II. LLISTA DELS PROGRAMES DEL SISTEMA EXPERIMENTAL AL PC I AL DSP.

A continuació es detalla la llista dels programes que s'executen sobre el PC i sobre el DSP.

Programa **Sensor1.c** (annex II.1). Programa desenvolupat en llenguatge C que s'executa sobre el PC i que realitza les funcions següents:

- Inicialitza tot el sistema.
- Envia el programa a executar sobre el DSP.
- Fa la regulació de tots els llaços de control.
- Emmagatzema totes les dades obtingudes.
- Genera la consigna de velocitat de tipus graó.

Aquest programa ha estat implementat amb l'objectiu de posar a prova el sistema de control *sensorless* implementat davant consignes de tipus graó de velocitat. També sobre aquest programa, un cop arrencat el motor, s'ha provat que el llaç de control implementat és robust davant canvis bruscos de parell de càrrega.

Programa **Rampes.c** (annex II.2). Programa desenvolupat en llenguatge C que s'executa sobre el PC i que realitza les funcions següents:

- Inicialitza tot el sistema.
- Envia el programa a executar sobre el DSP.
- Fa la regulació de tots els llaços de control.
- Emmagatzema totes les dades obtingudes.
- Genera la consigna de velocitat de tipus seqüència de rampes.

Aquest programa ha estat implementat amb l'objectiu de posar a prova el sistema de control *sensorless* implementat davant consignes de velocitat de tipus rampa, incloent-hi el canvi de sentit de gir.

Programa **Graons.c** (annex II.3). Programa desenvolupat en llenguatge C que s'executa sobre el PC i que realitza les funcions següents:

- Inicialitza tot el sistema.
- Envia el programa a executar sobre el DSP.
- Fa la regulació de tots els llaços de control.
- Emmagatzema totes les dades obtingudes.
- Genera dues consignes de velocitat de tipus graó una rere l'altra.

Aquest programa ha estat implementat amb l'objectiu de posar a prova el sistema de control *sensorless* implementat davant consignes de velocitat consecutives de tipus graó.

Programa **Modul.c** (annex II.4). Programa desenvolupat sobre el DSP, en llenguatge d'assemblador i llenguatge C, que realitza les funcions següents:

- Executa la modulació vectorial de l'inversor de potència.
- Realitza l'adquisició de senyals de velocitat de gir del motor i corrent de dues fases.

Aquest programa s'executa permanentment i fa la modulació vectorial en funció dels paràmetres que li són subministrats per qualsevol dels programes anteriors.

II.1 Programa “Sensor1.c”

```
/*********************************************************************/
/* SENSOR1.C  Programa que implementa un llaç de control FAM sense  */
/*          sensor sobre el motor                                    */
/* Estima el valor de la velocitat amb l'estimador RP                */
/* Estima el valor de la velocitat amb l'estimador RT                */
/* Implementa el sistema Fuzzy per filtrar i per promitjar           */
/* Es tanca el llaç de control amb la velocitat estimada             */
/* El sistema genera respostes a consignes tipus graó                */
/*********************************************************************/
/* Aquest programa genera temps de 100us i a més corregeix en el
cas de que una rutina trigui més de 100us, la següent triga menys */

/* Inclusió de llibreries necessàries per la rutina de mesura de temps
i la de control */
#include <dos.h>
#include <time.h>
#include <math.h>
#include <conio.h>

/* Inclusió de llibreries necessàries per al control del DSP */
#include "c:\mon32\bc45\tic32.h"
#include<stdio.h>
#include<stdlib.h>

/* Es defineixen les posicions de memòria de la DPRAM a  on hi ha les  */
/* consignes del modulador vectorial */

#define  VREF       0xc0000dl   // Mòdul de Vref (Modulador vectorial)
#define  VDC        0xc0000el   // Valor del bus de contínua
#define  WM         0xc0000fl   // Freqüència del senyal (en rad/s)
#define  ZERO       0xc000101   // Bit indicant pas per zero del motor
#define  STOP       0xc000121   // Bit de stop del modulador
#define  TZUS       0xc000141   // Tz
#define  TMIN1      0xc000151    // Tmin
#define  CONTATGE   0xc000161
#define  FaseVREF   0xc000171   // Fase del vector VREF
#define  sextant    0xc000191   // Sextant de treball
#define  SFVS       0xc0001al    // correcció de la fase del FAM
#define  START      0xc0001cl   // Indica que el sistema ha engegat
#define  Posicio    0xc000261
#define  Velocitat  0xc000271
#define  Wm0        0xc000441   // Posició DPRAM canal0
#define  Wm1        0xc000451   // Posició DPRAM canal1
#define  Wm2        0xc000461   // Posició DPRAM canal2
#define  Wm3        0xc000471   // Posició DPRAM canal3
#define  PI 3.141592654

/* Es defineix l'arxiu que serà llençat per a que corri en el DSP  */
#define  FILENAME "MODUL.OUT"


/* Constants */
float RpmRads=0.2094395;      // Conversió de rpm a Rad/s
float taorotor=0.0634288;     // Lr/Rr = (lr+M)/Rr Nou motor
float taorotorinv=15.76570;   // Rr/Lr = Rr/(lr+M) Nou motor
float Tr=0.0634288;           // Lr/Rr = (lr+M)/Rr Nou motor
```

```
// FAM, EstRT
float ModV=1.0, DesfaseVI=0.0, Desfaseant=0.0, CsgnM=0.5, DifFase=0.0;
float ws=4.0,s=0.0,wm=0.0;
float Rs=4.3, Rr=5.05, Lm=0.3056, Lr=0.3203, Ls=0.3203, ImaM=2, p=2.0;
float Lsp=0.02872535;         // Lsp=sigma*Ls; sigma=1-(Lm^2/Ls*Lr);

// FAM Magnetització
float Vmagnet=30, Fasemagnet=1.371;

/* Variables de l'adquisició de dades DSP--->PC*/
float valorch0,valorch1,valorch2,valorch3,valorch4,valorch5;
float of0=-31.5,of1=-48.5,of2=-62.6,of3=-40.5; // Calibració adq. dades
unsigned long analog0,analog1,analog2,analog3,analog4,analog5;
short int tensio0,tensio1,tensio2,tensio3,tensio4,tensio5;
float conv=6.1035156e-5;            // Constant que converteix el valor
                            // entregat pel DSP en valor real


/* Variables del pas de paràmetres PC--->DSP */
unsigned long tmin=5,tz=100,tzminl,tzant; // Variables dels temps
unsigned long inici;                      //Variables d'inici
float  fc,m,tzmin;
unsigned long Vdc=537,Vref;         // Variables de les tensions
float fase;                   // Variable de l'increment de fase
                              // que ha d'imposar el modulador
float vel;                    //Pulsació angular passada al modulador

unsigned long stop=0;                 // Variable de posta en marxa

/* Variables de l'adquisició de dades DSP--->PC*/
unsigned long sex=0;          //Sextant actual imposat pel Modulador
float faseAgafa;        //Fase actual del modulador dins el 1r sextant
unsigned long EstaZero;       // Variable de Pas per Zero
float alfa;             //Angle que ha imposat el DSP a la sortida

/* Variables intermitges */
float freq;       // Pulsació angular
float freq2;            // Freqüència
float Usvm;
float FitaSvm=0;

float Vfaserms;
float VfasePic;

/* Variables Sensor-less */
float WestRP=0,WestRPant=0; // Vel (Rad/s) estimada per l'estimador RP
float WestRT=0,WestRTant=0; // Vel (Rad/s) estimada per l'estimador RT
float k=2.26;               // Constant de l'Estimador en RP
float UaRef=0, UbRef=0,UcRef=0;     // Tensió imposada a la fase A i B
float UaRefAnt=0, UbRefAnt=0,UcRefAnt=0;// Tensió imposada a la fase A,B
float FluxDispDsAnt=0, FluxDispQsAnt=0;// Variables de l'Estimador en RT
float FluxDrAnt=0, FluxQrAnt=0;     // Variables de l'Estimador en RT
float Kc=0.008;                     // Filtre 1 Estimador RP
float Kc2=0.001;              // Filtre 2 Estimador RT (Variable)
float Kc3=0.001;              // Filtre 2 Derivada de Iq
float Kc4=0.01;               // Filtre 3 Sortida final
float wx=0,wxAnt=0;           //Velocitat Estimada final
```

```
/* Variables sistema Derivació Iq */
float DerIq=0,DerIqAnt=0,DerIqFinal=0;
float iqAnt=0;

/* Variables sistema FUZZY Iq */
float P,M,G;
float RP=0.1,RPRT=0.4,RT=1;
float FB=0.0001,FM=0.0018,FS=0.0028;
float BaseFuzzy=20;
float Promig=0,Filtre;

// Model de l'inversor
int TensioPic=0;
float FactorTensio;

//Matriu de dades
float dades[5][2500];
int pujadades=0;
int pujadades2=0;

// Salts en el càlcul del FAM
int pujaFAM=1;

/* Variables d'us general */
char x='n';
int printa;
int FirstScan=1;

/* Consigna general */
float wcon=0;

/* Variables del mòdul PI Velocitat */
float iqcon;
float want;
float PI1max=20,PI1min=-20,BP1=2,Ti1=0.11;     // PIVelocitat

/* Variables mesurades (entrada al programa)*/
float ia,ib;
float w;
float w2;

/* Variables de sortida del mòdul Transformada Inversa de Park */
float id=0,iq=0;

/* Variables del mòdul Model del Motor */
float fi=0,imr=0;

/* Variables per la mesura de temps */
double estat1=0,estat2=0;
double temps,a,b;
unsigned int paraula,ticks;
unsigned int baix,alt;
double tempscicle,tempsciclereal=0;

double tempsdesig=0.00010;    // Temps de cicle
float Ts=0.00010;       // Temps d'integració

double tempsdif=0;
double acumulat=0;
float velocitat;
```

```
/* Definició de rutines */
void IniciDSP(void);
void PasParamIniciDSP(void);
void PasParamDSP(void);
void EngegaDSP(void);
void AturaDSP(void);
void LlegirDSP(void);
void FaseNova(void);
void PiVelocitat(void);
void CalculWestRP(void);
void CalculWestRT(void);
void FAM(void);
void ModelInversor(void);
void parkinv(void);
void motor(void);
void Derivada(void);
void Fuzzy(void);

void EscriuFitxer(void);

void main (void)
{
unsigned long puja=0;
unsigned long fipuja=1;
unsigned long fipujaMagnet=0;
float tmagnet=0.1;
int segons=0;
int voltes=0,voltes2=0;
tempscicle=tempsdesig;
clrscr();

printf("Entra temps de magnetització (s) 0.1 : ");
scanf("%f", &tmagnet);
fipujaMagnet=(unsigned long)(tmagnet/tempsdesig);
printf("\nEntra Tensió de magnetització (V) 30 : ");
scanf("%f", &Vmagnet);
printf("\nEntra fase de magnetització (Rad) 1.371 : ");
scanf("%f", &Fasemagnet);

printf("\nEntra nombre de segons (s) 3 : ");
scanf("%li", &fipuja);
fipuja=fipuja/tempsdesig;
printf("\nEntra la velocitat en RAD/S (300): ");
scanf("%f", &wcon);

printf("\nPrem un '1' per començar");
x = getch();

/*****************************************************************/
/*              Inicialització del DSP                           */
/*****************************************************************/
IniciDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Inicialització del DSP                        */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*              Pas dels primers paràmetres                      */
/*****************************************************************/
PasParamIniciDSP();
freq=0*2*PI;      //2PI Rad/seg
Usvm=0;
FitaSvm=0;
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Pas dels primers paràmetres         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*              Engega el DSP                       */
/*****************************************************************/
inici=0;
EngegaDSP();

do
{
Get_DPRAM_Word_32(START,&inici);
}
while (inici!=1);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*         Fi Engega el DSP                                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


if (x=='1')
{
/*Primera mesura de temps */
disable();                    //Deshabilita int
ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
outport((0x43),(0x00));             //Escriptura perifèric(c,d)
enable();                     //Habilita int
baix=inportb(0x40);                 //Lectura Perifèric
alt=inportb(0x40);                  //Lectura Perifèric
paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*Fi primera mesura de temps*/

/*****************************************************************/
/*          Magnetització de la màquina                */
/*                                                     */
/*                                                     */
/*                                                     */
/*****************************************************************/

TensioPic=Vmagnet;                //Entrada al model de l'inversor
ModelInversor();
ModV=Vmagnet;                       //Entrada al Estim Reg Trans
Usvm=(Vmagnet*100)/81.4;
Usvm=abs(Usvm);
FitaSvm=Fasemagnet;                 //Entrada fase magnetitzant
freq=0;                           //Freqüència magnetitzant
gotoxy(25,6);
printf("Magneto");
```

```
 do
 {
/*****************************************************************/
/*     Temps d'espera                                            */
/*****************************************************************/
/* Part del programa que genera un temps d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                    // Línia a eliminar
            disable();                   //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
            outport((0x43),(0x00));      //Escriptura perifèric(c,d)
            enable();                    //Habilita int
            baix=inportb(0x40);          //Lectura Perifèric
            alt=inportb(0x40);           //Lectura Perifèric
            paraula=(baix)|(alt)<<8;     //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;         //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                 Fi Temps d'espera                             */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*           Mesura Temps inicial de la rutina                   */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                    //Deshabilita int
      ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
      outport((0x43),(0x00));       //Escriptura perifèric(c,d)
      enable();                    //Habilita int
      baix=inportb(0x40);           //Lectura Perifèric
      alt=inportb(0x40);            //Lectura Perifèric
      paraula=(baix)|(alt)<<8;      //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*          Actualització de sortida                             */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Actualització de sortida                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*          Rutina de Control Magnetització                      */
/*****************************************************************/
// Captura dels paràmetres
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor
ia=7.0922*(valorch1);                 //Adequació corrents
ib=7.0922*(valorch2);
CalculWestRT();                       //Estimador RT

FitaSvm=0;              // Fase a zero per no anar incrementant
```

```
// Càlcul de la velocitat en rpm
w2=(valorch3/(6e-4))/9.55;           //Adequació velocitat


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;


// Visualització de dades (No necessari a la pràctica)
if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensió= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
// Fi visualització de les dades


/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control Magnetització                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*  FI Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
voltes2=voltes;
voltes=0;
puja++;
      }
while (puja<fipujaMagnet);

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          FINAL Magnetització de la màquina                    */
/*                                                  */
/*                                                  */
/*                                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                LLAÇ de CONTROL                                */
/*                                                  */
/*                                                  */
/*                                                  */
/*****************************************************************/


do
{
/*****************************************************************/
/*                Temps d'espera                                 */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                  // Línia a eliminar
            disable();                 //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));          //Escriptura perifèric(c,d)
            enable();                  //Habilita int
            baix=inportb(0x40);              //Lectura Perifèric
            alt=inportb(0x40);               //Lectura Perifèric
            paraula=(baix)|(alt)<<8;         //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;             //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*          Mesura Temps inicial de la rutina                    */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                                              //Deshabilita
int
      ticks=*(unsigned long far *)(0x46c);      //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                       //Habilita int
      baix=inportb(0x40);                 //LECTura Perifèric
      alt=inportb(0x40);                  //LECTura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                          */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                                   */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                          //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                       //Càlcul de iq
Derivada();
CalculWestRP();                      //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                           //Estimador RT
```

```
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                          //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                    ModV
//             Consigna de Freqüència                ws
//                 Correcció de fase per si cal      DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;                //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
```

```
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari                */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipuja);
}

/*****************************************************************/
/*                Atura el DSP
*/
/*****************************************************************/
AturaDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura el DSP                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                Escriu Dades al fitxer                         */
/*****************************************************************/
EscriuFitxer();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Escriu Dades al fitxer                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


printf("\nVoltes= %i",voltes2);
printf("  Acum= %f segons",acumulat);
printf("\nPrem una tecla per sortir");
x=getch();
}


/*****************************************************************/
/*                Inicialització del DSP                          */
/*****************************************************************/
void IniciDSP(void)
{
/* Selecciona el tamany de dades amb el que treballarà el DSP  */
Set_Processor_Data_Type_Size_32();

/* Inicialitza el DSP amb la direcció base de la targeta (0x290), la */
/* direcció base de la DPRAM (0xD00), i el tamany del bus ISA  */
Select_Board(0x290, 0xd00, 8);

/* Habilita la DPRAM  */
Enable_DPRAM();

/* Situa el DSP en estat conegut  */
Global_Reset();

/* Carrega el fitxer que es desitja executar  */
Load_Object_File(FILENAME);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Inicialització del DSP                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Primer Pas de Paràmetres del DSP                */
/*****************************************************************/
/*
Entrada:    Vdc         tensió del bus en Volts
            tmin  temps mínim del pols en Microsegons
            tz      temps del període en Microsegons
Sortida:    VDC         Pas sobre el DSP Vdc
            TZUS  Pas sobre DSP de tz
            TMIN1 Pas sobre DSP de tmin
*/
void PasParamIniciDSP(void)
{
/* Passa al DSP el valor del bus de contínua Per defecte Vdc=537 */
Put_DPRAM_Word_32(VDC, Vdc);

/* Col·loquem al DSP un valor de tz=100 i tmin=5; */
```

```
Put_DPRAM_Word_32(TZUS, tz);
Put_DPRAM_Word_32(TMIN1, tmin);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Primer Pas de Paràmetres del DSP                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Pas de Paràmetres al DSP                          */
/*****************************************************************/
void PasParamDSP(void)
{
/* Adequació de les variables per al pas de paràmetres/
Vref=(unsigned long)(Usvm);
fase=FitaSvm;
vel=freq;              //Pulsació angular

/* Posa la consigna Vref a la posició de memòria VREF de la DPRAM  */
Put_DPRAM_Word_32(VREF, Vref);
/* Actualitza la fase del vector Vref a la pos de DPRAM FaseVREF */
Put_DPRAM_Float_32(SFVS, fase); //Control V_F no cal fase
/* Posa la consigna vel a la posició de memòria WM de la DPRAM  */
Put_DPRAM_Word_32(WM, vel);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Pas de Paràmetres al DSP                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Llegir DSP                              */
/*****************************************************************/
/* Obtenció dels senyals de la DPRAM */
void LlegirDSP(void)
{

Get_DPRAM_Word_32(Wm1,&analog1);
Get_DPRAM_Word_32(Wm2,&analog2);
Get_DPRAM_Word_32(Wm3,&analog3);

tensio1 =(short int)analog1;
tensio2 =(short int)analog2;
tensio3 =(short int)analog3;

valorch1=tensio1;
valorch2=tensio2;
valorch3=tensio3;

if (FirstScan==1) of3=-valorch3;

valorch1 =-1.0*(valorch1+of1)*conv;
valorch2 =-1.0*(valorch2+of2)*conv;
valorch3 =-1.0*(valorch3+of3)*conv;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Llegir DSP                               */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                Engega DSP                                     */
/*****************************************************************/
void EngegaDSP(void)
{ stop=0;
      Put_DPRAM_Word_32(STOP, stop);
      /* Executa el fitxer carregat  */
      Run_From(Get_Entry_PC());
      delay(2000);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Engega DSP                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Atura DSP                                      */
/*****************************************************************/
void AturaDSP(void)
{ stop=1;
      Put_DPRAM_Word_32(STOP, stop);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura DSP                                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                PI Velocitat  PI1                              */
/*****************************************************************/
/*
Entrades    want:       valor real anterior de w       (PiVelocitat(void))
            wcon:       valor desitjat de w      (Usuari)
            w:          valor real de w  (Estimada)
            tempscicle: temps de durada del cicle    (general)
Sortides    iqcon:      corrent prop al parell desitjat      (FAM(void))
            want: valor real actual de iq  (PiVelociatat(void))
*/
void PiVelocitat(void)
{
float incI,incP;
float error;
error=wcon-w;
incI=(temps*error)/(BP1*Ti1);
incP=(want-w)/BP1;

iqcon=iqcon+incI+incP;

/* Anti Wind-Up */
if (iqcon>PI1max)
      iqcon=PI1max;
if (iqcon<PI1min)
      iqcon=PI1min;
want=w;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi PI Velocitat                                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*          Rutina Transformada Inversa de Park                  */
/*****************************************************************/
/*
Entrades    fi: angle de posició del flux (angle  integrat)
            ia: corrent estatòric   (Del DSP)
            ib: corrent estatòric   (Del DSP)
            centvingraus            (constant)
            factor:                 (constant)
Sortides          iq: valor real de iq          (Estimador RP)
*/
void parkinv(void)
{
float cos1,cos2,cos3,sin1,sin2,sin3;
float centvintgraus=2.09436512;     // cent vint graus en radiants
float factor=0.8164965;          // sqrt(2/3)

cos1=cos(fi);
cos2=cos(fi-centvintgraus);
cos3=cos(fi+centvintgraus);
sin1=sin(fi);
sin2=sin(fi-centvintgraus);
sin3=sin(fi+centvintgraus);

iq=factor*((sin3-sin1)*ia+(sin3-sin2)*ib);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Rutina Transformada Inversa de Park               */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Agafa la Fase actual imposada pel DSP            */
/*****************************************************************/
void FaseNova(void)
{
/* Agafa la consigna fase que hi ha a FaseVREF de la DPRAM  */
Get_DPRAM_Word_32(sextant,&sex);
Get_DPRAM_Float_32(FaseVREF,&faseAgafa);
alfa=(faseAgafa)+(sex*1.0471975);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Agafa la Fase actual imposada pel DSP           */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Estimador en Regim Permanent                   */
/*****************************************************************/
void CalculWestRP(void)
{
WestRP=(freq/2)-(k*iq);
// Filtre
WestRP=(Kc*(WestRP-WestRPant))+WestRPant;
WestRPant=WestRP;
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Estimador en Regim Permanent                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                Derivada de Iq                                 */
/*****************************************************************/
void Derivada(void)
{
DerIq=(iq-iqAnt)/Ts;
DerIq=(Kc3*(DerIq-DerIqAnt))+DerIqAnt;
iqAnt=iq;
DerIqAnt=DerIq;
DerIqFinal=abs(DerIq);
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Derivada de Iq
*/
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Estimador en Regim Transitori                   */
/*****************************************************************/
void CalculWestRT(void)
{

float UsD,UsQ;
float IsD,IsQ;
float DerFluxDs,DerFluxQs;
float FluxDr, FluxQr;
float DerFluxDr,DerFluxQr;
float FluxDispDs, FluxDispQs, FluxDispDx;
float DerFluxDispDs, DerFluxDispQs;

/* Generador de UaRef, UbRef */

UaRef=UaRefAnt;
UbRef=UbRefAnt;
UaRefAnt=ModV*FactorTensio*sin(alfa+(PI/6));
UbRefAnt=ModV*FactorTensio*sin(alfa+(+(PI/6)-(2*PI/3)));

      /* Expressió: (2*Usa+Usb)/sqrt(3) */
/* Clarke Tensions i corrents */
UsD=UaRef;
UsQ = ( 2.0 * UaRef + UbRef ) / sqrt( 3.0);

      /* Expressió: (2*Isa+Isb)/sqrt(3) */
IsD=ia;
IsQ = ( 2.0 * ia + ib ) / sqrt( 3.0);

/* Derivades de Flux */
/* Expressió: UsD-Rs*IsD */
DerFluxDs = UsD - Rs * IsD;

/* Expressió: UsQ-Rs*IsQ */
DerFluxQs = UsQ - 4.3 * IsQ;

      /* Expressió: Lsp*IsD */
FluxDispDs = Lsp * IsD;
DerFluxDispDs=(FluxDispDs-FluxDispDsAnt)/Ts;
FluxDispDsAnt =FluxDispDs;
```

```
      /* Expressió: Lsp*IsQ */
FluxDispQs = Lsp * IsQ;

DerFluxDispQs=(FluxDispQs-FluxDispQsAnt)/Ts;
FluxDispQsAnt =FluxDispQs;

      /* Expressió: (Lm/Tr)*IsD */
FluxDispDx = (Lm / Tr) * IsD;

      /* Expressió: (DerFluxDr) */
DerFluxDr= (Lr/Lm)* (DerFluxDs-DerFluxDispDs);

      /* Expressió: (FluxDr) */
FluxDr=FluxDrAnt+(DerFluxDr*Ts);
FluxDrAnt=FluxDr;

      /* Expressió: (DerFluxQr) */
DerFluxQr= (Lr/Lm)* (DerFluxQs-DerFluxDispQs);

      /* Expressió: (FluxQr) */
FluxQr=FluxQrAnt+(DerFluxQr*Ts);
FluxQrAnt=FluxQr;

WestRT=(FluxDispDx-DerFluxDr-(FluxDr/Tr))/(FluxQr*p);
WestRT=(Filtre*(WestRT-WestRTant))+WestRTant;
WestRTant=WestRT;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*            Fi Estimador en Regim Transitori                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/****************************************************************/
/*           Rutina Escriu Dades al Fitxer                    */
/****************************************************************/
void EscriuFitxer(void)
{
int j=0;
FILE *f_grf;

if ((f_grf=fopen("C:\\JORDI\\V_F3_1.GRF","w"))==0)
            {
            printf("No es pot obrir el fitxer");
            return;
            }
      else
            {
            for(j=0;j<pujadades;j++)
                  {
                  fprintf(f_grf, "%6.3f %6.3f %6.3f %6.3f %6.3f
\n",dades[0][j],dades[1][j],dades[2][j],dades[3][j],dades[4][j]);
                  }
            fclose(f_grf);
            }
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Rutina Escriu Dades al Fitxer                        */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                              FAM                              */
/*****************************************************************/
/*
Entrades    CsgnM:  Consigna de Parell        (PiVelociatat(void))
            wn:valor de la velocitat del rótor (Sensor-less)
            Rs, Rr, Lm, Lr, Ls, ImaM, p;       (Paràmetres motor)
Sortides    ws:         Pulsació síncrona de sortida   (Al DSP)
          ModV:   Valor de la tensió de fase
          DifFase:    Actualització de la fase de la tensió
*/

void FAM(void)
{
double ca, cb, cc, sws1, sws2, inter;

/**** FAM/MAC *****/
/* càlcul de la consigna de sws a partir del parell. */
if ( CsgnM != 0.0)
      {
      ca = 2.0 * CsgnM * pow( Lr * Ls - Lm * Lm, 2);
      cb = -3.0 * p * 2.0 * Rr * pow( Ls * Lm * ImaM, 2);
      cc = 2.0 * CsgnM * pow( (Ls * Rr), 2);
      inter=(cb*cb) - (4*ca*cc);
      if (inter>0)
            {
            sws1 = ( -1.0 * cb - sqrt(inter) ) / (2*ca);
            sws2 = ( -1.0 * cb + sqrt(inter) ) / (2*ca);
            if (fabs(sws1)>fabs(sws2)) sws1=sws2; /* Es pren sws menor */
            }                                     /* en valor absolut  */

      else sws1=10.0;

      /* Càlcul de la pulsació. */
      ws = wm*p + sws1;
      s=sws1/ws;
      }

/* Càlcul vector tensió en valor eficaç. */
ModV=ImaM*sqrt(pow(sws1*Ls*Lr*Rs+ws*Ls*Ls*Rr,2)+pow(ws*Ls*sws1*(Ls*Lr-
Lm*Lm)-Ls*Rs*Rr,2));
ModV=ModV/sqrt(pow(sws1*(Ls*Lr-Lm*Lm),2)+pow(Ls*Rr,2));

/* Càlcul del desfase actual entre tensió i corrent magnetizant. */
DesfaseVI=atan2(sws1*ws*(Ls*Lr-Lm*Lm)-(Rs*Rr),ws*(s*Lr*Rs+Ls*Rr));
DesfaseVI=DesfaseVI - atan2(-Rr,sws1*(Lr-Lm*Lm/Ls));

/* Càlcul del salto de desfase tensi¢-corrent magnetizant entre */
/* l'instant actual i l'anterior. */

DifFase=DesfaseVI-Desfaseant;

Desfaseant=DesfaseVI;

if (ModV>311.0) ModV=311.0;
if (ws>650.0) ws=650.0;
if (ws<-650.0) ws=-650.0;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                       Fi FAM                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                      Rutina Model de l'inversor II            */
/*****************************************************************/

void ModelInversor(void)
{
      if (TensioPic<=50)
            {
            FactorTensio=0.003988*TensioPic+0.6136;
            }
      if ((TensioPic>50)&&(TensioPic<=100))
            {
            FactorTensio=0.00206*(TensioPic-50)+0.813;
            }
      if ((TensioPic>100)&&(TensioPic<=150))
            {
            FactorTensio=0.00056*(TensioPic-100)+0.916;
            }
      if ((TensioPic>150)&&(TensioPic<=200))
            {
            FactorTensio=0.00028*(TensioPic-150)+0.944;
            }
      if ((TensioPic>200)&&(TensioPic<=250))
            {
            FactorTensio=0.00026*(TensioPic-200)+0.958;
            }
      if ((TensioPic>250)&&(TensioPic<=300))
            {
            FactorTensio=0.0001*(TensioPic-250)+0.971;
            }
      if ((TensioPic>300)&&(TensioPic<=350))
            {
            FactorTensio=0.00008*(TensioPic-300)+0.976;
            }

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                  Fi Rutina Model de l'inversor                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                      Rutina Fuzzy                             */
/*****************************************************************/
void Fuzzy(void)
{
float DerIqDividit;
if (DerIqFinal<=BaseFuzzy)
      {
      DerIqDividit=DerIqFinal/BaseFuzzy;
      P=1-(DerIqDividit);
      M=DerIqDividit;
      G=0;
      }
else
      {
      if ((DerIqFinal>BaseFuzzy)&&(DerIqFinal<=2*BaseFuzzy))
            {
            DerIqDividit=(DerIqFinal-BaseFuzzy)/BaseFuzzy;
            P=0;
```

```
            M=1-DerIqDividit;
            G=DerIqDividit;
            }
      else
            {
            P=0;
            M=0;
            G=1;
            }
      }
Promig=P*RP+M*RPRT+G*RT;
Filtre=P*FB+M*FM+G*FS;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                      Fi Rutina Fuzzy                 */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

II.2 Programa "Rampes.c"

```
/*********************************************************************/
/* RAMPES.C  Programa que implementa un llaç de control FAM sense  */
/*          sensor sobre el motor                                    */
/* Estima el valor de la velocitat amb l'estimador RP                */
/* Estima el valor de la velocitat amb l'estimador RT                */
/* Implementa el sistema Fuzzy per filtrar i per promitjar           */
/* Es tanca el llaç de control amb la velocitat estimada             */
/* El sistema genera respostes a consignes tipus rampes i trams plans */
/*********************************************************************/
/* Aquest programa genera temps de 100us i a més corregeix en el
cas de que una rutina trigui més de 100us, la següent triga menys */

/* Inclusió de llibreries necessàries per la rutina de mesura de temps
i la de control */
#include <dos.h>
#include <time.h>
#include <math.h>
#include <conio.h>

/* Inclusió de llibreries necessàries per al control del DSP */
#include "c:\mon32\bc45\tic32.h"
#include<stdio.h>
#include<stdlib.h>

/* Es defineixen les posicions de memòria de la DPRAM a  on hi ha les  */
/* consignes del modulador vectorial */

#define  VREF       0xc0000dl   // Mòdul de Vref (Modulador vectorial)
#define  VDC        0xc0000el   // Valor del bus de contínua
#define  WM         0xc0000fl   // Freqüència del senyal (en rad/s)
#define  ZERO       0xc000101   // Bit indicant pas per zero del motor
#define  STOP       0xc000121   // Bit de stop del modulador
#define  TZUS       0xc000141   // Tz
#define  TMIN1      0xc000151    // Tmin
#define  CONTATGE   0xc000161
#define  FaseVREF   0xc000171   // Fase del vector VREF
#define  sextant    0xc000191   // Sextant de treball
#define  SFVS       0xc0001al    // correcció de la fase del FAM
#define  START      0xc0001cl   // Indica que el sistema ha engegat
#define  Posicio    0xc000261
#define  Velocitat  0xc000271
#define  Wm0        0xc000441   // Posició DPRAM canal0
#define  Wm1        0xc000451   // Posició DPRAM canal1
#define  Wm2        0xc000461   // Posició DPRAM canal2
#define  Wm3        0xc000471   // Posició DPRAM canal3
#define  PI 3.141592654

/* Es defineix l'arxiu que serà llençat per a que corri en el DSP  */
#define  FILENAME "MODUL.OUT"


/* Constants */
float RpmRads=0.2094395;      // Conversió de rpm a Rad/s
float taorotor=0.0634288;     // Lr/Rr = (lr+M)/Rr Nou motor
float taorotorinv=15.76570;   // Rr/Lr = Rr/(lr+M) Nou motor
float Tr=0.0634288;           // Lr/Rr = (lr+M)/Rr Nou motor
```

```
// FAM, EstRT
float ModV=1.0, DesfaseVI=0.0, Desfaseant=0.0, CsgnM=0.5, DifFase=0.0;
float ws=4.0,s=0.0,wm=0.0;
float Rs=4.3, Rr=5.05, Lm=0.3056, Lr=0.3203, Ls=0.3203, ImaM=2, p=2.0;
float Lsp=0.02872535;         // Lsp=sigma*Ls; sigma=1-(Lm^2/Ls*Lr);

// FAM Magnetització
float Vmagnet=30, Fasemagnet=1.371;

/* Variables de l'adquisició de dades DSP--->PC*/
float valorch0,valorch1,valorch2,valorch3,valorch4,valorch5;
float of0=-31.5,of1=-48.5,of2=-62.6,of3=-40.5; // Calibració adq. dades
unsigned long analog0,analog1,analog2,analog3,analog4,analog5;
short int tensio0,tensio1,tensio2,tensio3,tensio4,tensio5;
float conv=6.1035156e-5;            // Constant que converteix el valor
                            // entregat pel DSP en valor real


/* Variables del pas de paràmetres PC--->DSP */
unsigned long tmin=5,tz=100,tzminl,tzant; // Variables dels temps
unsigned long inici;                      //Variables d'inici
float  fc,m,tzmin;
unsigned long Vdc=537,Vref;         // Variables de les tensions
float fase;                   // Variable de l'increment de fase
                              // que ha d'imposar el modulador
float vel;                    //Pulsació angular passada al modulador

unsigned long stop=0;                 // Variable de posta en marxa

/* Variables de l'adquisició de dades DSP--->PC*/
unsigned long sex=0;          //Sextant actual imposat pel Modulador
float faseAgafa;        //Fase actual del modulador dins el 1r sextant
unsigned long EstaZero;       // Variable de Pas per Zero
float alfa;             //Angle que ha imposat el DSP a la sortida

/* Variables intermitges */
float freq;       // Pulsació angular
float freq2;            // Freqüència
float Usvm;
float FitaSvm=0;

float Vfaserms;
float VfasePic;

/* Variables Sensor-less */
float WestRP=0,WestRPant=0; // Vel (Rad/s) estimada per l'estimador RP
float WestRT=0,WestRTant=0; // Vel (Rad/s) estimada per l'estimador RT
float k=2.26;               // Constant de l'Estimador en RP
float UaRef=0, UbRef=0,UcRef=0;     // Tensió imposada a la fase A i B
float UaRefAnt=0, UbRefAnt=0,UcRefAnt=0;// Tensió imposada a la fase A,B
float FluxDispDsAnt=0, FluxDispQsAnt=0;// Variables de l'Estimador en RT
float FluxDrAnt=0, FluxQrAnt=0;     // Variables de l'Estimador en RT
float Kc=0.008;                     // Filtre 1 Estimador RP
float Kc2=0.001;              // Filtre 2 Estimador RT (Variable)
float Kc3=0.001;              // Filtre 2 Derivada de Iq
float Kc4=0.01;               // Filtre 3 Sortida final
float wx=0,wxAnt=0;           //Velocitat Estimada final
```

```
/* Variables sistema Derivació Iq */
float DerIq=0,DerIqAnt=0,DerIqFinal=0;
float iqAnt=0;

/* Variables sistema FUZZY Iq */
float P,M,G;
float RP=0.1,RPRT=0.4,RT=1;
float FB=0.0001,FM=0.0018,FS=0.0028;
float BaseFuzzy=20;
float Promig=0,Filtre;

// Model de l'inversor
int TensioPic=0;
float FactorTensio;

//Matriu de dades
float dades[5][2500];
int pujadades=0;
int pujadades2=0;

// Salts en el càlcul del FAM
int pujaFAM=1;

/* Variables d'us general */
char x='n';
int printa;
int FirstScan=1;

/* Consigna general */
float wcon=0;

/* Variables del mòdul PI Velocitat */
float iqcon;
float want;
float PI1max=20,PI1min=-20,BP1=2,Ti1=0.11;     // PIVelocitat

/* Variables mesurades (entrada al programa)*/
float ia,ib;
float w;
float w2;

/* Variables de sortida del mòdul Transformada Inversa de Park */
float id=0,iq=0;

/* Variables del mòdul Model del Motor */
float fi=0,imr=0;

/* Variables per la mesura de temps */
double estat1=0,estat2=0;
double temps,a,b;
unsigned int paraula,ticks;
unsigned int baix,alt;
double tempscicle,tempsciclereal=0;

double tempsdesig=0.00010;    // Temps de cicle
float Ts=0.00010;       // Temps d'integració

double tempsdif=0;
double acumulat=0;
float velocitat;
```

```
/* Definició de rutines */
void IniciDSP(void);
void PasParamIniciDSP(void);
void PasParamDSP(void);
void EngegaDSP(void);
void AturaDSP(void);
void LlegirDSP(void);
void FaseNova(void);
void PiVelocitat(void);
void CalculWestRP(void);
void CalculWestRT(void);
void FAM(void);
void ModelInversor(void);
void parkinv(void);
void motor(void);
void Derivada(void);
void Fuzzy(void);

void EscriuFitxer(void);

void main (void)
{
unsigned long puja=0;
unsigned long fipuja=3;
unsigned long fipla=0;
unsigned long fibaixa=0;
unsigned long fipla2=0;
unsigned long fipujaMagnet=0;
float tassaig;
float tmagnet=0.1;
int segons=0;
int voltes=0,voltes2=0;
tempscicle=tempsdesig;
clrscr();

printf("Entra temps de magnetització (s) 0.1 : ");
scanf("%f", &tmagnet);
fipujaMagnet=(unsigned long)(tmagnet/tempsdesig);
printf("\nEntra Tensió de magnetització (V) 30 : ");
scanf("%f", &Vmagnet);
printf("\nEntra fase de magnetització (Rad) 1.371 : ");
scanf("%f", &Fasemagnet);

printf("Entra temps de pujada(s) (Mínim 0.2 s): ");
scanf("%f", &tassaig);
fipuja=(unsigned long)(tassaig/tempsdesig);

printf("Entra temps d'espera(s): (Màxim 3 s)  ");
scanf("%f", &tassaig);
if (tassaig>3) tassaig=3;
fipla=(unsigned long)(tassaig/tempsdesig);

printf("Entra temps de baixada(s): (Màxim 3 s) ");
scanf("%f", &tassaig);
if (tassaig>3) tassaig=3;
fibaixa=(unsigned long)(tassaig/tempsdesig);

printf("Entra temps d'espera 2 (s): (Màxim 3 s)  ");
scanf("%f", &tassaig);
if (tassaig>3) tassaig=3;
fipla2=(unsigned long)(tassaig/tempsdesig);
```

```
printf("\nEntra la velocitat en RAD/S (100): ");
scanf("%f", &wtemp);

printf("\nPrem un '1' per començar");
x = getch();

/*****************************************************************/
/*              Inicialització del DSP                           */
/*****************************************************************/
IniciDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Inicialització del DSP                        */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*              Pas dels primers paràmetres                      */
/*****************************************************************/
PasParamIniciDSP();
freq=0*2*PI;      //2PI Rad/seg
Usvm=0;
FitaSvm=0;
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Pas dels primers paràmetres                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*              Engega el DSP                                    */
/*****************************************************************/
inici=0;
EngegaDSP();

do
{
Get_DPRAM_Word_32(START,&inici);
}
while (inici!=1);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*         Fi Engega el DSP                                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

if (x=='1')
{
/*Primera mesura de temps */
disable();                    //Deshabilita int
ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
outport((0x43),(0x00));             //Escriptura perifèric(c,d)
enable();                     //Habilita int
baix=inportb(0x40);                 //Lectura Perifèric
alt=inportb(0x40);                  //Lectura Perifèric
paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*Fi primera mesura de temps*/
/*****************************************************************/
/*          Magnetització de la màquina                          */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
```

```
TensioPic=Vmagnet;                //Entrada al model de l'inversor
ModelInversor();
ModV=Vmagnet;                       //Entrada al Estim Reg Trans
Usvm=(Vmagnet*100)/81.4;
Usvm=abs(Usvm);
FitaSvm=Fasemagnet;                 //Entrada fase magnetitzant
freq=0;                           //Freqüència magnetitzant
gotoxy(25,6);
printf("Magneto");

 do
 {
/*****************************************************************/
/*     Temps d'espera                                            */
/*****************************************************************/
/* Part del programa que genera un temps d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                     // Línia a eliminar
            disable();                    //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
            outport((0x43),(0x00));       //Escriptura perifèric(c,d)
            enable();                     //Habilita int
            baix=inportb(0x40);           //Lectura Perifèric
            alt=inportb(0x40);            //Lectura Perifèric
            paraula=(baix)|(alt)<<8;      //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;          //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                 Fi Temps d'espera                             */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*           Mesura Temps inicial de la rutina                   */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                    //Deshabilita int
      ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);                 //Lectura Perifèric
      alt=inportb(0x40);                  //Lectura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*          Actualització de sortida                             */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Actualització de sortida                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*          Rutina de Control Magnetització                      */
/*****************************************************************/
// Captura dels paràmetres
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor
ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);
CalculWestRT();                     //Estimador RT

FitaSvm=0;              // Fase a zero per no anar incrementant

// Càlcul de la velocitat en rpm
w2=(valorch3/(6e-4))/9.55;           //Adequació velocitat

if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

// Visualització de dades (No necessari a la pràctica)
if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensió= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
```

```
      {
      printa++;
      }
// Fi visualització de les dades


/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control Magnetització                 */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*  FI Càlcul de l'Ajust de temps d'espera necessari             */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
voltes2=voltes;
voltes=0;
puja++;
      }
while (puja<fipujaMagnet);

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          FINAL Magnetització de la màquina                    */
/*                                                          */
/*                                                          */
/*                                                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                LLAÇ de CONTROL PUJADA                    */
/*                                                          */
/*                                                          */
/*                                                          */
/*****************************************************************/


do
{
/*****************************************************************/
/*                Temps d'espera                              */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                  // Línia a eliminar
            disable();                 //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));          //Escriptura perifèric(c,d)
            enable();                  //Habilita int
            baix=inportb(0x40);              //Lectura Perifèric
            alt=inportb(0x40);               //Lectura Perifèric
            paraula=(baix)|(alt)<<8;         //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
```

```
            temps=estat2-estat1;              //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*          Mesura Temps inicial de la rutina                    */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                                                  //Deshabilita
int
      ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);                 //LECTura Perifèric
      alt=inportb(0x40);                  //LECTura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                  */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                           */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=(wtemp*(puja-fipujaMagnet+1))/(fipuja-fipujaMagnet);
/*************************/
/*Fi càlcul consigna      */
/*************************/

// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);
```

```
// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                          //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                       //Càlcul de iq
Derivada();
CalculWestRP();                      //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                            //Estimador RT
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                         //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                  ModV
//             Consigna de Freqüència               ws
//                 Correcció de fase per si cal      DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;               //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
```

```
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*******************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari          */
/*    en funció de l'error de temps acumulat                 */
/*******************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipuja);
}

/*****************************************************************/
/*                LLAÇ de CONTROL PLA                */
/*                                                   */
/*                                                   */
/*                                                   */
/*****************************************************************/


do
{
/*****************************************************************/
/*                Temps d'espera                          */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                  // Línia a eliminar
            disable();                 //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));          //Escriptura perifèric(c,d)
            enable();                  //Habilita int
            baix=inportb(0x40);              //Lectura Perifèric
            alt=inportb(0x40);               //Lectura Perifèric
            paraula=(baix)|(alt)<<8;         //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;             //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/




/*****************************************************************/
/*          Mesura Temps inicial de la rutina                 */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                                          //Deshabilita
int
      ticks=*(unsigned long far *)(0x46c);      //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);                 //LECTura Perifèric
      alt=inportb(0x40);                  //LECTura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina             */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*             Actualització de sortida                  */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                                   */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=wtemp;
/*************************/
/*Fi càlcul consigna      */
/*************************/

// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                         //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                      //Càlcul de iq
Derivada();
CalculWestRP();                     //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                           //Estimador RT
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();
```

```
wm=wx;                          //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                    ModV
//             Consigna de Freqüència                ws
//                 Correcció de fase per si cal       DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;                //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
```

```
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                        */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipla);
}

/*****************************************************************/
/*                LLAÇ de CONTROL BAIXADA                        */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/

do
{
/*****************************************************************/
/*                Temps d'espera                                 */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                  // Línia a eliminar
```

```
            disable();                  //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));           //Escriptura perifèric(c,d)
            enable();                   //Habilita int
            baix=inportb(0x40);               //Lectura Perifèric
            alt=inportb(0x40);                //Lectura Perifèric
            paraula=(baix)|(alt)<<8;          //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;              //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/




/*****************************************************************/
/*          Mesura Temps inicial de la rutina                */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                              /Deshabilita int
      ticks=*(unsigned long far *)(0x46c);    //Lectura memòria(c)
      outport((0x43),(0x00));           //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);               //LECTura Perifèric
      alt=inportb(0x40);                //LECTura Perifèric
      paraula=(baix)|(alt)<<8;          //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina            */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                  */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*          Rutina de Control                            */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=wtemp-(2*(wtemp*(puja-fipla))/(fibaixa-fipla));
/*************************/
/*Fi càlcul consigna      */
/*************************/
```

```
// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                         //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                      //Càlcul de iq
Derivada();
CalculWestRP();                     //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                            //Estimador RT
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                        //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                   ModV
//             Consigna de Freqüència               ws
//                 Correcció de fase per si cal      DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;               //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
```

```
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                  //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                        */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari             */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fibaixa);
}


/*****************************************************************/
/*                LLAÇ de CONTROL PLA2                           */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/

do
{
/*****************************************************************/
/*                Temps d'espera                                 */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                   // Línia a eliminar
            disable();                  //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));           //Escriptura perifèric(c,d)
            enable();                   //Habilita int
            baix=inportb(0x40);               //Lectura Perifèric
            alt=inportb(0x40);                //Lectura Perifèric
            paraula=(baix)|(alt)<<8;          //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;              //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/




/*****************************************************************/
/*          Mesura Temps inicial de la rutina                    */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                              /Deshabilita int
      ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
      outport((0x43),(0x00));            //Escriptura perifèric(c,d)
      enable();                    //Habilita int
```

```
      baix=inportb(0x40);                //LECTura Perifèric
      alt=inportb(0x40);                 //LECTura Perifèric
      paraula=(baix)|(alt)<<8;           //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                  */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                               */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=-wtemp;
/*************************/
/*Fi càlcul consigna      */
/*************************/


// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                         //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                      //Càlcul de iq
Derivada();
CalculWestRP();                     //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                           //Estimador RT
// Fi Sensor-less RT
```

```
// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                          //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                    ModV
//             Consigna de Freqüència                ws
//                 Correcció de fase per si cal       DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;                //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
```

```
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari                */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipla2);
}



/*****************************************************************/
/*                Atura el DSP
*/
/*****************************************************************/
AturaDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura el DSP                                    */
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                Escriu Dades al fitxer                         */
/*****************************************************************/
EscriuFitxer();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Escriu Dades al fitxer                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


printf("\nVoltes= %i",voltes2);
printf("  Acum= %f segons",acumulat);
printf("\nPrem una tecla per sortir");
x=getch();
}


/*****************************************************************/
/*                Inicialització del DSP                          */
/*****************************************************************/
void IniciDSP(void)
{
/* Selecciona el tamany de dades amb el que treballarà el DSP  */
Set_Processor_Data_Type_Size_32();

/* Inicialitza el DSP amb la direcció base de la targeta (0x290), la */
/* direcció base de la DPRAM (0xD00), i el tamany del bus ISA  */
Select_Board(0x290, 0xd00, 8);

/* Habilita la DPRAM  */
Enable_DPRAM();

/* Situa el DSP en estat conegut  */
Global_Reset();

/* Carrega el fitxer que es desitja executar  */
Load_Object_File(FILENAME);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Inicialització del DSP                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Primer Pas de Paràmetres del DSP                */
/*****************************************************************/
/*
Entrada:    Vdc         tensió del bus en Volts
            tmin  temps mínim del pols en Microsegons
            tz      temps del període en Microsegons
Sortida:    VDC         Pas sobre el DSP Vdc
            TZUS  Pas sobre DSP de tz
            TMIN1 Pas sobre DSP de tmin
*/
void PasParamIniciDSP(void)
{
/* Passa al DSP el valor del bus de contínua Per defecte Vdc=537 */
Put_DPRAM_Word_32(VDC, Vdc);
```

```
/* Col·loquem al DSP un valor de tz=100 i tmin=5; */
Put_DPRAM_Word_32(TZUS, tz);
Put_DPRAM_Word_32(TMIN1, tmin);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Primer Pas de Paràmetres del DSP                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Pas de Paràmetres al DSP                       */
/*****************************************************************/
void PasParamDSP(void)
{
/* Adequació de les variables per al pas de paràmetres/
Vref=(unsigned long)(Usvm);
fase=FitaSvm;
vel=freq;              //Pulsació angular

/* Posa la consigna Vref a la posició de memòria VREF de la DPRAM  */
Put_DPRAM_Word_32(VREF, Vref);
/* Actualitza la fase del vector Vref a la pos de DPRAM FaseVREF */
Put_DPRAM_Float_32(SFVS, fase); //Control V_F no cal fase
/* Posa la consigna vel a la posició de memòria WM de la DPRAM  */
Put_DPRAM_Word_32(WM, vel);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Pas de Paràmetres al DSP                         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Llegir DSP                                     */
/*****************************************************************/
/* Obtenció dels senyals de la DPRAM */
void LlegirDSP(void)
{

Get_DPRAM_Word_32(Wm1,&analog1);
Get_DPRAM_Word_32(Wm2,&analog2);
Get_DPRAM_Word_32(Wm3,&analog3);

tensio1 =(short int)analog1;
tensio2 =(short int)analog2;
tensio3 =(short int)analog3;

valorch1=tensio1;
valorch2=tensio2;
valorch3=tensio3;

if (FirstScan==1) of3=-valorch3;

valorch1 =-1.0*(valorch1+of1)*conv;
valorch2 =-1.0*(valorch2+of2)*conv;
valorch3 =-1.0*(valorch3+of3)*conv;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Llegir DSP                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                Engega DSP                                     */
/*****************************************************************/
void EngegaDSP(void)
{ stop=0;
      Put_DPRAM_Word_32(STOP, stop);
      /* Executa el fitxer carregat  */
      Run_From(Get_Entry_PC());
      delay(2000);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Engega DSP                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Atura DSP                                      */
/*****************************************************************/
void AturaDSP(void)
{ stop=1;
      Put_DPRAM_Word_32(STOP, stop);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura DSP                                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                PI Velocitat  PI1                              */
/*****************************************************************/
/*
Entrades    want:       valor real anterior de w       (PiVelocitat(void))
            wcon:       valor desitjat de w      (Usuari)
            w:          valor real de w  (Estimada)
            tempscicle: temps de durada del cicle    (general)
Sortides    iqcon:      corrent prop al parell desitjat      (FAM(void))
            want: valor real actual de iq  (PiVelociatat(void))
*/
void PiVelocitat(void)
{
float incI,incP;
float error;
error=wcon-w;
incI=(temps*error)/(BP1*Ti1);
incP=(want-w)/BP1;

iqcon=iqcon+incI+incP;

/* Anti Wind-Up */
if (iqcon>PI1max)
      iqcon=PI1max;
if (iqcon<PI1min)
      iqcon=PI1min;
want=w;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi PI Velocitat                                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*          Rutina Transformada Inversa de Park                  */
/*****************************************************************/
/*
Entrades    fi: angle de posició del flux (angle  integrat)
            ia: corrent estatòric   (Del DSP)
            ib: corrent estatòric   (Del DSP)
            centvingraus            (constant)
            factor:                 (constant)
Sortides          iq: valor real de iq          (Estimador RP)
*/
void parkinv(void)
{
float cos1,cos2,cos3,sin1,sin2,sin3;
float centvintgraus=2.09436512;     // cent vint graus en radiants
float factor=0.8164965;          // sqrt(2/3)

cos1=cos(fi);
cos2=cos(fi-centvintgraus);
cos3=cos(fi+centvintgraus);
sin1=sin(fi);
sin2=sin(fi-centvintgraus);
sin3=sin(fi+centvintgraus);

iq=factor*((sin3-sin1)*ia+(sin3-sin2)*ib);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Rutina Transformada Inversa de Park               */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Agafa la Fase actual imposada pel DSP            */
/*****************************************************************/
void FaseNova(void)
{
/* Agafa la consigna fase que hi ha a FaseVREF de la DPRAM  */
Get_DPRAM_Word_32(sextant,&sex);
Get_DPRAM_Float_32(FaseVREF,&faseAgafa);
alfa=(faseAgafa)+(sex*1.0471975);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Agafa la Fase actual imposada pel DSP            */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Estimador en Regim Permanent                   */
/*****************************************************************/
void CalculWestRP(void)
{
WestRP=(freq/2)-(k*iq);
// Filtre
WestRP=(Kc*(WestRP-WestRPant))+WestRPant;
WestRPant=WestRP;
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Estimador en Regim Permanent                */
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                Derivada de Iq                                 */
/*****************************************************************/
void Derivada(void)
{
DerIq=(iq-iqAnt)/Ts;
DerIq=(Kc3*(DerIq-DerIqAnt))+DerIqAnt;
iqAnt=iq;
DerIqAnt=DerIq;
DerIqFinal=abs(DerIq);
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Derivada de Iq
*/
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Estimador en Regim Transitori                   */
/*****************************************************************/
void CalculWestRT(void)
{

float UsD,UsQ;
float IsD,IsQ;
float DerFluxDs,DerFluxQs;
float FluxDr, FluxQr;
float DerFluxDr,DerFluxQr;
float FluxDispDs, FluxDispQs, FluxDispDx;
float DerFluxDispDs, DerFluxDispQs;

/* Generador de UaRef, UbRef */

UaRef=UaRefAnt;
UbRef=UbRefAnt;
UaRefAnt=ModV*FactorTensio*sin(alfa+(PI/6));
UbRefAnt=ModV*FactorTensio*sin(alfa+(+(PI/6)-(2*PI/3)));

      /* Expressió: (2*Usa+Usb)/sqrt(3) */
/* Clarke Tensions i corrents */
UsD=UaRef;
UsQ = ( 2.0 * UaRef + UbRef ) / sqrt( 3.0);

      /* Expressió: (2*Isa+Isb)/sqrt(3) */
IsD=ia;
IsQ = ( 2.0 * ia + ib ) / sqrt( 3.0);

/* Derivades de Flux */
/* Expressió: UsD-Rs*IsD */
DerFluxDs = UsD - Rs * IsD;

/* Expressió: UsQ-Rs*IsQ */
DerFluxQs = UsQ - 4.3 * IsQ;

      /* Expressió: Lsp*IsD */
FluxDispDs = Lsp * IsD;
DerFluxDispDs=(FluxDispDs-FluxDispDsAnt)/Ts;
FluxDispDsAnt =FluxDispDs;
```

```
      /* Expressió: Lsp*IsQ */
FluxDispQs = Lsp * IsQ;

DerFluxDispQs=(FluxDispQs-FluxDispQsAnt)/Ts;
FluxDispQsAnt =FluxDispQs;

      /* Expressió: (Lm/Tr)*IsD */
FluxDispDx = (Lm / Tr) * IsD;

      /* Expressió: (DerFluxDr) */
DerFluxDr= (Lr/Lm)* (DerFluxDs-DerFluxDispDs);

      /* Expressió: (FluxDr) */
FluxDr=FluxDrAnt+(DerFluxDr*Ts);
FluxDrAnt=FluxDr;

      /* Expressió: (DerFluxQr) */
DerFluxQr= (Lr/Lm)* (DerFluxQs-DerFluxDispQs);

      /* Expressió: (FluxQr) */
FluxQr=FluxQrAnt+(DerFluxQr*Ts);
FluxQrAnt=FluxQr;

WestRT=(FluxDispDx-DerFluxDr-(FluxDr/Tr))/(FluxQr*p);
WestRT=(Filtre*(WestRT-WestRTant))+WestRTant;
WestRTant=WestRT;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Estimador en Regim Transitori                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*           Rutina Escriu Dades al Fitxer                       */
/*****************************************************************/
void EscriuFitxer(void)
{
int j=0;
FILE *f_grf;

if ((f_grf=fopen("C:\\JORDI\\V_F3_1.GRF","w"))==0)
            {
            printf("No es pot obrir el fitxer");
            return;
            }
      else
            {
            for(j=0;j<pujadades;j++)
                  {
                  fprintf(f_grf, "%6.3f %6.3f %6.3f %6.3f %6.3f
\n",dades[0][j],dades[1][j],dades[2][j],dades[3][j],dades[4][j]);
                  }
            fclose(f_grf);
            }
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Rutina Escriu Dades al Fitxer                             */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                              FAM                              */
/*****************************************************************/
/*
Entrades    CsgnM:  Consigna de Parell        (PiVelociatat(void))
            wn:valor de la velocitat del rótor (Sensor-less)
            Rs, Rr, Lm, Lr, Ls, ImaM, p;       (Paràmetres motor)
Sortides    ws:         Pulsació síncrona de sortida   (Al DSP)
          ModV:   Valor de la tensió de fase
          DifFase:    Actualització de la fase de la tensió
*/

void FAM(void)
{
double ca, cb, cc, sws1, sws2, inter;

/**** FAM/MAC *****/
/* càlcul de la consigna de sws a partir del parell. */
if ( CsgnM != 0.0)
      {
      ca = 2.0 * CsgnM * pow( Lr * Ls - Lm * Lm, 2);
      cb = -3.0 * p * 2.0 * Rr * pow( Ls * Lm * ImaM, 2);
      cc = 2.0 * CsgnM * pow( (Ls * Rr), 2);
      inter=(cb*cb) - (4*ca*cc);
      if (inter>0)
            {
            sws1 = ( -1.0 * cb - sqrt(inter) ) / (2*ca);
            sws2 = ( -1.0 * cb + sqrt(inter) ) / (2*ca);
            if (fabs(sws1)>fabs(sws2)) sws1=sws2; /* Es pren sws menor */
            }                                     /* en valor absolut  */

      else sws1=10.0;

      /* Càlcul de la pulsació. */
      ws = wm*p + sws1;
      s=sws1/ws;
      }

/* Càlcul vector tensió en valor eficaç. */
ModV=ImaM*sqrt(pow(sws1*Ls*Lr*Rs+ws*Ls*Ls*Rr,2)+pow(ws*Ls*sws1*(Ls*Lr-
Lm*Lm)-Ls*Rs*Rr,2));
ModV=ModV/sqrt(pow(sws1*(Ls*Lr-Lm*Lm),2)+pow(Ls*Rr,2));

/* Càlcul del desfase actual entre tensió i corrent magnetizant. */
DesfaseVI=atan2(sws1*ws*(Ls*Lr-Lm*Lm)-(Rs*Rr),ws*(s*Lr*Rs+Ls*Rr));
DesfaseVI=DesfaseVI - atan2(-Rr,sws1*(Lr-Lm*Lm/Ls));

/* Càlcul del salto de desfase tensi¢-corrent magnetizant entre */
/* l'instant actual i l'anterior. */

DifFase=DesfaseVI-Desfaseant;

Desfaseant=DesfaseVI;

if (ModV>311.0) ModV=311.0;
if (ws>650.0) ws=650.0;
if (ws<-650.0) ws=-650.0;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                       Fi FAM                                  */
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                      Rutina Model de l'inversor II              */
/*****************************************************************/

void ModelInversor(void)
{
      if (TensioPic<=50)
            {
            FactorTensio=0.003988*TensioPic+0.6136;
            }
      if ((TensioPic>50)&&(TensioPic<=100))
            {
            FactorTensio=0.00206*(TensioPic-50)+0.813;
            }
      if ((TensioPic>100)&&(TensioPic<=150))
            {
            FactorTensio=0.00056*(TensioPic-100)+0.916;
            }
      if ((TensioPic>150)&&(TensioPic<=200))
            {
            FactorTensio=0.00028*(TensioPic-150)+0.944;
            }
      if ((TensioPic>200)&&(TensioPic<=250))
            {
            FactorTensio=0.00026*(TensioPic-200)+0.958;
            }
      if ((TensioPic>250)&&(TensioPic<=300))
            {
            FactorTensio=0.0001*(TensioPic-250)+0.971;
            }
      if ((TensioPic>300)&&(TensioPic<=350))
            {
            FactorTensio=0.00008*(TensioPic-300)+0.976;
            }

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                  Fi Rutina Model de l'inversor                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                      Rutina Fuzzy                             */
/*****************************************************************/
void Fuzzy(void)
{
float DerIqDividit;
if (DerIqFinal<=BaseFuzzy)
      {
      DerIqDividit=DerIqFinal/BaseFuzzy;
      P=1-(DerIqDividit);
      M=DerIqDividit;
      G=0;
      }
else
      {
      if ((DerIqFinal>BaseFuzzy)&&(DerIqFinal<=2*BaseFuzzy))
            {
            DerIqDividit=(DerIqFinal-BaseFuzzy)/BaseFuzzy;
```

```
            P=0;
            M=1-DerIqDividit;
            G=DerIqDividit;
            }
      else
            {
            P=0;
            M=0;
            G=1;
            }
      }
Promig=P*RP+M*RPRT+G*RT;
Filtre=P*FB+M*FM+G*FS;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                      Fi Rutina Fuzzy                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

II.3 Programa "Graons.c"

```
/*****************************************************************/
/* GRAONS.C  Programa que implementa un llaç de control FAM sense  */
/*          sensor sobre el motor                                 */
/* Estima el valor de la velocitat amb l'estimador RP             */
/* Estima el valor de la velocitat amb l'estimador RT             */
/* Implementa el sistema Fuzzy per filtrar i per promitjar        */
/* Es tanca el llaç de control amb la velocitat estimada          */
/* El sistema genera respostes a consignes tipus graó             */
/*****************************************************************/
/* Aquest programa genera temps de 100us i a més corregeix en el
cas de que una rutina trigui més de 100us, la següent triga menys */

/* Inclusió de llibreries necessàries per la rutina de mesura de temps
i la de control */
#include <dos.h>
#include <time.h>
#include <math.h>
#include <conio.h>

/* Inclusió de llibreries necessàries per al control del DSP */
#include "c:\mon32\bc45\tic32.h"
#include<stdio.h>
#include<stdlib.h>

/* Es defineixen les posicions de memòria de la DPRAM a  on hi ha les  */
/* consignes del modulador vectorial */

#define  VREF       0xc0000dl   // Mòdul de Vref (Modulador vectorial)
#define  VDC        0xc0000el   // Valor del bus de contínua
#define  WM         0xc0000fl   // Freqüència del senyal (en rad/s)
#define  ZERO       0xc000101   // Bit indicant pas per zero del motor
#define  STOP       0xc000121   // Bit de stop del modulador
#define  TZUS       0xc000141   // Tz
#define  TMIN1      0xc000151    // Tmin
#define  CONTATGE   0xc000161
#define  FaseVREF   0xc000171   // Fase del vector VREF
#define  sextant    0xc000191   // Sextant de treball
#define  SFVS       0xc0001al    // correcció de la fase del FAM
#define  START      0xc0001cl   // Indica que el sistema ha engegat
#define  Posicio    0xc000261
#define  Velocitat  0xc000271
#define  Wm0        0xc000441   // Posició DPRAM canal0
#define  Wm1        0xc000451   // Posició DPRAM canal1
#define  Wm2        0xc000461   // Posició DPRAM canal2
#define  Wm3        0xc000471   // Posició DPRAM canal3
#define  PI 3.141592654

/* Es defineix l'arxiu que serà llençat per a que corri en el DSP  */
#define  FILENAME "MODUL.OUT"


/* Constants */
float RpmRads=0.2094395;      // Conversió de rpm a Rad/s
float taorotor=0.0634288;     // Lr/Rr = (lr+M)/Rr Nou motor
float taorotorinv=15.76570;   // Rr/Lr = Rr/(lr+M) Nou motor
float Tr=0.0634288;           // Lr/Rr = (lr+M)/Rr Nou motor
```

```
// FAM, EstRT
float ModV=1.0, DesfaseVI=0.0, Desfaseant=0.0, CsgnM=0.5, DifFase=0.0;
float ws=4.0,s=0.0,wm=0.0;
float Rs=4.3, Rr=5.05, Lm=0.3056, Lr=0.3203, Ls=0.3203, ImaM=2, p=2.0;
float Lsp=0.02872535;         // Lsp=sigma*Ls; sigma=1-(Lm^2/Ls*Lr);

// FAM Magnetització
float Vmagnet=30, Fasemagnet=1.371;

/* Variables de l'adquisició de dades DSP--->PC*/
float valorch0,valorch1,valorch2,valorch3,valorch4,valorch5;
float of0=-31.5,of1=-48.5,of2=-62.6,of3=-40.5; // Calibració adq. dades
unsigned long analog0,analog1,analog2,analog3,analog4,analog5;
short int tensio0,tensio1,tensio2,tensio3,tensio4,tensio5;
float conv=6.1035156e-5;            // Constant que converteix el valor
                            // entregat pel DSP en valor real


/* Variables del pas de paràmetres PC--->DSP */
unsigned long tmin=5,tz=100,tzminl,tzant; // Variables dels temps
unsigned long inici;                      //Variables d'inici
float  fc,m,tzmin;
unsigned long Vdc=537,Vref;         // Variables de les tensions
float fase;                   // Variable de l'increment de fase
                              // que ha d'imposar el modulador
float vel;                    //Pulsació angular passada al modulador

unsigned long stop=0;                 // Variable de posta en marxa

/* Variables de l'adquisició de dades DSP--->PC*/
unsigned long sex=0;          //Sextant actual imposat pel Modulador
float faseAgafa;        //Fase actual del modulador dins el 1r sextant
unsigned long EstaZero;       // Variable de Pas per Zero
float alfa;             //Angle que ha imposat el DSP a la sortida

/* Variables intermitges */
float freq;       // Pulsació angular
float freq2;            // Freqüència
float Usvm;
float FitaSvm=0;

float Vfaserms;
float VfasePic;

/* Variables Sensor-less */
float WestRP=0,WestRPant=0; // Vel (Rad/s) estimada per l'estimador RP
float WestRT=0,WestRTant=0; // Vel (Rad/s) estimada per l'estimador RT
float k=2.26;               // Constant de l'Estimador en RP
float UaRef=0, UbRef=0,UcRef=0;     // Tensió imposada a la fase A i B
float UaRefAnt=0, UbRefAnt=0,UcRefAnt=0;// Tensió imposada a la fase A,B
float FluxDispDsAnt=0, FluxDispQsAnt=0;// Variables de l'Estimador en RT
float FluxDrAnt=0, FluxQrAnt=0;     // Variables de l'Estimador en RT
float Kc=0.008;                     // Filtre 1 Estimador RP
float Kc2=0.001;              // Filtre 2 Estimador RT (Variable)
float Kc3=0.001;              // Filtre 2 Derivada de Iq
float Kc4=0.01;               // Filtre 3 Sortida final
float wx=0,wxAnt=0;           //Velocitat Estimada final
```

```
/* Variables sistema Derivació Iq */
float DerIq=0,DerIqAnt=0,DerIqFinal=0;
float iqAnt=0;

/* Variables sistema FUZZY Iq */
float P,M,G;
float RP=0.1,RPRT=0.4,RT=1;
float FB=0.0001,FM=0.0018,FS=0.0028;
float BaseFuzzy=20;
float Promig=0,Filtre;

// Model de l'inversor
int TensioPic=0;
float FactorTensio;

//Matriu de dades
float dades[5][2500];
int pujadades=0;
int pujadades2=0;

// Salts en el càlcul del FAM
int pujaFAM=1;

/* Variables d'us general */
char x='n';
int printa;
int FirstScan=1;

/* Consigna general */
float wcon=0;

/* Variables del mòdul PI Velocitat */
float iqcon;
float want;
float PI1max=20,PI1min=-20,BP1=2,Ti1=0.11;     // PIVelocitat

/* Variables mesurades (entrada al programa)*/
float ia,ib;
float w;
float w2;

/* Variables de sortida del mòdul Transformada Inversa de Park */
float id=0,iq=0;

/* Variables del mòdul Model del Motor */
float fi=0,imr=0;

/* Variables per la mesura de temps */
double estat1=0,estat2=0;
double temps,a,b;
unsigned int paraula,ticks;
unsigned int baix,alt;
double tempscicle,tempsciclereal=0;

double tempsdesig=0.00010;    // Temps de cicle
float Ts=0.00010;       // Temps d'integració

double tempsdif=0;
double acumulat=0;
float velocitat;
```

```
/* Definició de rutines */
void IniciDSP(void);
void PasParamIniciDSP(void);
void PasParamDSP(void);
void EngegaDSP(void);
void AturaDSP(void);
void LlegirDSP(void);
void FaseNova(void);
void PiVelocitat(void);
void CalculWestRP(void);
void CalculWestRT(void);
void FAM(void);
void ModelInversor(void);
void parkinv(void);
void motor(void);
void Derivada(void);
void Fuzzy(void);

void EscriuFitxer(void);

void main (void)
{
unsigned long puja=0;
unsigned long fipuja=3;
unsigned long fipla=0;
unsigned long fibaixa=0;
unsigned long fipla2=0;
unsigned long fipujaMagnet=0;
float tassaig;
float tmagnet=0.1;
int segons=0;
int voltes=0,voltes2=0;
tempscicle=tempsdesig;
clrscr();

printf("Entra temps de magnetització (s) 0.1 : ");
scanf("%f", &tmagnet);
fipujaMagnet=(unsigned long)(tmagnet/tempsdesig);
printf("\nEntra Tensió de magnetització (V) 30 : ");
scanf("%f", &Vmagnet);
printf("\nEntra fase de magnetització (Rad) 1.371 : ");
scanf("%f", &Fasemagnet);

printf("Entra temps d'espera 1(s): (Màxim 3 s)  ");
scanf("%f", &tassaig);
if (tassaig>3) tassaig=3;
fipla=(unsigned long)(tassaig/tempsdesig);

printf("Entra temps d'espera 2 (s): (Màxim 3 s)  ");
scanf("%f", &tassaig);
if (tassaig>3) tassaig=3;
fipla2=(unsigned long)(tassaig/tempsdesig);
printf("\nEntra la velocitat en RAD/S (100): ");
scanf("%f", &wtemp);

printf("\nPrem un '1' per començar");
x = getch();

/*****************************************************************/
/*              Inicialització del DSP                           */
/*****************************************************************/
```

```
IniciDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Inicialització del DSP              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*******************************************************************/
/*              Pas dels primers paràmetres                  */
/*******************************************************************/
PasParamIniciDSP();
freq=0*2*PI;      //2PI Rad/seg
Usvm=0;
FitaSvm=0;
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*              Fi Pas dels primers paràmetres         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*******************************************************************/
/*              Engega el DSP                  */
/*******************************************************************/
inici=0;
EngegaDSP();

do
{
Get_DPRAM_Word_32(START,&inici);
}
while (inici!=1);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*         Fi Engega el DSP                            */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

if (x=='1')
{
/*Primera mesura de temps */
disable();                  //Deshabilita int
ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
outport((0x43),(0x00));           //Escriptura perifèric(c,d)
enable();                   //Habilita int
baix=inportb(0x40);               //Lectura Perifèric
alt=inportb(0x40);                //Lectura Perifèric
paraula=(baix)|(alt)<<8;          //BytesAWord(b,a)
estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*Fi primera mesura de temps*/
/*********************************************************************/
/*          Magnetització de la màquina              */
/*                                                */
/*                                                */
/*                                                */
/*********************************************************************/
TensioPic=Vmagnet;               //Entrada al model de l'inversor
ModelInversor();
ModV=Vmagnet;                       //Entrada al Estim Reg Trans
Usvm=(Vmagnet*100)/81.4;
Usvm=abs(Usvm);
FitaSvm=Fasemagnet;                 //Entrada fase magnetitzant
freq=0;                          //Freqüència magnetitzant
gotoxy(25,6);
printf("Magneto");
```

```
 do
 {
/*****************************************************************/
/*     Temps d'espera                                            */
/*****************************************************************/
/* Part del programa que genera un temps d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                     // Línia a eliminar
            disable();                    //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
            outport((0x43),(0x00));       //Escriptura perifèric(c,d)
            enable();                     //Habilita int
            baix=inportb(0x40);           //Lectura Perifèric
            alt=inportb(0x40);            //Lectura Perifèric
            paraula=(baix)|(alt)<<8;      //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;          //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                  Fi Temps d'espera                            */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*           Mesura Temps inicial de la rutina                   */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                    //Deshabilita int
      ticks=*(unsigned long far *)(0x46c);     //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);                 //Lectura Perifèric
      alt=inportb(0x40);                  //Lectura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*          Actualització de sortida                             */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Actualització de sortida                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*          Rutina de Control Magnetització                      */
/*****************************************************************/
// Captura dels paràmetres
LlegirDSP();                        //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor
ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);
```

```
CalculWestRT();                     //Estimador RT

FitaSvm=0;              // Fase a zero per no anar incrementant

// Càlcul de la velocitat en rpm
w2=(valorch3/(6e-4))/9.55;           //Adequació velocitat

if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

// Visualització de dades (No necessari a la pràctica)
if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensió= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
// Fi visualització de les dades


/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control Magnetització                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*  FI Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
voltes2=voltes;
voltes=0;
puja++;
      }
while (puja<fipujaMagnet);

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          FINAL Magnetització de la màquina                    */
/*                                                               */
/*                                                               */
/*                                                               */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                LLAÇ de CONTROL PLA                            */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/


do
{
/*****************************************************************/
/*                Temps d'espera                                 */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                  // Línia a eliminar
            disable();                 //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));          //Escriptura perifèric(c,d)
            enable();                  //Habilita int
            baix=inportb(0x40);              //Lectura Perifèric
            alt=inportb(0x40);               //Lectura Perifèric
            paraula=(baix)|(alt)<<8;         //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;             //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*          Mesura Temps inicial de la rutina                    */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                                          //Deshabilita
int
      ticks=*(unsigned long far *)(0x46c);      //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                      //Habilita int
      baix=inportb(0x40);                 //LECTura Perifèric
      alt=inportb(0x40);                  //LECTura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                  */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                               */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=wtemp;
/*************************/
/*Fi càlcul consigna      */
/*************************/

// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                         //Sensor-less TOTAL
```

```
// Sensor-less RP
fi=alfa;
parkinv();                      //Càlcul de iq
Derivada();
CalculWestRP();                     //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                          //Estimador RT
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                        //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                  ModV
//             Consigna de Freqüència               ws
//                 Correcció de fase per si cal      DifFase

// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;               //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
```

```
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                 */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*******************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari             */
/*    en funció de l'error de temps acumulat                    */
/*******************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari                */
/*    en funció de l'error de temps acumulat                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipla);
}
```

```
/*****************************************************************/
/*                LLAÇ de CONTROL PLA2                           */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/

do
{
/*****************************************************************/
/*                Temps d'espera                                 */
/*****************************************************************/
/* Part del programa que genera una tems d'espera fins arribar
als microsegons marcats per tempscicle */
      do
            {
            voltes++;                   // Línia a eliminar
            disable();                  //Deshabilita int
            ticks=*(unsigned long far *)(0x46c);//Lectura memòria(c)
            outport((0x43),(0x00));           //Escriptura perifèric(c,d)
            enable();                   //Habilita int
            baix=inportb(0x40);               //Lectura Perifèric
            alt=inportb(0x40);                //Lectura Perifèric
            paraula=(baix)|(alt)<<8;          //BytesAWord(b,a)
      estat2=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
            temps=estat2-estat1;              //Càlcul del temps real
            }
      while (temps<tempscicle);
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Temps d'espera                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/




/*****************************************************************/
/*          Mesura Temps inicial de la rutina                    */
/*****************************************************************/
/* Part del programa que mesura el tems inicial de la rutina de control*/
      disable();                              /Deshabilita int
      ticks=*(unsigned long far *)(0x46c);      //Lectura memòria(c)
      outport((0x43),(0x00));             //Escriptura perifèric(c,d)
      enable();                     //Habilita int
      baix=inportb(0x40);                 //LECTura Perifèric
      alt=inportb(0x40);                  //LECTura Perifèric
      paraula=(baix)|(alt)<<8;            //BytesAWord(b,a)
      estat1=(double)(ticks/CLK_TCK)+(double)(65535-paraula)/1193180;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Mesura Temps inicial de la rutina                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*             Actualització de sortida                          */
/*****************************************************************/
PasParamDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*             Fi Actualització de sortida                       */
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*           Rutina de Control                                   */
/*****************************************************************/
      acumulat=acumulat+temps;
      tempsciclereal=temps*1000000;

/*************************/
/*Càlcul consigna         */
/*************************/
wcon=-wtemp;
/*************************/
/*Fi càlcul consigna      */
/*************************/


// Captura de velocitat i corrents
LlegirDSP();                          //I i Wreal
FaseNova();                 //Fase real imposada per modulador
ModelInversor();            //Model inversor

ia=7.0922*(valorch1);               //Adequació corrents
ib=7.0922*(valorch2);

// Càlcul de la velocitat
w2=(valorch3/(6e-4))/9.55;

wx=WestRP*(1-Promig)+WestRT*Promig;  //llaç tancat Sensor-less
wx=(Kc4*(wx-wxAnt))+wxAnt;
wxAnt=wx;
w=wx;                         //Sensor-less TOTAL

// Sensor-less RP
fi=alfa;
parkinv();                      //Càlcul de iq
Derivada();
CalculWestRP();                     //Càlcul de la velocitat
// Fi Sensor-less RP

// Sensor-less RT
CalculWestRT();                           //Estimador RT
// Fi Sensor-less RT

// Crida a una funció PI
PiVelocitat();
CsgnM=iqcon;
// La sortida del PI ,s consigna de parell

Fuzzy();

wm=wx;                        //Sensor-less TOTAL

FAM();

// El FAM retorna: Consigna de Tensió                  ModV
//             Consigna de Freqüència               ws
//                 Correcció de fase per si cal      DifFase
```

```
// Adequació de la sortida FAM
//freq=iqcon;   // freq = pulsació angular
            // valors 50 Hz = 314 Rad / s
            //        31 Hz = 200 Rad / s

freq=ws;                //FAM

freq2=freq/(2*PI);

//Correcció del mòdul de la tensió
TensioPic=ModV;
ModelInversor();

Vfaserms=ModV/sqrt(2);                          //FAM

if (Vfaserms>220) Vfaserms=220;         //Adequació a límits (Tensió)
if (Vfaserms<-220) Vfaserms=220;
VfasePic=Vfaserms*1.4142;           //Pas de paràmetres al modulador
Usvm=(VfasePic*100)/81.4;
Usvm=abs(Usvm);                     //Mòdul de la tensió
FitaSvm=DifFase;                    //Diferència de fase

if (freq>620) freq=620;                 //Adequació a límits (Freq)


if (pujadades2==15)
      {
      dades[0][pujadades]=w2;
      dades[1][pujadades]=WestRP;
      dades[2][pujadades]=WestRT;
      dades[3][pujadades]=wx;
      dades[4][pujadades]=Promig;
      pujadades2=0;
      pujadades++;
      }
pujadades2++;

if (printa==550)
      {
      gotoxy(1,8);
      printf("freq= %f",freq2);
      gotoxy(1,9);
      printf("Tensi¢= %f",Vfaserms);
      gotoxy(1,10);
      printf("fi= %f",fi);
      gotoxy(1,11);
      printf("ia= %f",ia);
      gotoxy(1,12);
      printf("ib= %f",ib);
      printf("DerIq= %f",DerIq);
      gotoxy(1,13);
      printf("id= %f",id);
      gotoxy(1,14);
      printf("iq= %f",iq);
      gotoxy(1,15);
      printf("imr= %f",imr);
      gotoxy(1,16);
      printf("Vel= %f",w2);
      gotoxy(1,17);
      printf("VeloRP= %f",WestRP);
      gotoxy(1,18);
```

```
      printf("VeloRT= %f",WestRT);
      gotoxy(1,19);
      printf("VeloPromig= %f",wx);
      gotoxy(1,20);
      printf("alfa= %f",(alfa));
      printa=0;
      }
      else
      {
      printa++;
      }
voltes2=voltes;
voltes=0;
puja++;
if (FirstScan==1) FirstScan=0;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi rutina de Control                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/******************************************************************/
/*    Càlcul de l'Ajust de temps d'espera necessari              */
/*    en funció de l'error de temps acumulat                     */
/******************************************************************/
      tempsdif=temps-tempscicle;
      tempscicle=tempsdesig-tempsdif;
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*    FI Càlcul de l'Ajust de temps d'espera necessari                */
/*    en funció de l'error de temps acumulat                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

      }
while (puja<fipla2);
}



/******************************************************************/
/*                Atura el DSP
*/
/******************************************************************/
AturaDSP();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura el DSP                                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/******************************************************************/
/*                Escriu Dades al fitxer                         */
/******************************************************************/
EscriuFitxer();
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Escriu Dades al fitxer                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


printf("\nVoltes= %i",voltes2);
printf("  Acum= %f segons",acumulat);
printf("\nPrem una tecla per sortir");
x=getch();
}
```

```
/*****************************************************************/
/*                Inicialització del DSP                         */
/*****************************************************************/
void IniciDSP(void)
{
/* Selecciona el tamany de dades amb el que treballarà el DSP  */
Set_Processor_Data_Type_Size_32();

/* Inicialitza el DSP amb la direcció base de la targeta (0x290), la */
/* direcció base de la DPRAM (0xD00), i el tamany del bus ISA  */
Select_Board(0x290, 0xd00, 8);

/* Habilita la DPRAM  */
Enable_DPRAM();

/* Situa el DSP en estat conegut  */
Global_Reset();

/* Carrega el fitxer que es desitja executar  */
Load_Object_File(FILENAME);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Inicialització del DSP                      */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Primer Pas de Paràmetres del DSP               */
/*****************************************************************/
/*
Entrada:    Vdc         tensió del bus en Volts
                tmin    temps mínim del pols en Microsegons
                tz        temps del període en Microsegons
Sortida:    VDC         Pas sobre el DSP Vdc
                TZUS    Pas sobre DSP de tz
                TMIN1 Pas sobre DSP de tmin
*/
void PasParamIniciDSP(void)
{
/* Passa al DSP el valor del bus de contínua Per defecte Vdc=537 */
Put_DPRAM_Word_32(VDC, Vdc);

/* Col·loquem al DSP un valor de tz=100 i tmin=5; */
Put_DPRAM_Word_32(TZUS, tz);
Put_DPRAM_Word_32(TMIN1, tmin);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Primer Pas de Paràmetres del DSP               */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Pas de Paràmetres al DSP                       */
/*****************************************************************/
void PasParamDSP(void)
{
/* Adequació de les variables per al pas de paràmetres/
```

```
Vref=(unsigned long)(Usvm);
fase=FitaSvm;
vel=freq;               //Pulsació angular

/* Posa la consigna Vref a la posició de memòria VREF de la DPRAM  */
Put_DPRAM_Word_32(VREF, Vref);
/* Actualitza la fase del vector Vref a la pos de DPRAM FaseVREF */
Put_DPRAM_Float_32(SFVS, fase); //Control V_F no cal fase
/* Posa la consigna vel a la posició de memòria WM de la DPRAM  */
Put_DPRAM_Word_32(WM, vel);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Pas de Paràmetres al DSP                              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Llegir DSP                                     */
/*****************************************************************/
/* Obtenció dels senyals de la DPRAM */
void LlegirDSP(void)
{

Get_DPRAM_Word_32(Wm1,&analog1);
Get_DPRAM_Word_32(Wm2,&analog2);
Get_DPRAM_Word_32(Wm3,&analog3);

tensio1 =(short int)analog1;
tensio2 =(short int)analog2;
tensio3 =(short int)analog3;

valorch1=tensio1;
valorch2=tensio2;
valorch3=tensio3;

if (FirstScan==1) of3=-valorch3;

valorch1 =-1.0*(valorch1+of1)*conv;
valorch2 =-1.0*(valorch2+of2)*conv;
valorch3 =-1.0*(valorch3+of3)*conv;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Llegir DSP                                         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Engega DSP                                     */
/*****************************************************************/
void EngegaDSP(void)
{ stop=0;
      Put_DPRAM_Word_32(STOP, stop);
      /* Executa el fitxer carregat  */
      Run_From(Get_Entry_PC());
      delay(2000);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Engega DSP                                         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*****************************************************************/
/*                Atura DSP                                      */
/*****************************************************************/
void AturaDSP(void)
{ stop=1;
      Put_DPRAM_Word_32(STOP, stop);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Atura DSP                                   */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                PI Velocitat  PI1                               */
/*****************************************************************/
/*
Entrades    want:       valor real anterior de w       (PiVelocitat(void))
            wcon:       valor desitjat de w      (Usuari)
            w:          valor real de w  (Estimada)
            tempscicle: temps de durada del cicle    (general)
Sortides    iqcon:      corrent prop al parell desitjat      (FAM(void))
            want: valor real actual de iq  (PiVelociatat(void))
*/
void PiVelocitat(void)
{
float incI,incP;
float error;
error=wcon-w;
incI=(temps*error)/(BP1*Ti1);
incP=(want-w)/BP1;

iqcon=iqcon+incI+incP;

/* Anti Wind-Up */
if (iqcon>PI1max)
      iqcon=PI1max;
if (iqcon<PI1min)
      iqcon=PI1min;
want=w;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi PI Velocitat                                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*          Rutina Transformada Inversa de Park                  */
/*****************************************************************/
/*
Entrades    fi: angle de posició del flux(angle  integrat)
            ia: corrent estatòric   (Del DSP)
            ib: corrent estatòric   (Del DSP)
            centvingraus            (constant)
            factor:                 (constant)
Sortides          iq: valor real de iq         (Estimador RP)
*/
void parkinv(void)
{
float cos1,cos2,cos3,sin1,sin2,sin3;
float centvintgraus=2.09436512;    // cent vint graus en radiants
```

```
float factor=0.8164965;         // sqrt(2/3)

cos1=cos(fi);
cos2=cos(fi-centvintgraus);
cos3=cos(fi+centvintgraus);
sin1=sin(fi);
sin2=sin(fi-centvintgraus);
sin3=sin(fi+centvintgraus);

iq=factor*((sin3-sin1)*ia+(sin3-sin2)*ib);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Rutina Transformada Inversa de Park                 */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                Agafa la Fase actual imposada pel DSP             */
/*****************************************************************/
void FaseNova(void)
{
/* Agafa la consigna fase que hi ha a FaseVREF de la DPRAM  */
Get_DPRAM_Word_32(sextant,&sex);
Get_DPRAM_Float_32(FaseVREF,&faseAgafa);
alfa=(faseAgafa)+(sex*1.0471975);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Agafa la Fase actual imposada pel DSP           */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Estimador en Regim Permanent                    */
/*****************************************************************/
void CalculWestRP(void)
{
WestRP=(freq/2)-(k*iq);
// Filtre
WestRP=(Kc*(WestRP-WestRPant))+WestRPant;
WestRPant=WestRP;
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Estimador en Regim Permanent                 */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                Derivada de Iq                                  */
/*****************************************************************/
void Derivada(void)
{
DerIq=(iq-iqAnt)/Ts;
DerIq=(Kc3*(DerIq-DerIqAnt))+DerIqAnt;
iqAnt=iq;
DerIqAnt=DerIq;
DerIqFinal=abs(DerIq);
}

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                Fi Derivada de Iq
*/
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Estimador en Regim Transitori                  */
/*****************************************************************/
void CalculWestRT(void)
{

float UsD,UsQ;
float IsD,IsQ;
float DerFluxDs,DerFluxQs;
float FluxDr, FluxQr;
float DerFluxDr,DerFluxQr;
float FluxDispDs, FluxDispQs, FluxDispDx;
float DerFluxDispDs, DerFluxDispQs;

/* Generador de UaRef, UbRef */

UaRef=UaRefAnt;
UbRef=UbRefAnt;
UaRefAnt=ModV*FactorTensio*sin(alfa+(PI/6));
UbRefAnt=ModV*FactorTensio*sin(alfa+(+(PI/6)-(2*PI/3)));

      /* Expressió: (2*Usa+Usb)/sqrt(3) */
/* Clarke Tensions i corrents */
UsD=UaRef;
UsQ = ( 2.0 * UaRef + UbRef ) / sqrt( 3.0);

      /* Expressió: (2*Isa+Isb)/sqrt(3) */
IsD=ia;
IsQ = ( 2.0 * ia + ib ) / sqrt( 3.0);

/* Derivades de Flux */
/* Expressió: UsD-Rs*IsD */
DerFluxDs = UsD - Rs * IsD;

/* Expressió: UsQ-Rs*IsQ */
DerFluxQs = UsQ - 4.3 * IsQ;

      /* Expressió: Lsp*IsD */
FluxDispDs = Lsp * IsD;
DerFluxDispDs=(FluxDispDs-FluxDispDsAnt)/Ts;
FluxDispDsAnt =FluxDispDs;

      /* Expressió: Lsp*IsQ */
FluxDispQs = Lsp * IsQ;

DerFluxDispQs=(FluxDispQs-FluxDispQsAnt)/Ts;
FluxDispQsAnt =FluxDispQs;

      /* Expressió: (Lm/Tr)*IsD */
FluxDispDx = (Lm / Tr) * IsD;

      /* Expressió: (DerFluxDr) */
DerFluxDr= (Lr/Lm)* (DerFluxDs-DerFluxDispDs);

      /* Expressió: (FluxDr) */
FluxDr=FluxDrAnt+(DerFluxDr*Ts);
FluxDrAnt=FluxDr;
```

```
      /* Expressió: (DerFluxQr) */
DerFluxQr= (Lr/Lm)* (DerFluxQs-DerFluxDispQs);

      /* Expressió: (FluxQr) */
FluxQr=FluxQrAnt+(DerFluxQr*Ts);
FluxQrAnt=FluxQr;

WestRT=(FluxDispDx-DerFluxDr-(FluxDr/Tr))/(FluxQr*p);
WestRT=(Filtre*(WestRT-WestRTant))+WestRTant;
WestRTant=WestRT;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*           Fi Estimador en Regim Transitori                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*          Rutina Escriu Dades al Fitxer                        */
/*****************************************************************/
void EscriuFitxer(void)
{
int j=0;
FILE *f_grf;

if ((f_grf=fopen("C:\\JORDI\\V_F3_1.GRF","w"))==0)
            {
            printf("No es pot obrir el fitxer");
            return;
            }
      else
            {
            for(j=0;j<pujadades;j++)
                  {
                  fprintf(f_grf, "%6.3f %6.3f %6.3f %6.3f %6.3f
\n",dades[0][j],dades[1][j],dades[2][j],dades[3][j],dades[4][j]);
                  }
            fclose(f_grf);
            }
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*          Fi Rutina Escriu Dades al Fitxer                              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



/*****************************************************************/
/*                             FAM                                */
/*****************************************************************/
/*
Entrades    CsgnM:  Consigna de Parell        (PiVelociatat(void))
            wn:valor de la velocitat del rótor (Sensor-less)
            Rs, Rr, Lm, Lr, Ls, ImaM, p;       (Paràmetres motor)
Sortides    ws:        Pulsació síncrona de sortida   (Al DSP)
          ModV:   Valor de la tensió de fase
          DifFase:    Actualització de la fase de la tensió
*/

void FAM(void)
{
double ca, cb, cc, sws1, sws2, inter;
```

```
/**** FAM/MAC *****/
/* càlcul de la consigna de sws a partir del parell. */
if ( CsgnM != 0.0)
      {
      ca = 2.0 * CsgnM * pow( Lr * Ls - Lm * Lm, 2);
      cb = -3.0 * p * 2.0 * Rr * pow( Ls * Lm * ImaM, 2);
      cc = 2.0 * CsgnM * pow( (Ls * Rr), 2);
      inter=(cb*cb) - (4*ca*cc);
      if (inter>0)
            {
            sws1 = ( -1.0 * cb - sqrt(inter) ) / (2*ca);
            sws2 = ( -1.0 * cb + sqrt(inter) ) / (2*ca);
            if (fabs(sws1)>fabs(sws2)) sws1=sws2; /* Es pren sws menor */
            }                                     /* en valor absolut  */

      else sws1=10.0;

      /* Càlcul de la pulsació. */
      ws = wm*p + sws1;
      s=sws1/ws;
      }

/* Càlcul vector tensió en valor eficaç. */
ModV=ImaM*sqrt(pow(sws1*Ls*Lr*Rs+ws*Ls*Ls*Rr,2)+pow(ws*Ls*sws1*(Ls*Lr-
Lm*Lm)-Ls*Rs*Rr,2));
ModV=ModV/sqrt(pow(sws1*(Ls*Lr-Lm*Lm),2)+pow(Ls*Rr,2));

/* Càlcul del desfase actual entre tensió i corrent magnetizant. */
DesfaseVI=atan2(sws1*ws*(Ls*Lr-Lm*Lm)-(Rs*Rr),ws*(s*Lr*Rs+Ls*Rr));
DesfaseVI=DesfaseVI - atan2(-Rr,sws1*(Lr-Lm*Lm/Ls));

/* Càlcul del salto de desfase tensi¢-corrent magnetizant entre */
/* l'instant actual i l'anterior. */

DifFase=DesfaseVI-Desfaseant;

Desfaseant=DesfaseVI;

if (ModV>311.0) ModV=311.0;
if (ws>650.0) ws=650.0;
if (ws<-650.0) ws=-650.0;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                       Fi FAM                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                       Rutina Model de l'inversor II           */
/*****************************************************************/

void ModelInversor(void)
{
      if (TensioPic<=50)
            {
            FactorTensio=0.003988*TensioPic+0.6136;
            }
      if ((TensioPic>50)&&(TensioPic<=100))
            {
            FactorTensio=0.00206*(TensioPic-50)+0.813;
            }
      if ((TensioPic>100)&&(TensioPic<=150))
```

```
            {
            FactorTensio=0.00056*(TensioPic-100)+0.916;
            }
      if ((TensioPic>150)&&(TensioPic<=200))
            {
            FactorTensio=0.00028*(TensioPic-150)+0.944;
            }
      if ((TensioPic>200)&&(TensioPic<=250))
            {
            FactorTensio=0.00026*(TensioPic-200)+0.958;
            }
      if ((TensioPic>250)&&(TensioPic<=300))
            {
            FactorTensio=0.0001*(TensioPic-250)+0.971;
            }
      if ((TensioPic>300)&&(TensioPic<=350))
            {
            FactorTensio=0.00008*(TensioPic-300)+0.976;
            }

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                  Fi Rutina Model de l'inversor              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                      Rutina Fuzzy                          */
/*****************************************************************/
void Fuzzy(void)
{
float DerIqDividit;
if (DerIqFinal<=BaseFuzzy)
      {
      DerIqDividit=DerIqFinal/BaseFuzzy;
      P=1-(DerIqDividit);
      M=DerIqDividit;
      G=0;
      }
else
      {
      if ((DerIqFinal>BaseFuzzy)&&(DerIqFinal<=2*BaseFuzzy))
            {
            DerIqDividit=(DerIqFinal-BaseFuzzy)/BaseFuzzy;
            P=0;
            M=1-DerIqDividit;
            G=DerIqDividit;
            }
      else
            {
            P=0;
            M=0;
            G=1;
            }
      }
Promig=P*RP+M*RPRT+G*RT;
Filtre=P*FB+M*FM+G*FS;
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                      Fi Rutina Fuzzy                */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

II.4 Programa "Modul.c"

```
/* Arxiu : MODUL.C   */
/* Descripció : Programa pel DSP que realitza la modulació vectorial del */
/*              pont i a mes mostreja senyals utilitzant la placa        */
/*              d'adquisició AM/D16QS                                    */
/*           Rutina d'interrupció timer 1  en asm. 9p en 'main'       */
/*             Variables d'entrada: - Tmin,Tz, vref, vdc, w, STOP.      */
/*             Sortides: 7 sortides digitals que governen el pont       */
/*             inversor.                                                */
/*           A més és capaç de seguir una fase imposada per             */
/*           pas de paràmetres                                        */
/*           Detecta el pas per zero del rótor                    */
/*           És capaç de llegir dades de la DIO Ports A,B,C       */
/*_____________________________________________________________________*/

/* S'inclou la capçalera de la llibreria on es troba la funció sinus */

#include <c:\proves\math.h>

/* Consignes del modulador totes ubicades en posicions de la DPRAM */
/* vref - mòdul vector tensió de referència. ( sqr(3/2)*amplitud fonamental) */
/* vdc - tensió del bus de continua  */
/* w - pulsació de la fonamental ( rd/s ) */
/* STOP - bit de parada  */
/* Tz - període durant el qual s'imposa en el pont inversor la tripleta */
/*        temps-estats. Expressat en microsegons  */
/* Tmin - temps a partir del qual l'estat temps no s'envia al pont i  */
/*        passa pel ae. Expressat en microsegons  */


#define posicio_memoria_vref 0x0c0000d
#define posicio_memoria_vdc 0x0c0000e
#define posicio_memoria_w 0x0c0000f
#define posicio_memoria_ZERO 0x0c00010
/*Línia nova Indicarà amb un 1 que el rótor acaba de passar pel zero*/
#define posicio_memoria_STOP 0x0c00012
#define posicio_memoria_Tz 0x0c00014
#define posicio_memoria_Tmin 0x0c00015
#define posicio_memoria_FaseVREF 0x0c00017
#define posicio_memoria_sextant 0x0c00018
#define posicio_memoria_Posicio 0x0c00026
#define posicio_memoria_Velocitat 0x0c00027
#define posicio_memoria_ch0 0x0c00044
#define posicio_memoria_ch1 0x0c00045
#define posicio_memoria_ch2 0x0c00046
#define posicio_memoria_ch3 0x0c00047
#define PI 3.141592654

/* Numero d'elements de la taula de sinus que va de 0 60§ */


/* #define NUM_EL_TAULA  1666 */
#define NUM_EL_TAULA  5000         /*Linea nova*/

/* DT8: cicles de timer a dins de la interrupció abans d'activar el timer
+ 8 de latència. S'ha de restar dels cicles que s'envien al Timer */

#define DT8 70

/* DT8R: cicles de timer que hi hauran entre una interrupció i la
              següent. Es el temps mínim que s'enviarà. L'anomenem també,
              Llindar_Soft */

#define DT8R 145

/* R: cicles de timer que hi ha entre que ja s'ha activat el timer1 i
```

```
es surt i es surt de la rutina cint10 (POP's inclosos). Aquest temps
ha estat determinat experimentalment. */

#define R 75

asm("  .data");

/* #define   RSTCTRL   00000601 */
asm("RSTCTRL .set 00000601h");
/* Paraula de control que engega el timer 1  */

/* #define   SETCTRL  000006c1 */
asm("SETCTRL .set 000006c1h");
/* Direcció registre de configuració de la DIO32  */

asm("COUNTDPRAM  .long 0c00016h");

/* Defineixo les adreces dels quatre canals analògics */

asm("Wm0            .long 0c00044h");
asm("Wm1            .long 0c00045h");
asm("Wm2            .long 0c00046h");
asm("Wm3            .long 0c00047h");

/* Direcció registre de control del timer 1  */

/*#define     TIMECTRL    00808030*/

asm("TIMECTRL.long 00808030h");

/* Direcció del DSPLINK2  */

/* #define posicio_DIO 0828000   */
asm("posicio_DIO    .long 00828000h");
asm("posicio_DIO2   .long 00828002h");
asm("posicio_DIO3   .long 00828004h");
asm("posicio_DIO4   .long 00828006h");

/*  Direcció registre període timer 1  */
asm("COUNTER .long 00808034h");
/* Direcció del registre de control del timer1 per al contatge */
/* #define PERIOD    00808038 */
asm("PERIOD  .long 00808038h");
/* Direcció del registre de control del timer1 per al període */

/* #define PORTREG 00828007 */
asm("PORTREG .long 00828007h");

/* Direcció registre de control de la DIO32  */
/* #define   CONTROL 00828005  */
asm("CONTROL .long 00828005h");

asm("bandera .long 0h");
asm("bandera9p  .long 0h");
asm("dio1    .long 00ff0000h");
asm("dio2    .long 00ff0000h");
asm("dio3    .long 00ff0000h");
asm("dio1p   .long 00ff0000h");
asm("dio2p   .long 00ff0000h");
asm("dio3p   .long 00ff0000h");
asm("c1    .long 450");
asm("c2    .long 450");
asm("c3    .long 450");
asm("c1p    .long 450");
asm("c2p    .long 450");
asm("c3p    .long 450");
asm("et1    .long 3h");
asm("et2    .long 3h");
```

```
asm("pointerdio  .long dio1");
asm("addio   .long dio1");
asm("addiop  .long dio1p");
asm("primer .long 1h");
asm("primerp .long 0h");
asm("valor   .long 0h");
asm("salida  .long valor");
asm("valor2   .long 0h");
asm("salida2  .long valor2");

asm("STR0A          .word  00808064h");
asm("STR0D          .word  004F0900h");
asm("STR1A          .word  00808068h");
asm("STR1D          .word  00070900h");
asm("IOSTRA         .word  00808060h");
asm("IOSTRD         .word  00000000h");

asm("AMELIA0 .word  0081E000h");

asm("ITTP           .word  06000000h");

asm("Mascara    .word   0000ffffh");

/* Mapa de registres del AMELIA2 */

asm("DR3            .equ   01h");
asm("DR0     .equ    02h");
asm("DR2            .equ   03h");
asm("TMR0           .equ   04h");
asm("TMR1           .equ   05h");
asm("DR1     .equ    06h");
asm("IMR     .equ    07h");
asm("CTR     .equ    08h");
asm("ECTR    .equ    09h");
asm("SMR     .equ   0ah");
asm("CMR     .equ   0ah");
asm("ISR     .equ    0bh");
asm("DCR            .equ    0ch");
asm("SDC     .equ    0dh");
asm("ECFR    .equ    0eh");
asm("CFR     .equ    0fh");


/* Secció de programa anomenada 'int01' que es col·loca en el vector  */
/* d'interrupcions. La direcció de la rutina de servei te l'etiqueta  */
/* c_int01  */

asm("    .sect \".int01\"");
asm("    .word _c_int01");

/* Seccio de programa anomenada 'int03' que es col·loca en el vector  */
/* d'interrupcions. La direcció de la rutina de servei te l'etiqueta  */
/* c_int03  */

asm("    .sect \".int03\"");
asm("    .word _c_int03");


/* Seccio de programa anomenada 'int05' que es col·loca en el vector  */
/* d'interrupcions. La direcció de la rutina de servei te l'etiqueta  */
/* c_int05  */

asm("    .sect \".int05\"");
asm("    .word _c_int05");

/* Seccio de programa anomenada 'int10' que es col·loca en el vector  */
/* d'interrupcions. La direcció de la rutina de servei te l'etiqueta  */
/* c_int10  */
```

```
asm("    .sect \".int10\"");
asm("    .word  _c_int10");

/* Indica la finalització del codi per a la localització de les seccions que  */
/* s'han creat pel vector d'interrupcions  */

asm("    .text");

/*  Es defineixen punters  */

long * pointer=(long *)posicio_memoria_vref;
long * pointer1=(long *)posicio_memoria_vdc;
/* float * pointer2=(float *)posicio_memoria_w; */
long * pointer2=(long *)posicio_memoria_w;
long * pointerZERO=(long *)posicio_memoria_ZERO;
/*Linea nova */
long * pointerSTOP=(long *)posicio_memoria_STOP;
long * pointertz=(long *)posicio_memoria_Tz;
long * pointertmin=(long *)posicio_memoria_Tmin;
long * pointerch0=(long *)posicio_memoria_ch0;
long * pointerch1=(long *)posicio_memoria_ch1;
long * pointerch2=(long *)posicio_memoria_ch2;
long * pointerch3=(long *)posicio_memoria_ch3;
/* float * pointerFaseVREF=(float *)posicio_memoria_FaseVREF; /*Linea nova */
long * pointerFaseVREF=(long *)posicio_memoria_FaseVREF; /*Linea nova */
long * pointersextant=(long *)posicio_memoria_sextant; /*Linea nova */

long * pointerPosicio=(long *)posicio_memoria_Posicio; /*Linea nova */
long * pointerVelocitat=(long *)posicio_memoria_Velocitat; /*Linea nova */


/* Variable que conte el valor de la fase. S'inicialitza a 60° = 1.0471975 rd */
/* perquè quan entri per primera vegada a la rutina nou_periode, es quedarà   */
/* a 0 i la variable sextant quedarà a 1 ( sex =1)  */

float f=1.0471975;

/* S'inicialitzen totes les variables  */
/* ei/ci - tripleta estat/temps actual que s'envia al pont inversor i al  */
/*         timer 1 respectivament   */
/* eis/cis - tripleta estat/temps calculada següent  */
/* sex - indica el sextant en el qual estem */
/* i - indica en quin estat/temps estem dins de dues tripletes  */
/* p - indica en quin estat/temps es cridarà nou_periode  */
/*     dins de dues tripletes  */

long int sex=0;
int e1=0x95,e2=0x95,e7=0x95,e0=0xaa,c1=0,c2=0,c7=0;/*i=0*/
int e1s=0x95,e2s=0x95,e7s=0x95,e0s=0xaa,c1s=450,c2s=450,c7s=450;/*p=3*/
int c1t=0,c2t=0,c7t=0;
int cuenta=0;
int STOP=0;
int ta=0,nant=3,dt8c=0;   /*Linea nova */
long int dt8=0;                  /*Linea nova */
long int * pdt8=&dt8;   /*Linea nova */

/* array de floats per guardar els valors de la taula de sinus */

/*float taula[NUM_EL_TAULA];*/
float taula[NUM_EL_TAULA+1];  /*Linea nova */

/* declarem les diferents funcions  */


void c_int01(void);     /* Interrupció de l'Amèlia*/
void c_int03(void);     /* Interrupció de la DIO Entrada EXTRIG0*/
void c_int05(void);
```

```
void c_int10(void);

void taula_sin(void);           /* Inicialització de taula de sinus
(Moduladors)*/
void set_c32_timer1(void);/* Inicialització del timer (Modulador)*/
void set_amd16qs(void);         /* Inicialització de la placa de ADC   */
void dio32(void);               /* Inicialització de la DIO            */



/*************************************************************************/
/*                          Inici Programa Principal                     */
/*************************************************************************/
void main(void)
{
*pointerFaseVREF=0;
*pointerZERO=0;

asm(" LDP STR0A,DP");
asm(" LDP @STR0A,AR0");
asm(" LDP @STR0D,R0");
asm(" STI R0,*AR0");

asm(" LDP STR1A,AR0");
asm(" LDP STR1D,R0");
asm(" STI R0,*AR0");

asm(" LDP IOSTRA,AR0");
asm(" LDP IOSTRD,R0");
asm(" STI R0,*AR0");


/*************************************************************************/
/* Inicicalització de l'entorn DSP                                       */
/*************************************************************************/

      dio32();

/* enviem al pont inversor un estat nul per no cortcircuitar una fase
/* del motor tot desplaçant-ho prèviament 16  bits cap a l'esquerra  */

      taula_sin();
      set_amd16qs();
      set_c32_timer1();

/* Es resetejen les interrupcions pendents */
      asm("  LDI @AMELIA0,AR0");
      asm("  LDI *+AR0(ISR),R0");

      asm("  LDI @ITTP,IF");

/* Habilitem la interrupció INT0 de la PC/C32 */
      asm("  LDI  0001h,IE");
/* Habilitem la interrupció INT2 de la PC/C32 */
      asm("   OR 004h,IE");
/* Habilitem la interrupció TIMER1 de la PC/C32 */
      asm("   OR 200h,IE");
/* Habilitem la interrupció XINT0 de la PC/C32 */
      asm("   OR 010h,IE");
/* Habilitem interrupcions  globals */
      asm("   OR 2000h,ST");

/* Posem en marxa la Cache */
      asm("  OR 0800h,ST");

/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/* FI Inicicalització de l'entorn DSP                                    */
```

```
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/



      do {
            STOP=*pointerSTOP;
            }
      while (STOP==0);

/* Deshabilitem interrupcions */

      asm("    AND 0000h,IE");
      asm("    AND 0000h,ST");

      asm("  LDI 00ffh,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                    Fi Programa Principal                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*************************************************************************/
/* Funció : c_int01                                                      */
/* Propòsit : Funció a la qual es salta quan l'AMELIA2           */
/*           sol·licita interrupció                          */
/* Retorna : res                                                         */
/*************************************************************************/
void c_int01(void)
{
asm("  PUSH AR0");
asm("  PUSH AR1");
asm("  PUSH AR2");
asm("  PUSH R0");
asm("  PUSH R1");
asm("  PUSH R2");
asm("  PUSH R3");
asm("  PUSH R4");

asm(" ANDN 2000h,ST");

asm("  LDI @AMELIA0,AR0");
asm("  LDI  *+AR0(ISR),R0");

/* Només llegint en els canals s'anul·la la causa de la interrupció */
asm("  LDI 0000h,R1");
asm("  LSH 16,R1");
asm("  LDI 0000h,R2");
asm("  LSH 16,R2");
```

```
asm("  LDI 0000h,R3");
asm("  LSH 16,R3");
asm("  LDI 0000h,R4");
asm("  LSH 16,R4");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  ASH -16,R0");
/* asm("  LDI R0,R1"); */
asm("  ADDI R0,R1");

asm("  LDI *+AR0(DR1),R0");
asm("  ASH -16,R0");
/* asm("  LDI R0,R2");  */
asm("  ADDI R0,R2");

asm("  LDI *+AR0(DR2),R0");
asm("  ASH -16,R0");
/* asm("  LDI R0,R3"); */
asm("  ADDI R0,R3");

asm("  LDI *+AR0(DR3),R0");
asm("  ASH -16,R0");
/* asm("  LDI R0,R4"); */
asm("  ADDI R0,R4");


/* Lectura de la velocitat a través de la Dinamo Tacomètrica */
asm("  LDI @_pointerch0,AR2");
asm("  STI R1,*AR2");

/* Lectura del corrent de la fase A */
asm("  LDI @_pointerch1,AR2");
asm("  STI R2,*AR2");

/* Lectura del corrent de la fase B */
asm("  LDI @_pointerch2,AR2");
asm("  STI R3,*AR2");

/* Lectura no emprada de moment */

asm("  LDI @_pointerch3,AR2");
asm("  STI R4,*AR2");


/* Lectura dels ports B, C i D i col·locació a la DPRAM */
asm("  LDI @posicio_DIO2,AR0");  /*Port B: Part baixa de la posició R0 */
asm("  LDI *AR0,R0");
asm("  LDI @posicio_DIO4,AR0");  /*Port D: Compartit X,S,v9,v8,X,p10,p9,p8*/
asm("  LDI *AR0,R1");
asm("  LDI @posicio_DIO3,AR0"); /*Port C: Part baixa de la velocitat R3 */
asm("  LDI *AR0,R3");
/* Millora a realitzar: Direccinament indexat */
```

```
asm("  LDI 0000h,R2");
asm("  LSH 16,R2");
asm("  ADDI R1,R2");

asm("  ASH -16,R0");
asm("  ASH -8,R1");
asm("  AND 0700h,R1");
asm("  ADDI R1,R0");

asm("  ASH -16,R3");
asm("  ASH -12,R2");
asm("  AND 0700h,R2");
asm("  ADDI R3,R2");


asm("  LDI @_pointerPosicio,AR1");
asm("  STI R0,*AR1");

asm("  LDI @_pointerVelocitat,AR1");
asm("  STI R2,*AR1");

asm("  POP R4");
asm("  POP R3");
asm("  POP R2");
asm("  POP R1");
asm("  POP R0");
asm("  POP AR2");
asm("  POP AR1");
asm("  POP AR0");

asm(" ANDN 1h,IF");
}
/*************************************************************************/
/*                  Fi Funció : c_int01                                  */
/*************************************************************************/



/*************************************************************************/
/* Funció : c_int03
/* Propòsit : Funció a la qual es salta quan la DIO32 demana interrupció */
/* Retorna : col·loca un 1 a la posició de la DPRAM 0x0c00010
*/
/*************************************************************************/
void c_int03(void)
{
       asm(" ANDN 2000h, ST");

       asm(" PUSH AR0");
       asm(" PUSH AR1");
       asm(" PUSH R0");
       asm(" PUSH R1");

       asm(" LDI @CONTROL,AR1");
       asm(" LDI *AR1,R0");

       asm(" ANDN 004h,IF");
       asm(" OR 2000h,ST");
/* Part del programa que col·loca un 1 a un bit del port c */
/* En modul.c deshabilitada aquesta part perquè el port C ,s d'entrada */
/*
       asm(" LDI @salida2,AR0");
       asm(" LDI *AR0,R0");
       asm(" NOT R0,R0");
       asm(" STI R0,*AR0");
       asm(" LSH 16,R0");
       asm(" LDI @posicio_DIO3,AR1");
       asm(" STI R0,*AR1");
```

```
*/

/* Pas per zero */
/* Es col·loca un 1 a la DPRAM 0x0c00010 quan es detecta el pas per zero */
/* del rótor */
       asm(" LDI 0000h,R1");
       asm(" LSH 16,R1");
       asm(" LDI 0001h,R1");
       asm(" LDI @_pointerZERO,AR1");
       asm(" STI R1,*AR1");
/* Fi de la col·locació de 1 per indicar pas per ZERO*/

       asm(" POP R1");
       asm(" POP R0");
       asm(" POP AR1");
       asm(" POP AR0");
}
/*************************************************************************/
/*           Fi Funció : c_int03                                         */
/*************************************************************************/


/*************************************************************************/
/* Funció : c_int05 (nou_periode)
/* Propòsit : S'actualitza la fase i sextant i es defineix la tripleta  */
/*            d'estats calculats.   */
/*            Es calculen la tripleta de temps calculats.  */
/*            Es determina a quin dels tres estats temps hi haurà temps  */
/*            de tornar a calcular la nova tripleta. */
/*            Si el bit de AAEE es actiu (ae=0) es passa per l'algorisme  */
/*            d'arrossegament d'error.  */
/*           Retorna : res  */
/*************************************************************************/
void c_int05(void)
{   float tmin,stz,tzus,ae,vref,vdc,w,x,tsin,itf,fase,svv;
       long fasePasa;
       float faseVectorial;
       int it;
       long sextant;

       asm(" ANDN 2000h, ST");
       asm(" ANDN 010h,IE");
       asm(" ANDN 010h,IF");

       /* Es capturen les noves consignes mentre les interrupcions
estan deshabilitades.  */

       vref=*pointer;
       vdc=*pointer1;
       w=*pointer2;
       tzus=*pointertz;
       fasePasa=*pointerFaseVREF;
       faseVectorial=(float)fasePasa;

       asm(" OR 2000h,ST");

       /*pas de tz de microsegons a segons*/

       stz=0.000001*tzus;

       /* S'actualitza la fase. */

if (c1s>0) nant=1;
if (c2s>0) nant=nant+1;
if (c7s>0) nant=nant+1;

if (faseVectorial==0.0)
             {
```

```
            /*
            f=f+(w*0.00000008*(c1s+c2s+c7s)) ;
            fase=w*(0.00000008*DT8);
            if (c1s>0) f=f+fase;
            if (c2s>0) f=f+fase;
            if (c7s>0) f=f+fase;
            */
            f=f+(w*0.00000008*(c1s+c2s+c7s+dt8c+7*nant)) ;
            dt8c=dt8c/nant;
            nant=0;
            }
      else
            {
            f=faseVectorial*0.000001;
            faseVectorial=0;
            *pointerFaseVREF=0;
/* Línies noves que pretenen que cada cop que se li introdueix
la fase vagi a parar al mateix lloc*/
            dt8=0;
            dt8c=0;
            nant=0;
            sex=0;
            }
do
      {
            if ( f>= 1.0471975)
                  {
                  sex++;
                  c1t=c2t;
                  c2t=0;
                  f=f-1.0471975;
                  switch(sex)
                        {
                        case 1:
                        e0s=0xaa;
                        e7s=0x95;
                        e1s=0xa9;
                        e2s=0xa5;
                        break;
                        case 2:
                        e0s=0x95;
                        e7s=0xaa;
                        e1s=0xa5;
                        e2s=0xa6;
                        break;
                        case 3:
                        e0s=0xaa;
                        e7s=0x95;
                        e1s=0xa6;
                        e2s=0x96;
                        break;
                        case 4:
                        e0s=0x95;
                        e7s=0xaa;
                        e1s=0x96;
                        e2s=0x9a;
                        break;
                        case 5:
                        e0s=0xaa;
                        e7s=0x95;
                        e1s=0x9a;
                        e2s=0x99;
                        break;
                        case 6:
                        sex=0;
                        e0s=0x95;
                        e7s=0xaa;
                        e1s=0x99;
```

```
                        e2s=0xa9;
                        break;
                        }
                }
        }
while (f>1.0471975);
do
        {
              if ( f< 0)
                        {
                        sex--;
                        c2t=c1t;
                        c1t=0;
                        f=f+1.0471975;
                        switch(sex)
                                {
                                case 1:
                                e0s=0xaa;
                                e7s=0x95;
                                e1s=0xa9;
                                e2s=0xa5;
                                break;
                                case 2:
                                e0s=0x95;
                                e7s=0xaa;
                                e1s=0xa5;
                                e2s=0xa6;
                                break;
                                case 3:
                                e0s=0xaa;
                                e7s=0x95;
                                e1s=0xa6;
                                e2s=0x96;
                                break;
                                case 4:
                                e0s=0x95;
                                e7s=0xaa;
                                e1s=0x96;
                                e2s=0x9a;
                                break;
                                case -1:
                                sex=5;
                                e0s=0xaa;
                                e7s=0x95;
                                e1s=0x9a;
                                e2s=0x99;
                                break;
                                case 0:
                                e0s=0x95;
                                e7s=0xaa;
                                e1s=0x99;
                                e2s=0xa9;
                                break;
                                }
                        }
        }
while (f<0.0);


        /*  es troba el valor del sin a la taula  */

        itf=f*(NUM_EL_TAULA/(PI/3));
        it=(int)itf;

        tsin=taula[it];
        svv=stz*vref*17677669.53/vdc;

        /* es calcula:  */
```

```
        /* c2s= tzs*(vref/vdc)*sin(angle)*(sqr(3/2))/sin60°*1000000  */
        /* EXPRESSAT EN MICRO SEGONS */
        /* a on :
        /*      - sin(angle)=tsin */
        /*      - sqr(3/2)/sin60°=sqr2  */
        /* c2s=1E6*tzs*(vref/vdc)*tsin*sqr2   EXPRESSAT EN MICROSEGONS  */
        /* c2s=1E6*tzs*(vref/vdc)*tsin*sqr2*1250ct/100microsegons  */
        /* EXPRESSAT EN CICLES TIMER1 =ct  */
        /* c2s= tzs(vref/vdc)*tsin*17677669.53   EXPRESSAT EN CICLES TIMER1 */

        c2s=(int)(tsin*svv);

        itf=((PI/3)-f)*(NUM_EL_TAULA/(PI/3));
        it=(int)itf;
        tsin=taula[it];
        c1s=(int)(tsin*svv);

        /* c7s=((1250/100)*tzus)-c1s-c2s; tot en cicles timer1  */
        /* 1250ct=100microsegons=Tz  */

        c7s=(int)((12.5*tzus)-c1s-c2s);
        if (c7s<0) c7s=0;

        /* Si en la tripleta calculada hi ha algun estat que dura menys de  */
        /* Tmin, no s'envia i s'acumula a cit. Quan un cit, es  */
        /* mes gran de Tmin aquest es passa a cis i s'envia  */
        /* tmin es variable */

tmin=*pointertmin;
tmin=tmin*12.5;

if (c1s <= (tmin+R+7+dt8c))
        {
        c1t=c1t+c1s;
        c1s=0;
        if (c1t>(tmin+R+7+dt8c))
                {
                c1s=c1t-dt8c-7;
                c1t=0;
                }
        }
else
        {
        c1s=c1s-dt8c-7;
        }

if (c2s <= (tmin+R+7+dt8c))
        {
        c2t=c2t+c2s;
        c2s=0;
        if (c2t>(tmin+R+7+dt8c))
                {
                c2s=c2t-dt8c-7;
                c2t=0;
                }
        }
else
        {
        c2s=c2s-dt8c-7;
        }

if (c7s <= (tmin+R+7+dt8c))
        {
        c7t=c7t+c7s;
        c7s=0;
        if (c7t>(tmin+R+7+dt8c))
                {
                c7s=c7t-dt8c-7;
```

```
              c7t=0;
              }
       }
else
       {
       c7s=c7s-dt8c-7;
       }

asm(" LDI 0h,R1");
asm(" LDI @_sex,R2");
asm(" ROR R2");
asm(" BC SENAR");
asm(" LDI @bandera9p,R2");
asm(" OR R2,R2");
asm(" BZ ESTAT2");
asm(" LDI @addio,AR0");

asm(" LDI @_c1s,R2");
asm(" OR R2,R2");
asm(" BZ SC2");

asm(" LDI @_e1s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c1s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SC2: LDI @_c2s,R2");
asm(" OR R2,R2");
asm(" BZ SC3");

asm(" LDI @_e2s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c2s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SC3: LDI @_c7s,R2");
asm(" OR R2,R2");
asm(" BZ FINAL1");

asm(" LDI @_e7s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c7s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");
asm(" BR FINAL1");

asm("ESTAT2: LDI @addiop,AR0 ");

asm(" LDI @_c2s,R2");
asm(" OR R2,R2");
asm(" BZ SCP2");

asm(" LDI @_e2s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c2s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SCP2: LDI @_c1s,R2");
asm(" OR R2,R2");
asm(" BZ SCP3");
```

```
asm(" LDI @_e1s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c1s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SCP3: LDI @_c7s,R2");
asm(" OR R2,R2");
asm(" BZ FINAL2");

asm(" LDI @_e0s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c7s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");
asm(" BR FINAL2");

asm("SENAR: LDI @bandera9p,R2");
asm(" OR R2,R2");
asm(" BZ ESTAT2S");
asm(" LDI @addio,AR0");

asm(" LDI @_c2s,R2");
asm(" OR R2,R2");
asm(" BZ SC2S");

asm(" LDI @_e2s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c2s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SC2S: LDI @_c1s,R2");
asm(" OR R2,R2");
asm(" BZ SC3S");

asm(" LDI @_e1s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c1s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SC3S: LDI @_c7s,R2");
asm(" OR R2,R2");
asm(" BZ FINAL1");

asm(" LDI @_e0s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c7s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");
asm(" BR FINAL1");

asm("ESTAT2S: LDI @addiop,AR0 ");

asm(" LDI @_c1s,R2");
asm(" OR R2,R2");
asm(" BZ SCP2S");

asm(" LDI @_e1s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c1s,R0");
```

```
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SCP2S: LDI @_c2s,R2");
asm(" OR R2,R2");
asm(" BZ SCP3S");

asm(" LDI @_e2s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c2s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");

asm("SCP3S: LDI @_c7s,R2");
asm(" OR R2,R2");
asm(" BZ FINAL2");

asm(" LDI @_e7s,R0");
asm(" LSH 16,R0");
asm(" STI R0,*AR0++(1)");
asm(" LDI @_c7s,R0");
asm(" STI R0,*+AR0(5)");
asm(" ADDI 1,R1");
asm(" BR FINAL2");

asm("FINAL1: STI R1,@et1");
asm(" BR FINAL3");
asm("FINAL2: STI R1,@et2");
asm("FINAL3: OR 010h,IE");
}
/*************************************************************************/
/*               Fi Funció : c_int05 (nou_periode)                       */
/*************************************************************************/



/*************************************************************************/
/* Funció : c_int10                                                      */
/* Propòsit : Funció a la qual es salta quan el timer 1 sol·licita int.  */
/*     Llegeix la taula estat/temps tot enviant temps timer1 i estat */
/*     a la DIO 32                                                       */
/* Retorna : res                                                         */
/*************************************************************************/
void c_int10(void)
{
asm(" ANDN 2000h,ST");
asm(" PUSH AR0");
asm(" PUSH AR1");
asm(" PUSH AR2");
asm(" PUSH R0");
asm(" PUSH R1");
asm(" PUSH R2");

asm(" LDI @primer,R1");
asm(" OR R1,R1");
asm(" BZ SALT1");
asm(" LDI @bandera,R1");
asm(" STI R1,@bandera9p");
asm(" LDI 0h,R0");
asm(" STI R0,@primer");
asm(" LDI 1h,R0");
asm(" STI R0,@primerp");

       /* Envia combinació estat/temps a la DIO_32/Timer1 */

asm("SALT1: LDI @pointerdio,AR0");
```

```
asm(" LDI *AR0,R0");
asm(" LDI @posicio_DIO,AR1");
asm(" STI R0,*AR1");


       /* Atura el timer1 per poder llegir-lo */
asm(" LDI @TIMECTRL,AR2");
asm(" LDI RSTCTRL,R0");
asm(" STI R0,*AR2");



asm(" LDI @COUNTER,AR1");
asm(" LDI *AR1,R0");
asm(" LDI @_dt8,R1");
asm(" ADDI R0,R1");
asm(" LDI @_pdt8,AR1");
asm(" STI R1,*AR1");

/* asm(" LDI @COUNTDPRAM,AR1");
asm(" STI R0,*AR1"); */


asm(" LDI *+AR0(6),R0");
asm(" LDI @PERIOD,AR1");
asm(" STI R0,*AR1");

       /* Engega el Timer1 */

asm(" LDI 06C1h,R0");
asm(" STI R0,*AR2");

asm(" LDI @bandera,R2");
asm(" OR R2,R2");
asm(" BZ FLAG0");
asm(" LDI @et2,R1");
asm(" SUBI 1,R1");
asm(" BZ FLAG10");
asm(" STI R1,@et2");
asm(" ADDI 1,AR0");
asm(" STI AR0,@pointerdio");
asm(" BR NOUP");
/* asm("FLAG10: STI R1,@et2");  */
asm("FLAG10: NOT @bandera,R0");
asm(" STI R0,@bandera");
asm(" LDI @addio,AR0");
asm(" STI AR0,@pointerdio");
asm(" LDI 01h,R1");
asm(" STI R1,@primer");
asm(" BR NOUP");

asm("FLAG0: LDI @et1,R1");
asm(" SUBI 1,R1");
asm(" BZ FLAG00");
asm(" STI R1,@et1");
asm(" ADDI 1,AR0");
asm(" STI AR0,@pointerdio");
asm(" BR NOUP");
/* asm("FLAG00: STI R1,@et1"); */
asm("FLAG00: NOT @bandera,R0");
asm(" STI R0,@bandera");
asm(" LDI @addiop,AR0");
asm(" STI AR0,@pointerdio");
asm(" LDI 01h,R1");
asm(" STI R1,@primer");

asm("NOUP: LDI @primerp,R1");
asm(" OR R1,R1");
```

```
asm(" BZ FI");
asm(" LDI 0h,R1");
asm (" STI R1,@primerp");

/* Es crida la funció nou_periode per al càlcul de la tripleta següent */
/* Aquesta funció la cridem a traves d'interrupcions: c_int05          */
/* La interrupció XINT0 te com a funció associada la subrutina c_int05 */

asm(" OR 010h,IF");

asm(" ANDN 200h,IF");
asm("FI: POP R2");
asm(" POP R1");
asm(" POP R0");
asm(" POP AR2");
asm(" POP AR1");
asm(" POP AR0");
}
/*************************************************************************/
/*                                        FI Funció : c_int10
*/
/*************************************************************************/



/*************************************************************************/
/* Inici de les rutines que nom,s serveixen per inicialitzar             */
/* les plaques                                                           */
/* Cada una d'elles nom,s s'executa un cop                               */
/*************************************************************************/


/*************************************************************************/
/* Funció : taula_sin
/* Propòsit : crea la taula de sinus */
/* Retorna : res  */
/*************************************************************************/
void taula_sin(void)

{   float s=0,r=0;
       int k=1;
       for(k=0;k<NUM_EL_TAULA;k++)
              {   s=(float)k*(PI/3)/((float)NUM_EL_TAULA);
                  r=sin(s);
                  taula[k]=r;
              }
}
/*************************************************************************/
/*                                Fi Funció taula_sin
*/
/*************************************************************************/




/*************************************************************************/
/* Funció : set_c32_timer1
                          */
/* Propòsit : Es programa el timer 1 perquè sol·liciti interrupció      */
/*            quan hagi passat  COUNT=ci
            */
/* Retorna : res
                          */
/*************************************************************************/
void set_c32_timer1(void)
{
       /* atura el timer */
       asm(" LDI @TIMECTRL,AR0");
```

```
       asm(" LDI RSTCTRL,R0");
       asm(" STI R0,*AR0");

       /* carrega el nou COUNT al timer  */

       asm(" LDI @PERIOD,AR1");
       asm(" LDI 1250,R0");
       asm(" STI R0,*AR1");

       /* engega el timer  */

       asm(" LDI SETCTRL,R0");
       asm(" STI R0,*AR0");
}
/*************************************************************************/
/*                                    Fi Funció : set_c32_timer1
                          */
/*************************************************************************/

/*************************************************************************/
/* Funció :  set_amd16qs                                                 */
/* Propòsit: Funció d'inicialitzacio de tots els registres corresponents */
/*           al AMELIA2, per a la adquisició de dades amb la placa       */
/*           adquisició AM/D16QS.                                      */
/* Retorna : res.                                                        */
/*************************************************************************/
void set_amd16qs(void)
{
asm("  LDI @AMELIA0,AR0");

asm("  LDI 0010h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(DCR)");

/* Posem la placa AM/D16QS en reset */
asm("  LDI 0000h,R0");
asm("  STI R0,*+AR0(CMR)");

asm("  STI R0,*+AR0(CFR)");

/* Configurem el User Control register */
/* Amb el valor 28E3h: Seleccionem el TCLK0 com a font de rellotge per als
   timers, preescalem amb un factor 1/8 els rellotges i configurem com a
   sortida els senyals MCLK_0 i MCLK_1 */
asm("  LDI 28E3h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CTR)");

/* Configurem el Serial Data Configuration Register */
/* Amb el valor 00c0h: Habilitem interrupcions de l'AMELIA2 */
asm("  LDI 00c0h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(SDC)");

/* Configurem el Amelia Control Register */
/* Amb el valor 00F6h: surt del reset, configurem com a Master, dividim
   el rellotge per 384, habilitem canals 2 i 3 i utilitzem com a rellotge
   del sistema M_CLOCK_0 */
asm("  LDI 00F6h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CMR)");

/* Configurem el Configuration Register */
/* El valor 8010h es fixe */
asm("  LDI 8010h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CFR)");
```

```
/* Configurem el Enhanced Configuration Register */
/* Amb el valor 00A0h: la interrupció es per nivell i les interrupcions
   es resetejen per lectura del ISR o per lectura dels canals */

asm("  LDI 00A0h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(ECFR)");

/* Configurem el Enhanced Control Register */
/* Amb el valor 0001h l'AMELIA2 demana interrupció quan el buffer FIFO
   esta ple al 50 % */
asm("  LDI 0003h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(ECTR)");

/* Configurem el Interrupt Mask Register */
/* Amb el valor 2000h interromp quan el buffer esta ple al % especificat */
asm("  LDI 2000h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(IMR)");

/* Bucle d'espera per al cicle de calibració */
asm("  LDI 0800h,R0");
asm("  LSH 13,R0");
asm("Calib_wait: SUBI 01h,R0");
asm("  BNZ Calib_wait");

/* Netegem la FIFO (Quatre lectures per canal) */
asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  LDI *+AR0(DR3),R0");

/* Netegem interrupcions pendents (llegint en el ISR, s'elimina la
   causa de la interrupció) */
asm("  LDI  *+AR0(ISR),R0");
}
/************************************************************************/
/*                  Fi Funció :  set_amd16qs                            */
/************************************************************************/

/************************************************************************/
/* Funció : dio32                                            */
/* Propòsit : Inicialilitza la dio32                         */
/* Retorna : res                                             */
/************************************************************************/
void dio32(void)
{
      /* configura la dio32 com a master i sense fer servir cap trigger */

/*    asm("  LDI 0241h,R1"); */
      asm("  LDI 0200h,R1");
/*    asm("  LDI 023Ch,R1"); */
```

```
       asm("   LSH 16,R1");
       asm("   LDI @CONTROL,AR1");
       asm("   STI R1,*AR1");

       /* configura els 32 bits com a sortida  */

/*     asm("  LDI 1111h,R1"); */ /* Configuració DIO 4 ports de sortida */
       asm("  LDI 0001h,R1");    /* Configuració DIO A=sortida B=C=D=entrada */
       asm("   LSH 16,R1");
       asm("   LDI @PORTREG,AR1");
       asm("   STI R1,*AR1");

       /* s'envia un estat de desconexió de tot el pont */
       asm("  LDI 00ffh,R1");
       asm("   LSH 16,R1");
       asm("   LDI @posicio_DIO,AR1");
       asm("   STI R1,*AR1");
}
/*************************************************************************/
/*                  Fi Funció : dio32                                    */
/*************************************************************************/
```