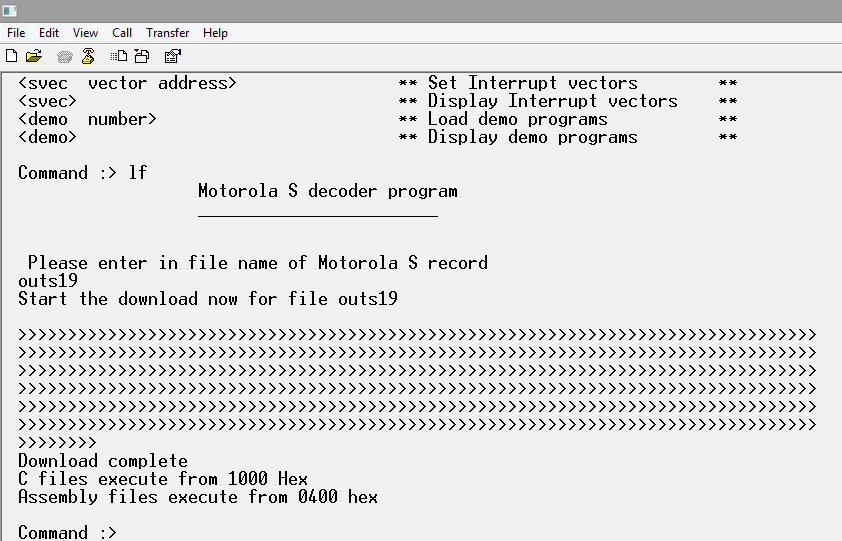
**Testing:**



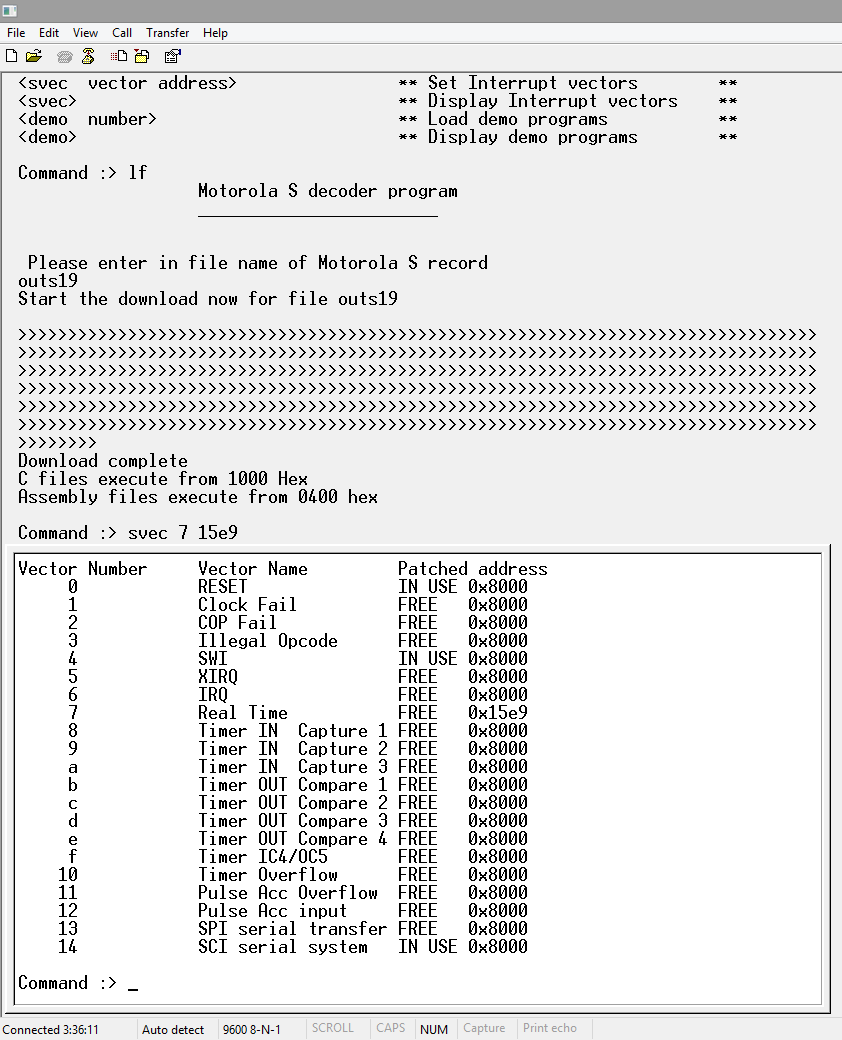
You compile your C file with Cram



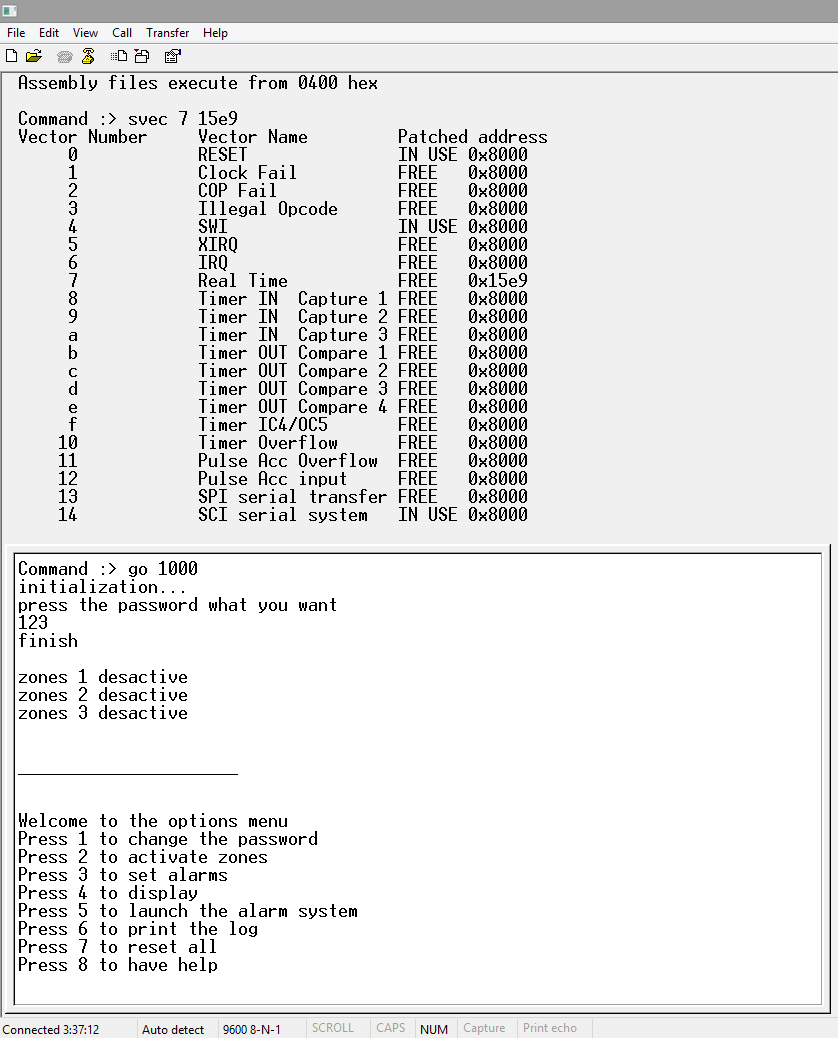
After compile with Cram you should see the map file to see \_timer



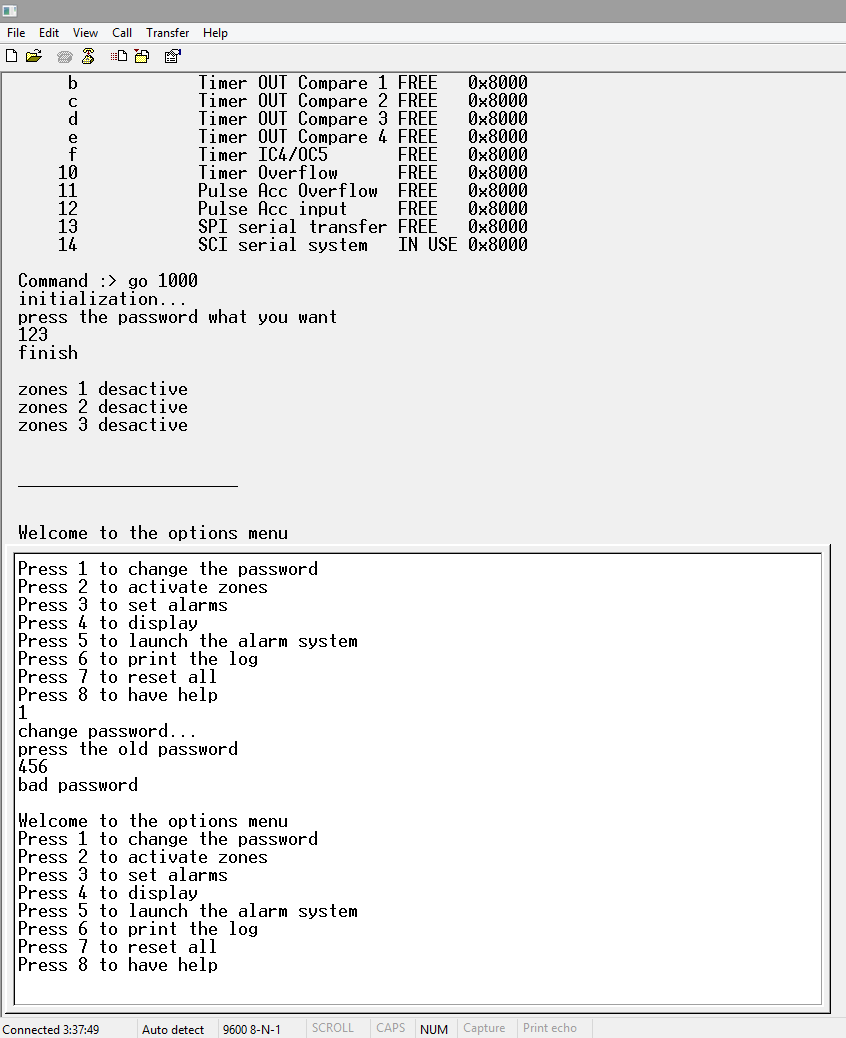
You should load the program in the board.



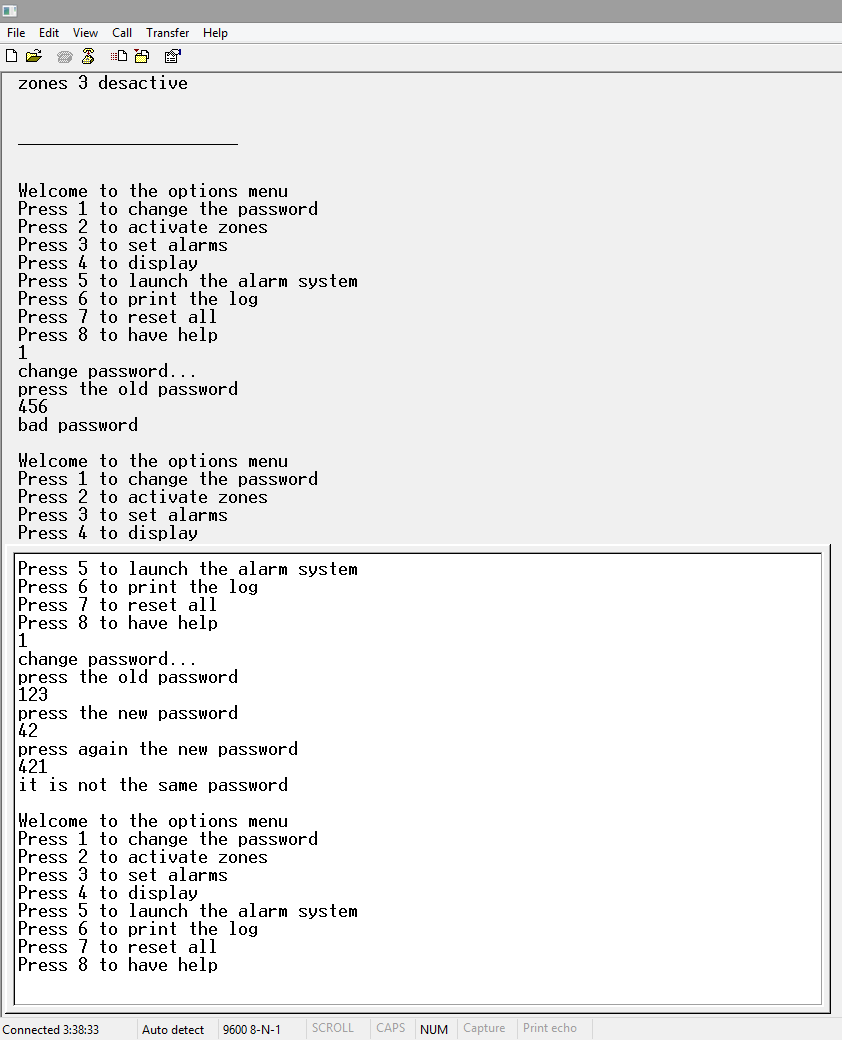
Press svec 7 and the \_timer number for change svec.



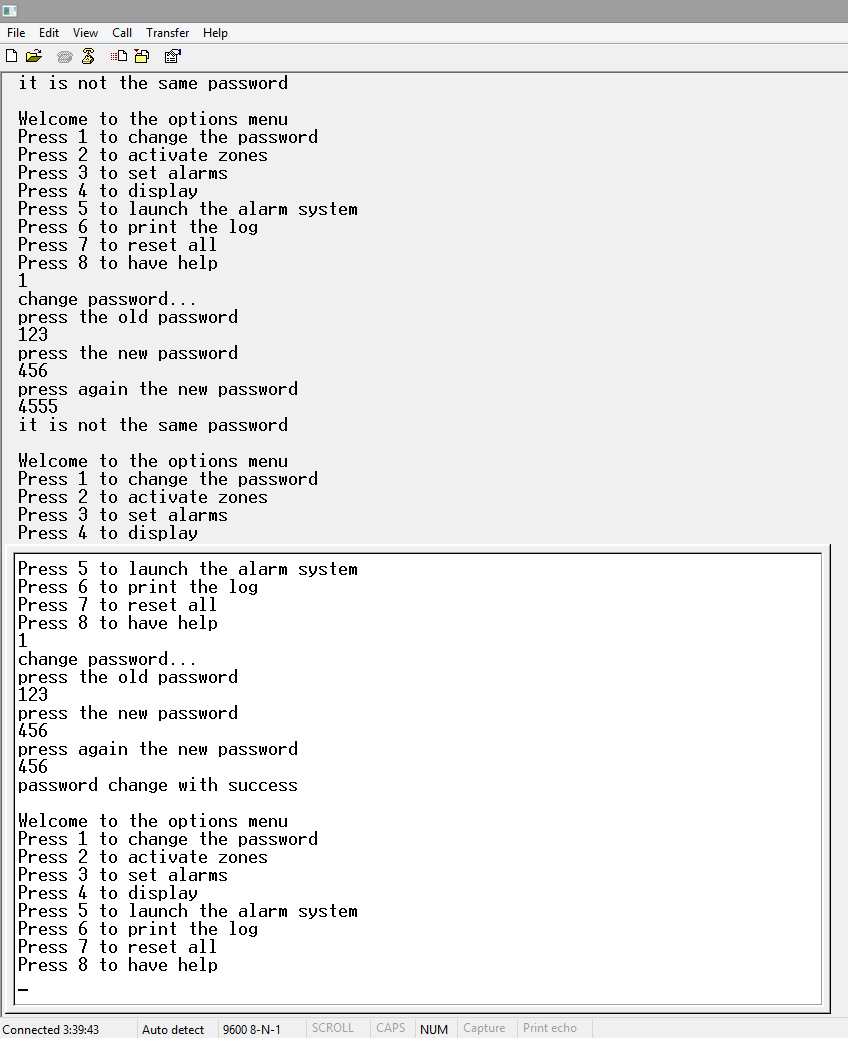
Launch the program, press your password and you can see the menu



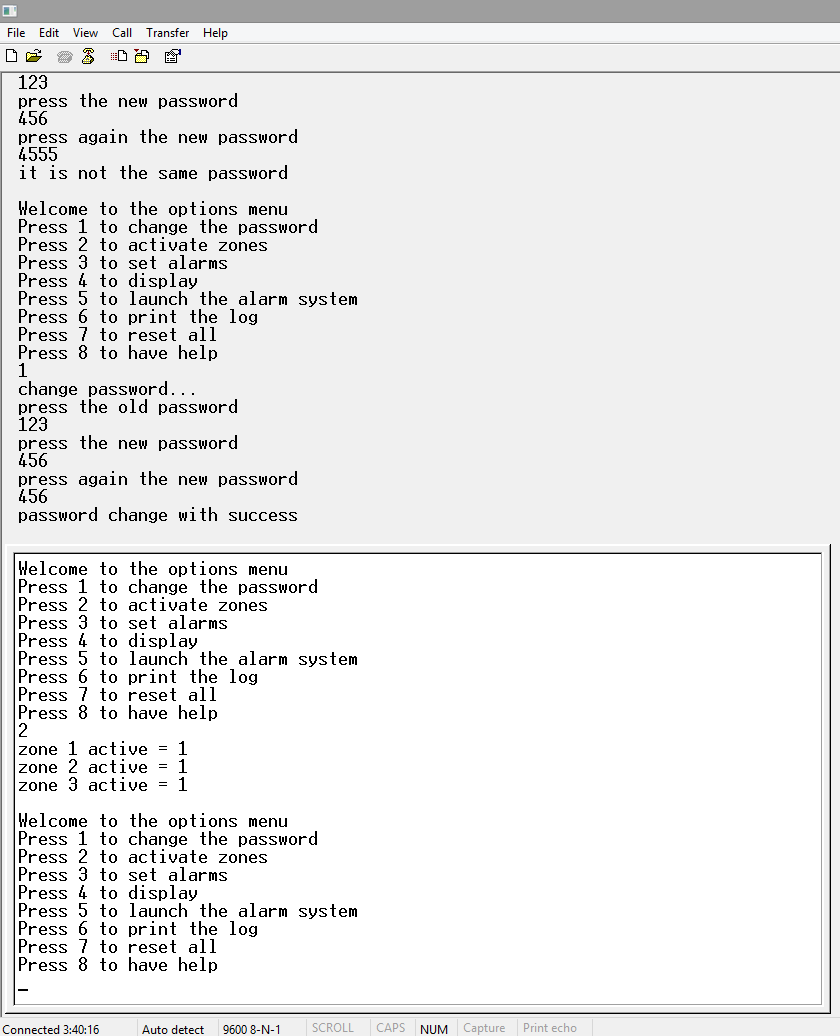
If you press 1 you can change the password but if you don't have the old password you can't.



If you want change the password but went you type again the new password you fail the password don't change.



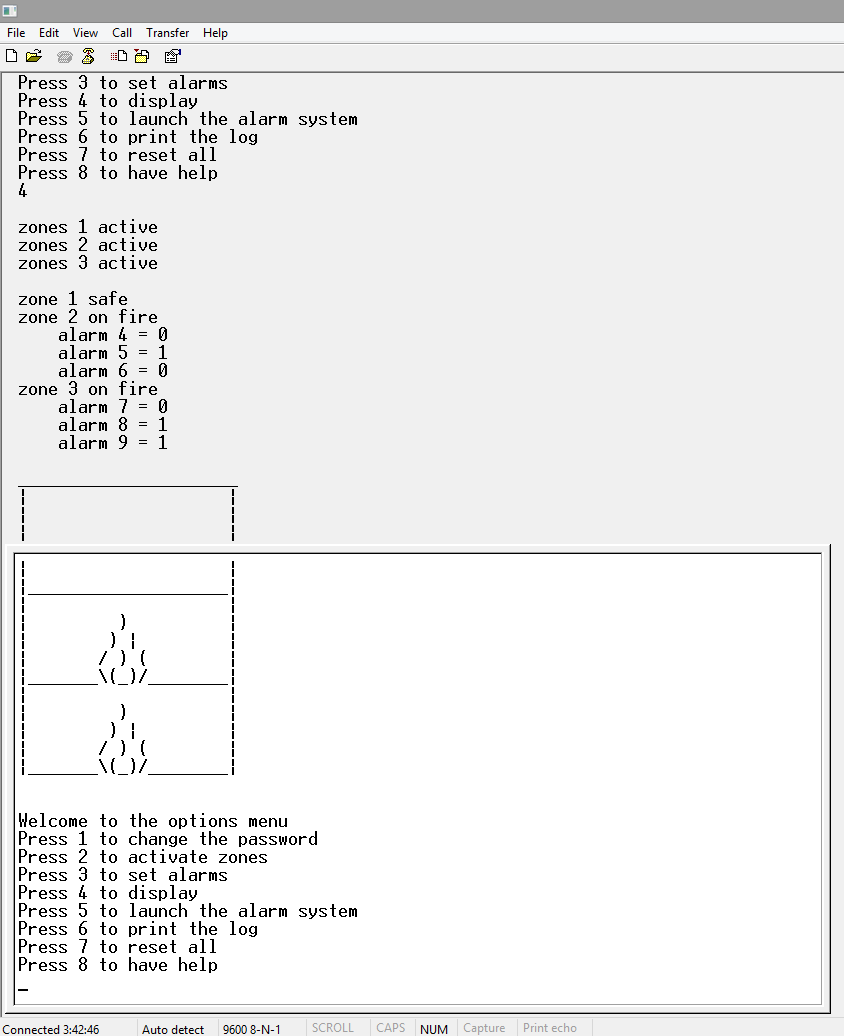
But if you don't make an error the password change.



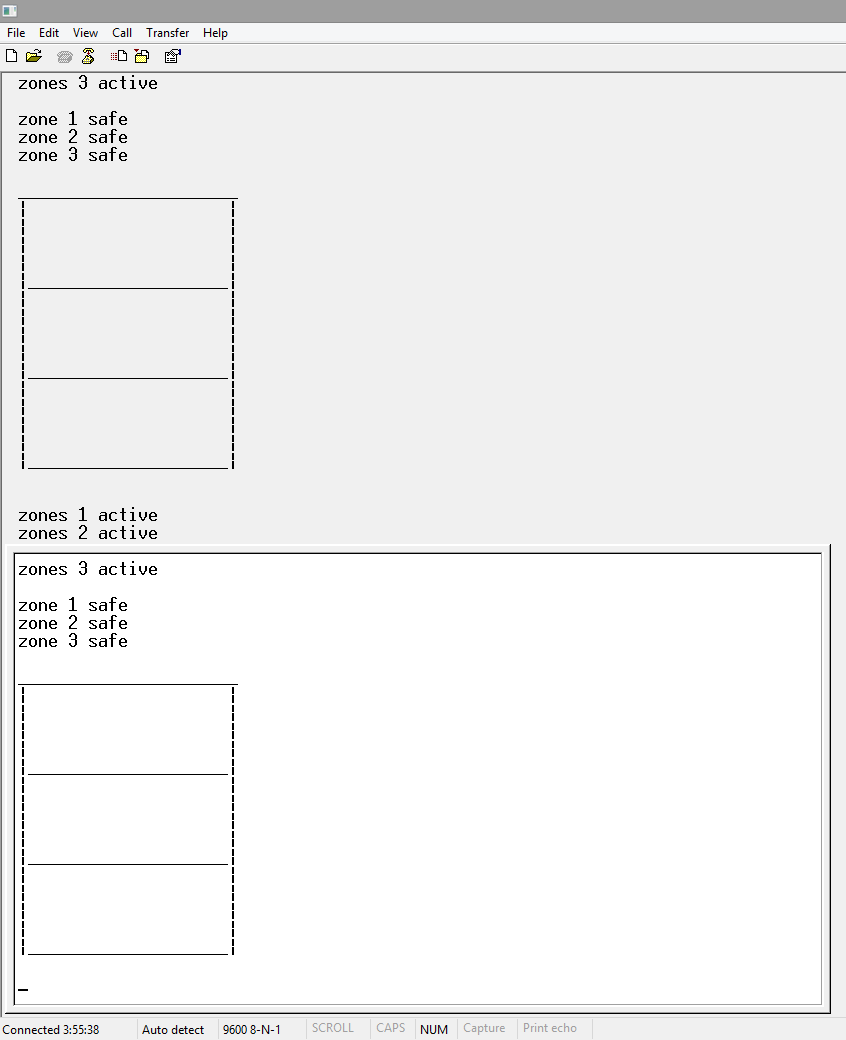
With option 2 you can change the zones which are activate.

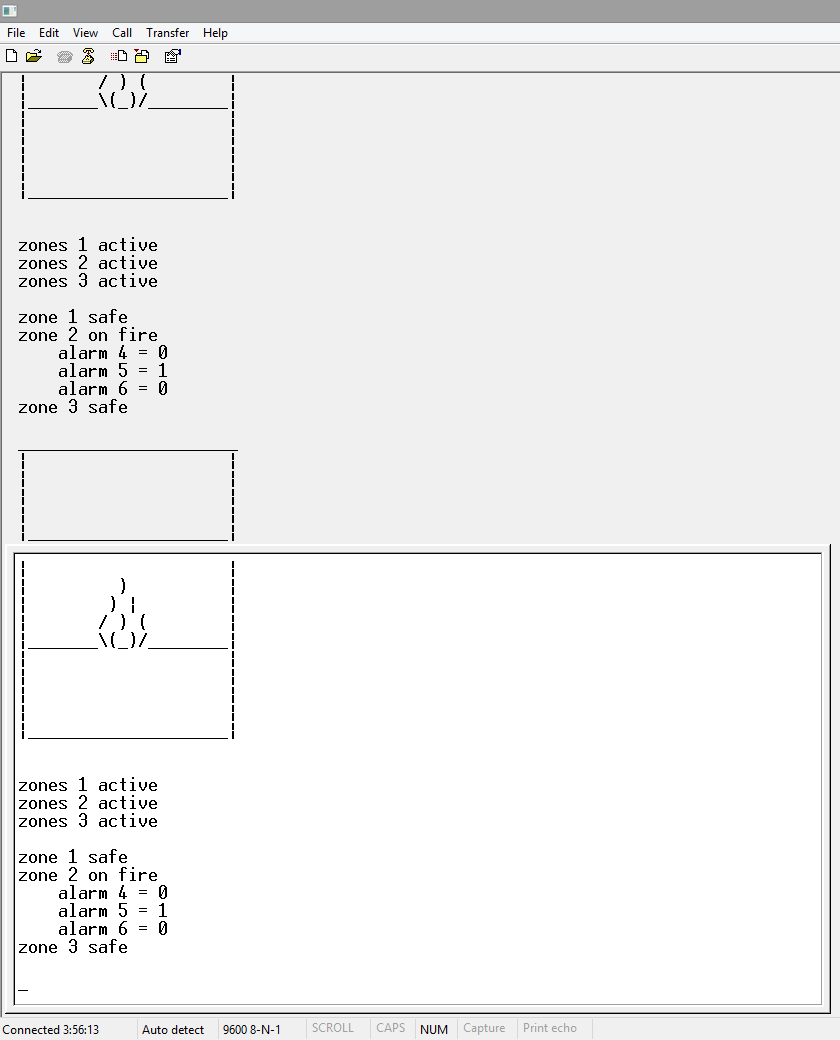


With option 3 you can change the value of the alarms.

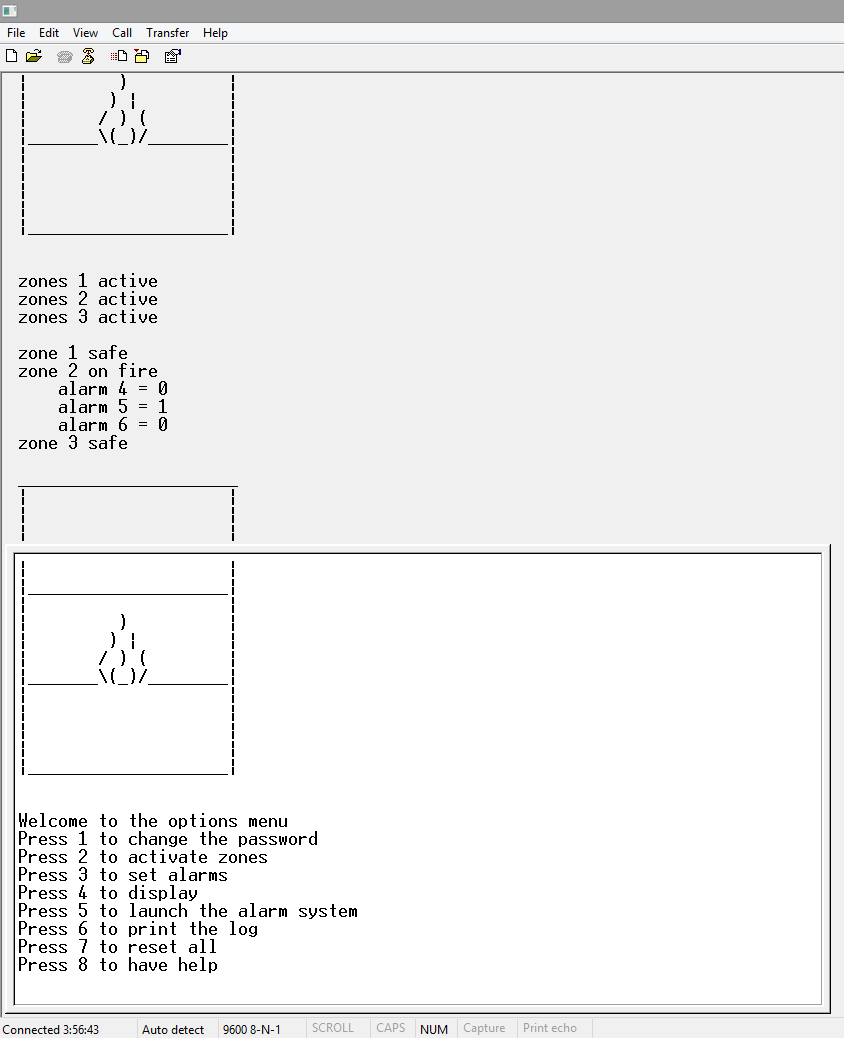


With option 4 you can display all information about zones and alarms.

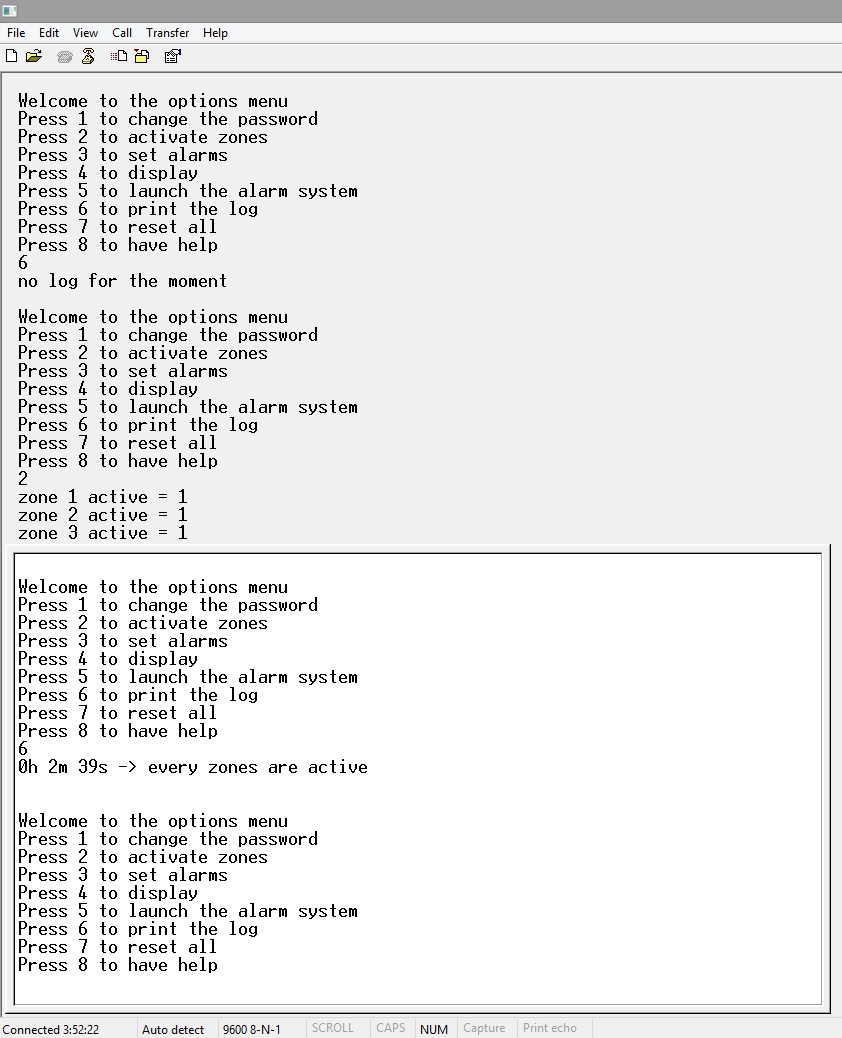


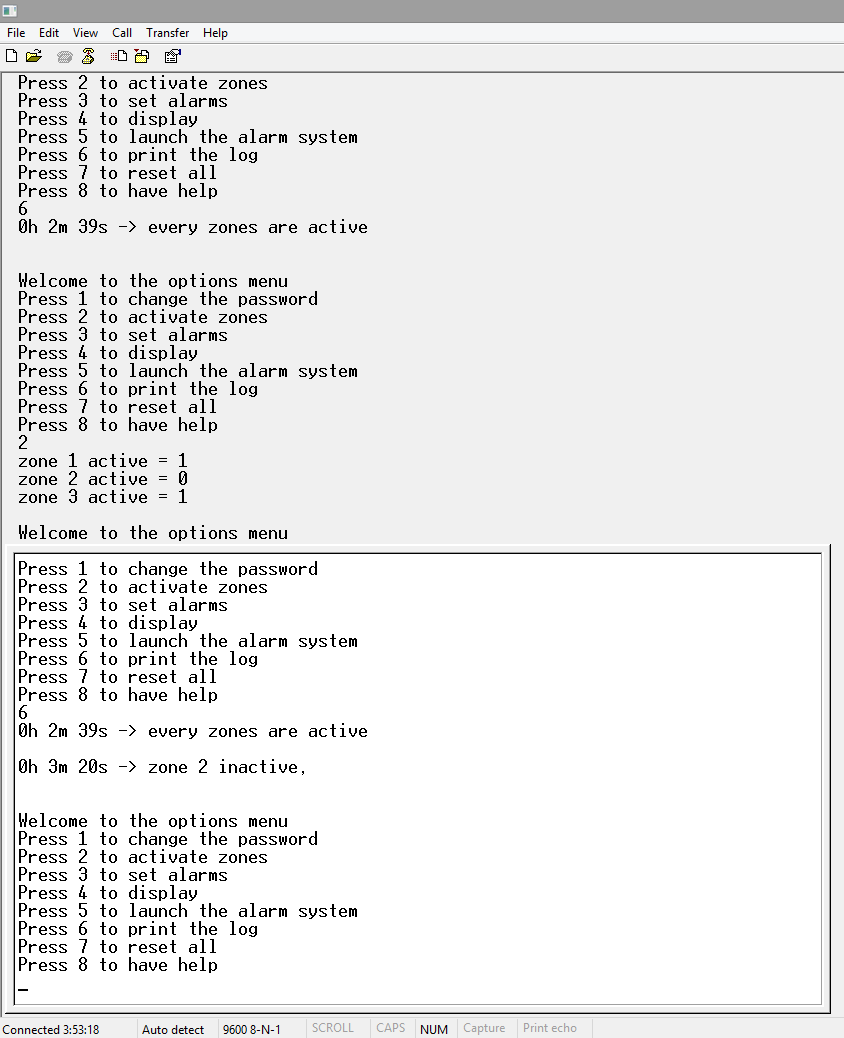


With option 5 you can launch the alarm system.

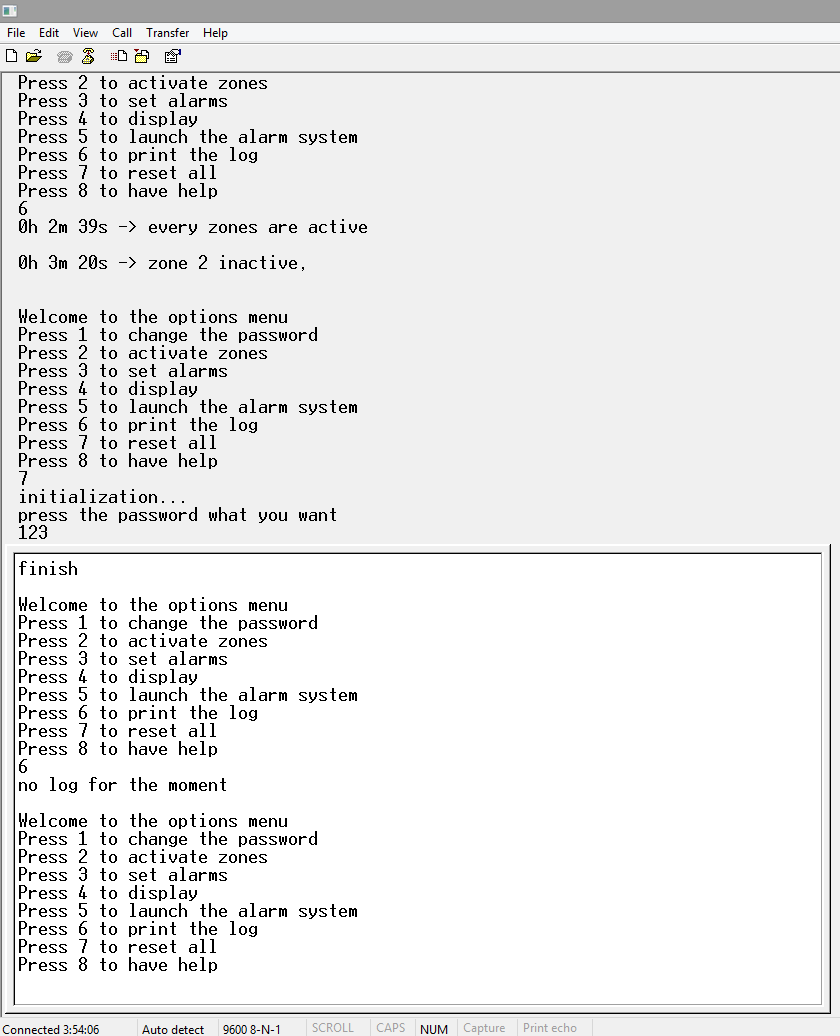


If you press any key it stop.

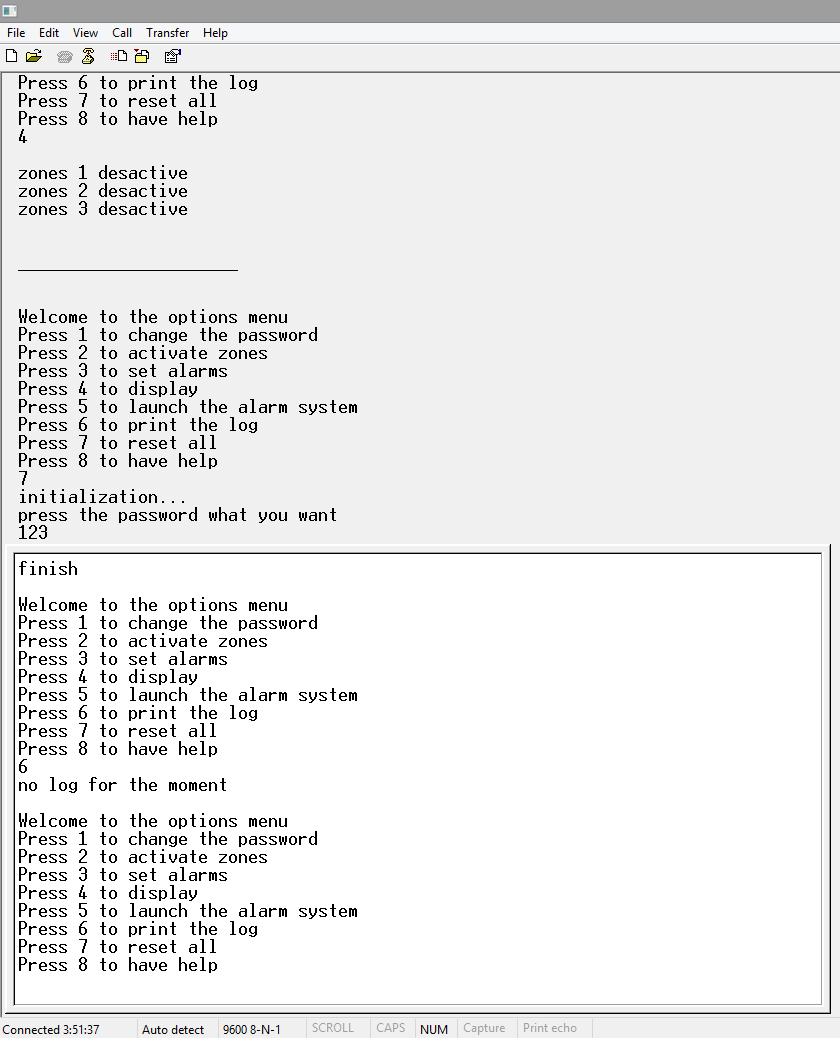




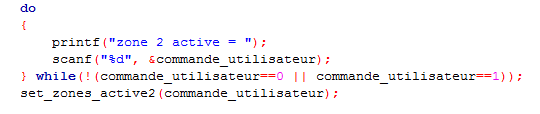
With option 6 you can print the log.



If there is no log it prints no log for the moment.



With option 7 you can reset all.



All the user input have this structure for have no bug.

**Conclusion:**

This lecture has taught me C programming, one of the fundamental programming language. It taught me to create programs for a specific hardware. And to port a Visual Studio Program for a micro-controller. I'm very interesting about embedded hardware, I have a Raspberry Pi and now I feel ready to program it.