

Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial

Anàlisi d'associacions de memristors per a la generació de bits aleatoris per aplicacions de seguretat hardware

ANNEXOS

ANNEX A:

Programes per a la realització de les corbes característiques

ANNEX B:

Programes per a la realització de la configuració sèrie

Autora:

Maitane Serrano Carranco

Directors:

Daniel Arumí Delgado i Rosa Rodríguez Montañés

Convocatòria:

Juny 2017



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



SUMARI

SUMARI	2
A. PROGRAMES PER A LA REALITZACIÓ DE LES CORBES	
CARACTERÍSTIQUES	3
A.1. Funció principal: b2912A_forming_dc.m	3
A.2. Funció que connecta l'ordinador a l'instrument: connecta_ins.m	8
A.3. Funció encarregada de guardar les dades capturades per l'smu a una variable: save_data.m.....	8
A.4. Funció per donar temps a que l'SMU acabi de fer les mesures: wait.m.....	8
A.5. Funció per dibuixar els gràfics amb les dades que s'han guardat amb les mesures a un document de text. Postprocés (canal 1): llegir_dades_dc.m.....	8
A.6. Funció per dibuixar els gràfics amb les dades que s'han guardat amb les mesures a un document de text. Postprocés (canal 2): llegir_dades_dc2.m.....	9
B. PROGRAMES PER A LA REALITZACIÓ DE LA CONFIGURACIÓ	
SÈRIE	11
B.1. Funció principal: b2912A_serial_config.m.....	11
B.2. Funció encarregada de connectar l'ordinador a l'instrument: connecta_ins.m.....	14
B.3. Funció encarregada de guardar les dades capturades per l'SMU a una variable: save_data.m.....	15
B.4. Funció encarregada de configurar els polsos de voltatge que ha d'enviar l'smu als canals 1 i 2: voltage_pulse_config.m.....	15
B.5. Funció encarregada de configurar els polsos la mesura que farà l'smu del canal triat: measurement_config.m.....	15
B.6. Funció per donar temps a que l'smu acabi de fer les mesures: wait.m.....	16
B.7. Funció per dibuixar els gràfics amb les dades que s'han guardat amb les mesures a un document de text: llegir_dades_serie.m.....	16
B.8. Funció per dibuixar els gràfics a partir d'un document de text amb les dades (postprocés): llegir_dades_dc_serie_post_proc_4.m.....	19

A. PROGRAMES PER A LA REALITZACIÓ DE LES CORBES CARACTERÍSTIQUES

El programa que realitza les corbes característiques dels memristors està format per un fitxer principal de MATLAB (extensió “.m”) i que crida a la resta de subprogrames, els quals estan col·locats a fitxers de MATLAB diferents.

A.1. FUNCIÓ PRINCIPAL: B2912A_FORMING_DC.m

```
clear all; clc;
close all;
%*****definició del tipus de mesura a fer i parametres modificables *****%
forming=0; %1 - es fa forming **** 0 - no es fa forming
sweeps=1; %1 - es fan sweeps **** 0 - no es fan sweeps
cycleini=220; %número de cicle inicial
cyclefinal=319; %número de cicle final de la seqüència
vreset=-0.6; %voltatge maxim de RESET
vset=0.8; %voltatge maxim de SET

%*****definició de paràmetres*****%
vini=0.0; %voltatge inicial del sweep del forming
vfinal=1.0; %voltatge final del sweep del forming
step=0.025; %grao de voltatge
npoint=(vfinal-vini)/step+1.0; %nombre de punts, ajustat segons la variable step
npoint=50;
taper=2.5e-4; %Temps d'apertura
trig_delay=0e-3; %trigger delay
ntrigger=npoint; %number of measurements, màxim 10^5
Ifcomp=1e-6; %Corrent limit pel forming
Icompr=10e-3; %Corrent limit per RESET
Icomps=1e-3; %Corrent limit per SET

%*****Definir i connectar a l'instrument*****%
vg=conecta_ins(); %Function to connect to the b2912A

%*****Inicializacio de l'instrument*****%
fprintf(vg, '*CLS;*RST');
%*****Definició de l'escombrat al canal 1*****%
fprintf(vg, ':sour1:func:mode volt'); %Definició com a font de tensió
fprintf(vg, ':sour1:volt:mode swe'); %Definició com a sweep
fprintf(vg, ':sour1:volt:star %f', vini); %Valor inicial de l'escombrat
```

```

fprintf(vg,':sour1:volt:stop %f', vfinal); %Valor final de l'escombrat
fprintf(vg,':sour1:volt:poin %f', npoint); %nombre de punts
fprintf(vg,':sour1:func:mode volt'); %Definició com a font de tensió
fprintf(vg,':sour1:volt:mode swe'); %Definició com a sweep
fprintf(vg,':sour1:volt:star %f', vini); %Valor inicial de l'escombrat
fprintf(vg,':sour1:volt:stop %f', vfinal); %Valor final de l'escombrat
fprintf(vg,':sour1:volt:poin %f', npoint); %nombre de punts

%*****Definició de l'escombrat al canal 2*****%
fprintf(vg,':sour2:func:mode volt'); %Definició com a font de tensió
fprintf(vg,':sour2:volt:mode swe'); %Definició com a sweep
fprintf(vg,':sour2:volt:star %f', vini); %Valor inicial de l'escombrat
fprintf(vg,':sour2:volt:stop %f', vfinal); %Valor final de l'escombrat
fprintf(vg,':sour2:volt:poin %f', npoint); %nombre de punts

%*****Configuració dels polsos*****%
fprintf(vg,':sens1:func "curr"'); %Activa la funcio per a mesurar corrent en
CH1
fprintf(vg,':sens1:curr:prot %f',Ifcomp); %Icompliance
fprintf(vg,':sens2:func "curr"'); %Activa la funcio per a mesurar corrent en
CH2
fprintf(vg,':sens2:curr:prot %f',Ifcomp); %Icompliance

%*****Configuració de les mesures per al canal 1*****%
fprintf(vg,':sens1:curr:rang:auto on'); %activa la funció de rang automàtic
fprintf(vg,':sens1:curr:aper:auto 1'); %activa/desactiva la funcio d'apertura
automàtica
fprintf(vg,':sens1:curr:aper %f',taper); %Temps d'apertura si no esta posat en
automatic
fprintf(vg,':trig1:sour aint'); %Font del trigger: automatic
fprintf(vg,':trig1:tran:del %f',trig_delay); %trigger delay
fprintf(vg,':trig1:coun %i',ntrigger); %Nombre de triggers. ha de ser igual a
npoints

fprintf(vg,':SYST:ERROR:ALL?'); %error info
error=fscanf(vg) %Treiem la info per pantalla
%*****Configuració de les mesures per al canal 1*****%
fprintf(vg,':sens2:curr:rang:auto on'); %activa la funció de rang automàtic
fprintf(vg,':sens2:curr:aper:auto 1'); %activa/desactiva la funcio d'apertura
automàtica
fprintf(vg,':sens2:curr:aper %f',taper); %Temps d'apertura si no esta posat en
automatic

fprintf(vg,':trig2:sour aint'); %Font del trigger: automatic

```

```

fprintf(vg,':trig2:tran:del %f',trig_delay);    %trigger delay
fprintf(vg,':trig2:coun %i',ntrigger);        %Nombre de triggers. ha de ser igual a
npoints

%*****Configuració del format dels resultatats 1*****%
fprintf(vg,':form:elem:sens volt,curr,time,sour');

%*****%
%*****          FORMING          *****%
%*****%
cycle=cycleini;
if (forming==1)
    %*****Configuració del forming*****%
    fprintf(vg,':sour1:volt:stop %f', vfinal); %Valor final de l'escombrat
    fprintf(vg,':sour2:volt:stop %f', vfinal); %Valor final de l'escombrat
    fprintf(vg,':sens1:curr:prot %f',Ifcomp); %Icompliance
    fprintf(vg,':sens2:curr:prot %f',Ifcomp); %Icompliance
    %*****Habilitació de les sortides*****%
    fprintf(vg,':outp1 on'); %activa sortida 1
    fprintf(vg,':outp2 on'); %activa sortida 2
    %Execució de les acció
    fprintf(vg,':init (@1,2)'); %inicia accions per als canals seleccionats
    %Pausa si queden operacions pendents
    vg=wait(vg);
    %Captura de resultats
    [vg,data1,array1]=save_data(vg,1,cycle,'f');
    [vg,data2,array2]=save_data(vg,2,cycle,'f');

    %Pausa si queden operacions pendents
    vg=wait(vg);

    %Mirem si hi ha hagut algun error
    fprintf(vg,':SYST:ERROR:ALL?'); %error info
    error=fscanf(vg) %Treiem la info per pantalla
    cycle=cycle+1;
    %*****%
    %*****          DIBUIX DEL FORMING          *****%
    %*****%
    eval('V1 = array1(:,1)');
    eval('I1 = abs(array1(:,2))');
    eval('V2 = array2(:,1)');
    eval('I2 = abs(array2(:,2))');
    semilogy (V1,I1,'bo',V2,I2,'r*');title ('Forming');

```

```

ylabel ('I (A)', 'FontSize', 18, 'Fontname', 'Times new roman');
xlabel ('V (V)', 'FontSize', 18, 'Fontname', 'Times new roman');
legend ( {'MEMR1', 'MEMR2'}, 'Location', 'Southeast', 'FontSize', 16, 'Fontname', 'Times
new roman');
set(gca, 'fontsize', 14); ylim('auto'); xlim('auto');
saveas(gcf, 'forming.fig')
saveas(gcf, 'forming.jpg')
end
clear V1 I1 V2 I2 array1 array2 data1 data2;
j=cycle;

if(sweeps==1)

    for (cycle=j:cyclefinal)
        cycle = cycle
        %*****Configuració del reset*****%
        ntrigger=2*npoint;                %nombre de mesures, màxim 10^5
        fprintf(vg, ':sour1:volt:stop %f', vreset); %Valor final de l'escombrat
        fprintf(vg, ':sour1:volt:poin %f', npoint); %nombre de punts
        fprintf(vg, ':sour2:volt:stop %f', vreset); %Valor final de l'escombrat
        fprintf(vg, ':sour2:volt:poin %f', npoint); %nombre de punts
        fprintf(vg, ':sens1:curr:prot %f', Icompr); %Icompliance
        fprintf(vg, ':sens2:curr:prot %f', Icompr); %Icompliance
        fprintf(vg, ':sour1:swe:sta doub'); %Double stair
        fprintf(vg, ':sour2:swe:sta doub'); %Double stair
        fprintf(vg, ':trig1:coun %i', ntrigger); %Nombre de triggers
        fprintf(vg, ':trig2:coun %i', ntrigger); %Nombre de triggers
        %Execució de les acció
        fprintf(vg, ':init (@1,2)'); %inicia accions per als canals seleccionats
        %Pausa si queden operacions pendents
        vg=wait(vg);
        %Captura de resultats
        [vg, data1, arrayr1]=save_data(vg, 1, cycle, 'n');
        [vg, data2, arrayr2]=save_data(vg, 2, cycle, 'n');

        %*****Configuració del set*****%
        ntrigger=2*npoint;                %nombre de mesures, màxim 10^5
        fprintf(vg, ':sour1:volt:stop %f', vset); %Valor final de l'escombrat
        fprintf(vg, ':sour1:volt:poin %f', npoint); %nombre de punts
        fprintf(vg, ':sour2:volt:stop %f', vset); %Valor final de l'escombrat
        fprintf(vg, ':sour2:volt:poin %f', npoint); %nombre de punts
        fprintf(vg, ':sens1:curr:prot %f', Icomps); %Icompliance
        fprintf(vg, ':sens2:curr:prot %f', Icomps); %Icompliance
        fprintf(vg, ':sour1:swe:sta doub'); %Double stair

```

```

fprintf(vg,':sour2:swe:sta doub'); %Double stair
fprintf(vg,':trig1:coun %i',ntrigger); %Nombre de triggers
fprintf(vg,':trig2:coun %i',ntrigger); %Nombre de triggers

%Execució de les acció
fprintf(vg,':init (@1,2)'); %inicia accions per als canals seleccionats
%Pausa si queden operacions pendants
vg=wait(vg);
%Captura de resultats
[vg,data1,arrays1]=save_data(vg,1,cycle,'p');
[vg,data2,arrays2]=save_data(vg,2,cycle,'p');
%*****%
%***** PLOTTING SET&RST *****%
%*****%

eval('V1s = arrays1(:,1);');
eval('I1s = abs(arrays1(:,2));');
eval('V2s = arrays2(:,1);');
eval('I2s = abs(arrays2(:,2));');
eval('V1r = arrayr1(:,1);');
eval('I1r = abs(arrayr1(:,2));');
eval('V2r = arrayr2(:,1);');
eval('I2r = abs(arrayr2(:,2));');
semilogy (V1s,I1s,'b-',V2s,I2s,'r-',V1r,I1r,'g-',V2r,I2r,'k-');title
('SET&RST');
ylabel ('I (A)','FontSize',18,'Fontname','Times new roman');
xlabel ('V (V)','FontSize',18,'Fontname','Times new roman');
legend
({'MEMR1s','MEMR2s','MEMR1r','MEMR2r'},'Location','Best','FontSize',16,'Fontname','Times new roman');
set(gca,'fontsize',14); ylim('auto'); xlim('auto');
saveas(gcf,'set-rst.fig')
hold on;
end
saveas(gcf,'set-rst.jpeg')
end
%Deshabilitació de les sortides
fprintf(vg,':outp1 off'); %desactiva sortida 1
fprintf(vg,':outp2 off'); %desactiva sortida 2
fclose(vg);
clear V1s I1s V2s I2s V1r I1r V2r I2r arrays1 arrays2 arrayr1 arrayr2
var = 'end' %avisa per pantalla que s'ha acabat el test

```

A.2. FUNCIO QUE CONNECTA L'ORDINADOR A L'INSTRUMENT: CONNECTA_INS.m

```
%Definint i connectant a l'instrument
function vg=conecta_ins()
vg=visa('agilent','GPIB0::23::INSTR');
set(vg,'EOSMOD','read&write');
set(vg,'EOSCharCode','LF');
vg.InputBufferSize = 955100;
vg.OutputBufferSize = 5100;
fopen(vg);
```

A.3. FUNCIO ENCARREGADA DE GUARDAR LES DADES CAPTURADES PER L'SMU A UNA VARIABLE: SAVE_DATA.M

```
function [vg,data,array]=save_data(vg,channel,cycle,prova)
aux=sprintf(':fetc:arr? (@%d)',channel);
fprintf(vg,'%s',aux); %retorna el array amb la mesura
data=fscanf(vg); %array amb tants elements com comptes de trigger
array=strsplit(data,',');
array=str2double(array); %cell a double
array=vec2mat(array,4); %cada mesura en una filera diferent
fid1 = ['Cycle_' num2str(cycle) prova '_R' num2str(channel) '.txt']; %indiquem número
cicle, tipus de prova (forming, set, reset) i canal
dlmwrite(fid1,array,'delimiter','\t','precision','%.6e');
```

A.4. FUNCIO PER DONAR TEMPS A QUE L'SMU ACABI DE FER LES MESURES: WAIT.m

```
function vg=wait(vg)
fprintf(vg,'*OPC?'); %monitoritza les operacions pendents. Si =1 - no n'hi ha
wait=fscanf(vg,'%d');
while(wait~=1) %espera si hi ha operacions pendents
wait=fscanf(vg,'%d');
end
```

A.5. FUNCIO PER DIBUIXAR ELS GRÀFICS AMB LES DADES QUE S'HAN GUARDAT AMB LES MESURES A UN DOCUMENT DE TEXT. POSTPROCÉS (CANAL 1): LLEGIR_DADES_DC.m

```
close all;
```

```

ini=0; %cicle a partir del qual es vol dibuixar
final=19; %cicle fins el qual es vol dibuixar
for (cycle=ini:final)
    fid1 = ['Cycle_' num2str(cycle) 'n_R1.txt']; %indiquem número cicle, tipus de prova
    (forming, set, reset) i canal
    dataR1=dlmread(fid1);
    eval(['V1n = dataR1(:,1);']);
    eval(['I1n = dataR1(:,2);']);
    eval(['t1n = dataR1(:,3);']);
    I1n=abs(I1n);
    fid1 = ['Cycle_' num2str(cycle) 'p_R1.txt']; %indiquem número cicle, tipus de prova
    (forming, set, reset) i canal
    clear dataR1;
    dataR1=dlmread(fid1);
    eval(['V1p = dataR1(:,1);']);
    eval(['I1p = dataR1(:,2);']);
    eval(['t1p = dataR1(:,3);']);
    I1p=abs(I1p);

    semilogy (V1p,I1p,'b-',V1n,I1n,'r-');
    if(cycle==ini)
        hold on;
    end
    clear dataR1, V1p,I1p,V1n,I1n;
end
title ('Corbes característiques');
ylabel ('I (A)', 'FontSize',18,'Fontname','Times new roman');
xlabel ('V (V)', 'FontSize',18,'Fontname','Times new roman');
set(gca, 'fontsize',14); ylim('auto'); xlim('auto');
saveas(gcf, 'ml.fig')

```

A.6. FUNCIÓ PER DIBUIXAR ELS GRÀFICS AMB LES DADES QUE S'HAN GUARDAT AMB LES MESURES A UN DOCUMENT DE TEXT. POSTPROCÉS (CANAL 2): LLEGIR_DADES_DC2.m

```

close all;
ini=0; %cicle a partir del qual es vol dibuixar
final=19; %cicle fins el qual es vol dibuixar
for (cycle=ini:final)
    fid1 = ['Cycle_' num2str(cycle) 'n_R2.txt']; %indiquem número cicle, tipus de prova
    (forming, set, reset) i canal
    dataR1=dlmread(fid1);
    eval(['V1n = dataR1(:,1);']);

```

```
eval(['I1n = dataR1(:,2);']);
eval(['t1n = dataR1(:,3);']);
I1n=abs(I1n);
fid1 = ['Cycle_' num2str(cycle) 'p_R2.txt']; %indiquem número cicle, tipus de prova
(forming, set, reset) i canal
clear dataR1;
dataR1=dlmread(fid1);
eval(['V1p = dataR1(:,1);']);
eval(['I1p = dataR1(:,2);']);
eval(['t1p = dataR1(:,3);']);
I1p=abs(I1p);

semilogy (V1p,I1p,'b-',V1n,I1n,'r-');
if(cycle==ini)
    hold on;
end
clear dataR1, V1p,I1p,V1n,I1n;
end
title ('Corbes característiques');
ylabel ('I (A)','FontSize',18,'Fontname','Times new roman');
xlabel ('V (V)','FontSize',18,'Fontname','Times new roman');
set(gca,'fontsize',14); ylim([1e-8 1e-2]); xlim('auto');
saveas(gcf,'m2.fig')
```

B. PROGRAMES PER A LA REALITZACIÓ DE LA CONFIGURACIÓ SÈRIE

El programa que realitza la configuració sèrie dels memristors està format per un fitxer principal de MATLAB (extensió “.m”) i que crida a la resta de subprogrames, els quals estan col·locats a fitxers de MATLAB diferents.

B.1. FUNCIÓ PRINCIPAL: B2912A_SERIAL_CONFIG.m

```
clear all; clc;
close all;
%constants
vrpos = +0.1;      %valor lectura pols positiu -> 100 mV
vrneg = -0.1;      %valor lectura pols negatiu -> -100 mV
vwpos = +1.7;      %valor escriptura pols positiu (SET) -> 1 V
vwneg = -0.6;      %valor lectura pols negatiu (RST)-> -1 V
cicleini=11;       %numero de cicle inicial
ciclefi=15;        %numero de cicle inicial

%variables
vbase=0.0;         %valor de la tensió del nivell baix del pols
vpulse=vwpos;      %valor de la tensió del nivell alt del pols
delay=1e-3;         %retard en l'inici del pols
width=2e-3;         %amplada del pols, mínim 50us amb resolució d'1us
Icomps=1e-3;        %Current compliance set. valor original 100e-6
Icompr=10e-3;       %Current compliance rst. valor original 100e-6
Irange=0.0001;     %rang de mesura en el cas que no se seleccioni el mode auto
trig_delay=0e-3;   % trigger delay
acq_delay=1.2e-3;  %acquire delay
taper=2.5e-4;      %temps apertura
N=1;               %nombre de vegades que repetim el pols/mesura, màxim 10^5
interval=10e-3;    %trigger timer interval - signal period, minimum 100u
meas='curr';       %mesura, de tensió 'volt' o de corrent 'curr'
v1=[];            %inicialitzacio vectors recolectors de dades
v2=[];            %inicialitzacio vectors recolectors de dades
i1=[];            %inicialitzacio vectors recolectors de dades
i2=[];            %inicialitzacio vectors recolectors de dades

%Definir i connectar a l'instrument
vg=conecta_ins(); %Function to connect to the b2912A
```

```

%Inicialització de l'instrument
fprintf(vg, '*CLS; *RST');

%Defició dels polsos
fprintf(vg, ':sour1:func:mode volt'); %Definició com a font de tensió
fprintf(vg, ':sour2:func:mode curr'); %Definició com a font de corrent
fprintf(vg, ':sour1:func:shap puls'); %Definició com a pols
fprintf(vg, ':sour2:func:shap dc'); %Definició com a valor constant

%Configuració dels polsos
vg=voltage_pulse_config(vg,1,vbase,vpulse,delay,width,Icomps); %Paràmetres del pols per
CH1. CH2 es I=0 ctt -configuració per defecte

%Configuració de les mesures
vg=measurement_config(vg,1,meas,Irange,taper,trig_delay,acq_delay,N, interval);
meas = 'volt'; %per CH2 vull mesurar tensio
vg=measurement_config(vg,2,meas,Irange,taper,trig_delay,acq_delay,N, interval); %Irange
s'ignora a la f

%Habilitació de les sortides
fprintf(vg, ':outp1 on'); %enable output
fprintf(vg, ':outp2 on'); %enable output

%obrir fitxer captura dades
f1 = fopen('serial_config_1.txt','w');
f2 = fopen('serial_config_2.txt','w');

for i=cicleini:ciclefi
    if (mod(i,5)) == 0 && (i > 4)
        i
    end
    vg=wait(vg);
    %%%SET%%%
    %Configuració dels polsos
    vpulse =vwpos;
    vg=voltage_pulse_config(vg,1,vbase,vpulse,delay,width,Icomps); %Paràmetres del pols
    vg=wait(vg);
    %Execució de les acció
    fprintf(vg, ':init (@1,2)'); %iniciar accions en canals seleccionats
    %Pausa si queden operacions pendents
    vg=wait(vg);
    %Captura de resultats
    meas='curr';
    [vg,datai1,i1]=save_data(vg,1,meas,i1);

```

```
[vg, datai2, i2]=save_data(vg, 2, meas, i2);
meas='volt';
[vg, datav1, v1]=save_data(vg, 1, meas, v1);
[vg, datav2, v2]=save_data(vg, 2, meas, v2);
fprintf(f1, '%s\t%s\n', v1(length(v1)), i1(length(i1)));
fprintf(f2, '%s\t%s\n', v2(length(v2)), i2(length(i2)));

#####readSET#####
%Configuració dels polsos
vpulse =vrpos;
vg=voltage_pulse_config(vg, 1, vbase, vpulse, delay, width, Icomps); %Paràmetres del pols
vg=wait(vg);
%Execució de les acció
fprintf(vg, ':init (@1,2)'); %iniciar accions en canals seleccionats
%Pausa si queden operacions pendents
vg=wait(vg);
%Captura de resultats
meas='curr';
[vg, datai1, i1]=save_data(vg, 1, meas, i1);
[vg, datai2, i2]=save_data(vg, 2, meas, i2);
meas='volt';
[vg, datav1, v1]=save_data(vg, 1, meas, v1);
[vg, datav2, v2]=save_data(vg, 2, meas, v2);
fprintf(f1, '%s\t%s\n', v1(length(v1)), i1(length(i1)));
fprintf(f2, '%s\t%s\n', v2(length(v2)), i2(length(i2)));

#####RESET#####
%Configuració dels polsos
vpulse =vwneg;
vg=voltage_pulse_config(vg, 1, vbase, vpulse, delay, width, Icompr); %Paràmetres del pols
vg=wait(vg);
%Execució de les acció
fprintf(vg, ':init (@1,2)'); %iniciar accions en canals seleccionats
%Pausa si queden operacions pendents
vg=wait(vg);

%Captura de resultats
meas='curr';
[vg, datai1, i1]=save_data(vg, 1, meas, i1);
[vg, datai2, i2]=save_data(vg, 2, meas, i2);
meas='volt';
[vg, datav1, v1]=save_data(vg, 1, meas, v1);
[vg, datav2, v2]=save_data(vg, 2, meas, v2);
fprintf(f1, '%s\t%s\n', v1(length(v1)), i1(length(i1)));
```

```

fprintf(f2, '%s\t%s\n', v2(length(v2)), i2(length(i2)));

%%%%%readRST%%%%%
%Configuració dels polsos
vpulse =vrneg;
vg=voltage_pulse_config(vg,1,vbase,vpulse,delay,width,Icompr); %Paràmetres del pols
vg=wait(vg);
%Execució de les acció
fprintf(vg,':init (@1,2)'); %iniciar accions en canals seleccionats
%Pausa si queden operacions pendents
vg=wait(vg);
%Captura de resultats
meas='curr';
[vg,datai1,i1]=save_data(vg,1,meas,i1);
[vg,datai2,i2]=save_data(vg,2,meas,i2);
meas='volt';
[vg,datav1,v1]=save_data(vg,1,meas,v1);
[vg,datav2,v2]=save_data(vg,2,meas,v2);
fprintf(f1, '%s\t%s\n\n', v1(length(v1)), i1(length(i1)));
fprintf(f2, '%s\t%s\n\n', v2(length(v2)), i2(length(i2)));
vg=wait(vg);
end

%Pausa si queden operacions pendents
vg=wait(vg);

%Mirem si hi ha hagut algun error
fprintf(vg,':SYST:ERROR:ALL?'); %error info
error=fscanf(vg) %Treiem la info per pantalla
%Deshabilitació de les sortides
fprintf(vg,':outp1 off'); %enable output
fprintf(vg,':outp2 off'); %enable output
fclose(vg);
%Calcul i plot de les resistències, Vm i corrents
llegir_dades_dc_serie(cicleini,ciclefi)
fclose('all');

```

B.2. FUNCIO ENCARREGADA DE CONNECTAR L'ORDINADOR A L'INSTRUMENT: CONNECTA_INS.m

És la mateixa que d'utilitzada per a fer les corbes característiques i el codi del programa es troba explicitat a l'apartat 0.

B.3. FUNCIÓ ENCARREGADA DE GUARDAR LES DADES CAPTURADES PER L'SMU A UNA VARIABLE: SAVE_DATA.m

```
function [vg,data,vector,temps]=save_data(vg,channel,meas,vector,temps,fitxer)
aux=sprintf(':fetc:arr:%s? (@%d)',meas,channel);
fprintf(vg,'%s',aux);           %retorna el array amb les mesures
data=fscanf(vg);                %array amb tants elements com comptes de trigger
value = str2double(cell2mat(strsplit(data,',')));
if value == 0
    value = 1e-30;
elseif value == ' '
    value = 1e-30;
elseif (value == 1) || (value == '1')
    value = 1.0000000000000001;
end
vector=[vector value];
```

B.4. FUNCIÓ ENCARREGADA DE CONFIGURAR ELS POLSOS DE VOLTATGE QUE HA D'ENVIAR L'SMU ALS CANALS 1 I 2: VOLTAGE_PULSE_CONFIG.m

```
function vg=voltage_pulse_config(vg,channel,vbase,vpulse,delay,width,Icomp)
fprintf(vg,':sour%i:volt %f',[channel,vbase]);
fprintf(vg,':sour%i:volt:trig %f',[channel,vpulse]);
fprintf(vg,':sour%i:puls:del %f',[channel,delay]);
fprintf(vg,':sour%i:puls:wid %f',[channel,width]);
fprintf(vg,':sens%i:curr:prot %f',[channel,Icomp]);
```

B.5. FUNCIÓ ENCARREGADA DE CONFIGURAR ELS POLSOS LA MESURA QUE FARÀ L'SMU DEL CANAL TRIAT: MEASUREMENT_CONFIG.m

```
function
vg=measurement_config(vg,channel,meas,rang,taper,trig_delay,acq_delay,N,interval)

fprintf(vg,':sens%i:func "%s"',[channel,meas]); %activa la funció de mesura triada
fprintf(vg,':sens%i:%s:rang:auto on',[channel,meas]); %activa/desactiva rang mesura
automàtic
fprintf(vg,':sens%i:%s:aper:auto 0',[channel,meas]); %temps d'apertura automàtic
desactivat
aux=sprintf(':sens%i:%s:aper %f',channel,meas,taper); %temps apertura, per defecte 0.1
fprintf(vg,'%s',aux);
fprintf(vg,':trig%i:tran:del %f',[channel,trig_delay]);
fprintf(vg,':trig%i:acq:del %f',[channel,acq_delay]);
fprintf(vg,':trig%i:coun %i',[channel,N]); %Nombre de mesures
```

```
fprintf(vg,':trig%i:tim %f',[channel,interval]); %interval del trigger timer, min. 100u
fprintf(vg,':trig%i:sour tim',channel); %font del trigger
```

B.6. FUNCIÓ PER DONAR TEMPS A QUE L'SMU ACABI DE FER LES MESURES:

WAIT.m

És la mateixa que d'utilitzada per a fer les corbes característiques i el codi del programa es troba explicitat a l'apartat A.4.

B.7. FUNCIÓ PER DIBUIXAR ELS GRÀFICS AMB LES DADES QUE S'HAN GUARDAT AMB LES MESURES A UN DOCUMENT DE TEXT:

LLEGIR_DADES_SERIE.m

```
function llegir_dades_dc_serie(cinicial,cfinal)
close all;
ini=cinicial; %els valor inicial de cicle que volem indicar en la grafica de la
configuració
final=cfinal; %els valor final de cicle que volem indicar en la grafica de la
configuració
cicle_ini=0;
cicle_final=(final-ini);

Vmnr=[]; %Vm rst en la lectura
Vmnw=[]; %Vm rst en l'escriptura
Vmpr=[]; %Vm set en la lectura
Vmpw=[]; %Vm set en l'escriptura
Ip=[]; %Iset
In=[]; %Irst
Rp1=[]; %Rset
Rn1=[]; %Rrst
Rp2=[]; %Rset
Rn2=[]; %Rrst

cicle_ini=cicle_ini+1;
cicle_final=cicle_final+1;
fid1=['serial_config_1.txt'];
fid2=['serial_config_2.txt'];
dataR1=dlmread(fid1);
dataR2=dlmread(fid2);
eval(['V1 = dataR1(:,1);']); %en el doc 1 tenim Vdd
eval(['I1 = dataR1(:,2);']); %en el doc 1 tenim I serie
eval(['Vm = dataR2(:,1);']); %en el doc 2 tenim Vm
```

```

for(cycle=cicle_ini:cicle_final)
    aux = 0;
    aux2 = 0;
    for(j=1:4)
        index=j+(cycle-1)*4;
        if (j == 1)
            Vmpw = [Vmpw Vm(index)]; %Vm escriptura set
        elseif (j==2)
            aux = abs((abs(Vl(index))-abs(Vm(index)))/I1(index));
            aux2 = abs(Vm(index)/I1(index));
            Ip=[Ip I1(index)]; %I en set
            Vmpr = [Vmpr Vm(index)]; %Vm lectura set
            if I1(index) ~= 0
                Rp1=[Rp1 aux];
                Rp2=[Rp2 aux2];
            else
                Rp1=[Rp1 1E20];
                Rp2=[Rp2 1E20];
            end
        elseif (j==3)
            Vmnw = [Vmnw Vm(index)]; %Vm escriptura rst
        else %j==4
            aux = abs((abs(Vl(index))-abs(Vm(index)))/I1(index));
            aux2 = abs(Vm(index)/I1(index));
            In=[In I1(index)]; % I en rst
            Vmnr = [Vmnr Vm(index)]; %Vm lectura rst
            if I1(index) ~= 0
                Rn1=[Rn1 aux];
                Rn2=[Rn2 aux2];
            else
                Rn1=[Rn1 1E20];
                Rn2=[Rn2 1E20];
            end
        end
    end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (cycle=ini:final)

    plot (cycle, In(cycle-ini+1),'g*',cycle, Ip(cycle-ini+1),'ro');
    hold on;

```

```

end
title ('Corrents en SET i RESET');
ylabel ('I (A)', 'FontSize', 18, 'Fontname', 'Times new roman');
xlabel ('Cicle', 'FontSize', 18, 'Fontname', 'Times new roman');
legend ( {'Ireset', 'Iset'}, 'Location', 'Best', 'FontSize', 14, 'Fontname', 'Times new
roman');
set(gca, 'fontsize', 14); ylim('auto'); xlim([ini, final]);
saveas(gcf, 'Lectura corrents.fig')
saveas(gcf, 'Lectura corrents.jpg')
close(gcf)
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (cycle=ini:final)
    semilogy (cycle, Rn1(cycle-ini+1), 'b*', cycle, Rp1(cycle-ini+1), 'go', cycle,
Rn2(cycle-ini+1), 'k+', cycle, Rp2(cycle-ini+1), 'rx');
    hold on;
end

title ('Resistències en SET i RESET');
ylabel ('R (Ohm)', 'FontSize', 18, 'Fontname', 'Times new roman');
xlabel ('Cicle', 'FontSize', 18, 'Fontname', 'Times new roman');
legend
( {'Rreset1', 'Rset1', 'Rreset2', 'Rset2'}, 'Location', 'Best', 'FontSize', 14, 'Fontname', 'Tim
es new roman');
set(gca, 'fontsize', 14); ylim('auto'); xlim([ini, final]);
saveas(gcf, 'Lectura resistencies_log.fig')
saveas(gcf, 'Lectura resistencies_log.jpg')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close(gcf)
hold off

for (cycle=ini:final)
    plot (cycle, Vmnr(cycle-ini+1), 'ro', cycle, Vmpr(cycle-ini+1), 'g*', cycle,
Vmnr(cycle-ini+1), 'b+', cycle, Vmpw(cycle-ini+1), 'kd');
    hold on;
end

title ('Configuració sèrie');
ylabel ('V (V)', 'FontSize', 18, 'Fontname', 'Times new roman');
xlabel ('Número de cicle', 'FontSize', 18, 'Fontname', 'Times new roman');
legend
( {'VrRST', 'VrSET', 'VwRST', 'VwSET'}, 'Location', 'Best', 'FontSize', 14, 'Fontname', 'Times
new roman');
set(gca, 'fontsize', 14); ylim('auto'); xlim([ini, final]);
saveas(gcf, 'Vm_serie.fig')

```

```
saveas(gcf,'Vm_serie.jpg')
hold off
close(gcf)
var_end = 'End'
```

B.8. FUNCIÓ PER DIBUIXAR ELS GRÀFICS A PARTIR D'UN DOCUMENT DE TEXT AMB LES DADES (POSTPROCÉS): LLEGIR_DADES_DC_SERIE_POST_PROC_4.m

```
close all;clear all;
cicle_ini=0; %cicle amb que es vol iniciar la numeracio
cicle_final=19; %cicle amb que es vol acabar la numeracio
ini=0;
final=(cicle_final-cicle_ini);
Ip=[]; %Iset
In=[]; %Irst
Rp1=[]; %Rset
Rn1=[]; %Rrst
Rp2=[]; %Rset
Rn2=[]; %Rrst
ini=ini+1;
final=final+1;
fid1 = ['serial_config_1.txt'];
fid2 = ['serial_config_2.txt'];
dataR1=dlmread(fid1);
dataR2=dlmread(fid2);
eval(['V1 = dataR1(:,1);']); %en el doc 1 tenim Vdd
eval(['I1 = dataR1(:,2);']); %en el doc 1 tenim I serie
eval(['Vm = dataR2(:,1);']); %en el doc 2 tenim Vm

for(cycle=ini:final)
    aux = 0;
    aux2 = 0;
    for(j=1:4)
        index=j+(cycle-1)*4;
        if (j==2)
            aux = abs((abs(V1(index))-abs(Vm(index)))/I1(index));
            aux2 = abs(Vm(index)/I1(index));
            Ip=[Ip I1(index)];
            if I1(index) ~= 0
                Rp1=[Rp1 aux];
                Rp2=[Rp2 aux2];
            else
                Rp1=[Rp1 1E20];
                Rp2=[Rp2 1E20];
            end
        end
    end
end
```

```

        end
    elseif (j==4)
        aux = abs((abs(V1(index))-abs(Vm(index)))/I1(index));
        aux2 = abs(Vm(index)/I1(index));
        In=[In I1(index)];
        if I1(index) ~= 0
            Rn1=[Rn1 aux];
            Rn2=[Rn2 aux2];
        else
            Rn1=[Rn1 1E20];
            Rn2=[Rn2 1E20];
        end
    end
end
end
end

for (cycle=cicle_ini:cicle_final)
    plot (cycle, In(cycle-cicle_ini+1),'g*',cycle,Ip(cycle-cicle_ini+1),'ro');
    hold on;
end
title ('Corrents en SET i RESET');
ylabel ('I (A)','FontSize',18,'Fontname','Times new roman');
xlabel ('Cicle','FontSize',18,'Fontname','Times new roman');
legend (({'Ireset','Iset'},'Location','Best','FontSize',16,'Fontname','Times new roman'));
set(gca,'fontsize',14); ylim('auto'); xlim([ini-1,final-1]);
saveas(gcf,'Lectura corrents.fig')
saveas(gcf,'Lectura corrents.jpg')
close(gcf)
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (cycle=cicle_ini:cicle_final)
    semilogy (cycle, Rn1(cycle-cicle_ini+1),'b*',cycle, Rp1(cycle-cicle_ini+1),'go',
cycle, Rn2(cycle-cicle_ini+1),'k+',cycle, Rp2(cycle-cicle_ini+1),'rx');
    hold on;
end
title ('Resistències en SET i RESET');
ylabel ('R (Ohm)','FontSize',18,'Fontname','Times new roman');
xlabel ('Cicle','FontSize',18,'Fontname','Times new roman');
legend (({'Rreset1','Rset1','Rreset2','Rset2'},'Location','Best','FontSize',16,'Fontname','Times new roman'));
set(gca,'fontsize',14); ylim('auto'); xlim([ini-1,final-1]);
saveas(gcf,'Lectura resistencies.fig')

```



```
saveas(gcf,'Lectura resistencies.jpg')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (cycle=cicle_ini:cicle_final)
    semilogy (Rn1(cycle-cicle_ini+1),Rn2(cycle-cicle_ini+1),'k*');
    hold on;
end
title ('R2 vs R1 RST');
ylabel ('R2 (Ohm)','FontSize',18,'Fontname','Times new roman');
xlabel ('R1','FontSize',18,'Fontname','Times new roman');
set(gca,'fontsize',14); ylim('auto'); xlim('auto');
saveas(gcf,'R2-R1_rst.fig')
saveas(gcf,'R2-R1_rst.jpg')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for (cycle=cicle_ini:cicle_final)
    semilogy (Rp1(cycle-cicle_ini+1),Rp2(cycle-cicle_ini+1),'k*');
    hold on;
end
title ('R2 vs R1 SET');
ylabel ('R2 (Ohm)','FontSize',18,'Fontname','Times new roman');
xlabel ('R1','FontSize',18,'Fontname','Times new roman');
set(gca,'fontsize',14); ylim('auto'); xlim('auto');
saveas(gcf,'R2-R1_set.fig')
saveas(gcf,'R2-R1_set.jpg')

var = 'End'
```