APPENDIXES.

A.1 - Programs.

In this section four programs are being listed:

- TH1_5c.c. This program executes the classical DTC. It is sent to the DSP and executed by it. Figure 4.3 in chapter 4 shows in detail its main functions.
- PCTH1_5c.c. The PC executes this program. It sends the th1_5c.c file to the DSP in order to perform the classical DTC. All data from the th1_5c.c is stored in the PC.
- TH1_9.c. This program executes the Fuzzy Logic DTC. It is sent to the DSP and executed by it. Figure 4.4 in chapter 4 shows in detail its main functions.
- PCTH1_9.c. This program is executed by the PC. It sends the th1_9.c file to the DSP in order to perform the Fuzzy Logic DTC. All data from the th1_9.c is stored in the PC. Also, the PC helps in the execution of the Fuzzy Logic controllers.

A.1.1 - TH1_5c.c.

```
/* File : TH1_5C.C   */
/* Description: DSP program that performs:
1/program the adquisition boards AM/D16QS + AM/D16SA
2/run the flux and torque estimator each 166us saving the data in the DPRAM
3/ Classical DTC. ATN table.
/*_____________________________________________________________________________*/

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

/* DPRAM Positions */
#define posicio_memoria_FLAG 0x0c00009   /*Flag*/
#define posicio_memoria_mfs_hst 0x0c0000b  /* DTC references values */
#define posicio_memoria_te_hst 0x0c0000c   /*         "             */
#define posicio_memoria_mfs_set 0x0c0000d  /*         "             */
#define posicio_memoria_te_set 0x0c0000e   /*         "             */
#define posicio_memoria_STOP 0x0c00012   /*Stop bit*/

#define posicio_memoria_f0  0x0c00048
#define posicio_memoria_f1  0x0c00049
#define posicio_memoria_f2  0x0c0004a
#define posicio_memoria_f3  0x0c0004b
#define posicio_memoria_f4  0x0c0004c
#define posicio_memoria_f5  0x0c0004d
#define posicio_memoria_f6  0x0c0004e
#define posicio_memoria_f7  0x0c0004f
#define posicio_memoria_f8  0x0c00050
#define posicio_memoria_f9  0x0c00051
/*10*/
#define posicio_memoria_fa  0x0c00052
#define posicio_memoria_fb  0x0c00053
#define posicio_memoria_fc  0x0c00054
#define posicio_memoria_fd  0x0c00055
#define posicio_memoria_fe  0x0c00056
#define posicio_memoria_ff  0x0c00057
#define posicio_memoria_f10  0x0c00058
#define posicio_memoria_f11  0x0c00059
#define posicio_memoria_f12  0x0c0005a
#define posicio_memoria_f13  0x0c0005b
/*20*/
#define posicio_memoria_f14  0x0c0005c
#define posicio_memoria_f15  0x0c0005d
#define posicio_memoria_f16  0x0c0005e
#define posicio_memoria_f17  0x0c0005f
#define posicio_memoria_f18  0x0c00060
#define posicio_memoria_f19  0x0c00061
#define posicio_memoria_f1a  0x0c00062
#define posicio_memoria_f1b  0x0c00063
#define posicio_memoria_f1c  0x0c00064
#define posicio_memoria_f1d  0x0c00065
```

```
/*30*/
#define posicio_memoria_f1e  0x0c00066
#define posicio_memoria_f1f  0x0c00067
#define posicio_memoria_f20  0x0c00068
#define posicio_memoria_f21  0x0c00069
#define posicio_memoria_f22  0x0c0006a
#define posicio_memoria_f23  0x0c0006b
#define posicio_memoria_f24  0x0c0006c
#define posicio_memoria_f25  0x0c0006d
#define posicio_memoria_f26  0x0c0006e
#define posicio_memoria_f27  0x0c0006f
/*40*/
#define posicio_memoria_f28  0x0c00070
#define posicio_memoria_f29  0x0c00071
#define posicio_memoria_f2a  0x0c00072
#define posicio_memoria_f2b  0x0c00073
#define posicio_memoria_f2c  0x0c00074
#define posicio_memoria_f2d  0x0c00075
#define posicio_memoria_f2e  0x0c00076
#define posicio_memoria_f2f  0x0c00077

#define PI 3.141592654


asm(" .data");

/* DIO32 config. register  */

asm("COUNTDPRAM  .long 0c00016h");

/* 4 analog channels addresses*/

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

/* #define PORTREG 00828007 */
asm("PORTREG      .long 00828007h");
/* DIO32 control register address */
/* #define  CONTROL 00828005  */
asm("CONTROL      .long 00828005h");

asm("dio1    .long 00ff0000h");
asm("pointerdio  .long dio1");


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 0081A000h");
```

```
asm("ITTP         .word 06000000h");

asm("Mascara    .word   0000ffffh");

/* Map AMELIA2 values / registers */

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");
/* Map AMELIA2 registers */

asm("TIMER1       .word 81B005h");
asm("UCR          .word 81B008h");
asm("ACR          .word 81B00Ah");
asm("CONFIG       .word 81B00Fh");
asm("IMR1         .word 81B00Bh");
asm("DCR1         .word 81B00Ch");
asm("CH0          .word 81B002h");
asm("CH1          .word 81B006h");

/* Constants */
asm("IMR1_DEF           .word 000010000h");
asm("DCR1_DEF           .word 0001e0000h");
asm("TIMER1_DEF         .word 0fd9b0000h");
asm("UCR_DEF            .word 0a4000000h");
asm("ACR_DEF            .word 000f20000h");
asm("CONFIG_DEF         .word 08dff0000h");


/*Program section called 'int0X' where the interruprion vector is placed */
/*c_int0X is the name of the rotuine address */

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


asm("    .sect \".int02\"");
asm("    .word _c_int02");


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

```
/* End int section */

asm("    .text");

/*  pointers  */
long * pointer_FLAG=(long *)posicio_memoria_FLAG;
float * pointer_mfs_set=(float *)posicio_memoria_mfs_set;
float * pointer_te_set=(float *)posicio_memoria_te_set;
float * pointer_mfs_hst=(float *)posicio_memoria_mfs_hst;
float * pointer_te_hst=(float *)posicio_memoria_te_hst;
long * pointerSTOP=(long *)posicio_memoria_STOP;

float * pointerf0=(float *)posicio_memoria_f0;
float * pointerf1=(float *)posicio_memoria_f1;
float * pointerf2=(float *)posicio_memoria_f2;
float * pointerf3=(float *)posicio_memoria_f3;
float * pointerf4=(float *)posicio_memoria_f4;
float * pointerf5=(float *)posicio_memoria_f5;
float * pointerf6=(float *)posicio_memoria_f6;
float * pointerf7=(float *)posicio_memoria_f7;
float * pointerf8=(float *)posicio_memoria_f8;
float * pointerf9=(float *)posicio_memoria_f9;
/*10*/
float * pointerfa=(float *)posicio_memoria_fa;
float * pointerfb=(float *)posicio_memoria_fb;
float * pointerfc=(float *)posicio_memoria_fc;
float * pointerfd=(float *)posicio_memoria_fd;
float * pointerfe=(float *)posicio_memoria_fe;
float * pointerff=(float *)posicio_memoria_ff;
float * pointerf10=(float *)posicio_memoria_f10;
float * pointerf11=(float *)posicio_memoria_f11;
float * pointerf12=(float *)posicio_memoria_f12;
float * pointerf13=(float *)posicio_memoria_f13;
/*20*/
float * pointerf14=(float *)posicio_memoria_f14;
float * pointerf15=(float *)posicio_memoria_f15;
float * pointerf16=(float *)posicio_memoria_f16;
float * pointerf17=(float *)posicio_memoria_f17;
float * pointerf18=(float *)posicio_memoria_f18;
float * pointerf19=(float *)posicio_memoria_f19;
float * pointerf1a=(float *)posicio_memoria_f1a;
float * pointerf1b=(float *)posicio_memoria_f1b;
float * pointerf1c=(float *)posicio_memoria_f1c;
float * pointerf1d=(float *)posicio_memoria_f1d;
/*30*/
float * pointerf1e=(float *)posicio_memoria_f1e;
float * pointerf1f=(float *)posicio_memoria_f1f;
float * pointerf20=(float *)posicio_memoria_f20;
float * pointerf21=(float *)posicio_memoria_f21;
float * pointerf22=(float *)posicio_memoria_f22;
float * pointerf23=(float *)posicio_memoria_f23;
float * pointerf24=(float *)posicio_memoria_f24;
float * pointerf25=(float *)posicio_memoria_f25;
float * pointerf26=(float *)posicio_memoria_f26;
float * pointerf27=(float *)posicio_memoria_f27;
```

```
/*40*/
float * pointerf28=(float *)posicio_memoria_f28;
float * pointerf29=(float *)posicio_memoria_f29;
float * pointerf2a=(float *)posicio_memoria_f2a;
float * pointerf2b=(float *)posicio_memoria_f2b;
float * pointerf2c=(float *)posicio_memoria_f2c;
float * pointerf2d=(float *)posicio_memoria_f2d;
float * pointerf2e=(float *)posicio_memoria_f2e;
float * pointerf2f=(float *)posicio_memoria_f2f;

int d=0,STOP=0,FLAG=0;
/*DTC varialbes*/
int e1=0x95,e2=0x95,e7=0x95,e0=0xaa;
int e1_b=0;
int e_fi_ti=0xaa,e_fd_ti=0xaa,e_fi_td=0xaa,e_fd_td=0xaa;
int e_fd_td_b=0,e_fi_ti_b=0,e_fd_ti_b=0,e_fi_td_b=0;

int te_e=0,mfs_e=0;
float te_plim=0,te_nlim=0,mfs_plim=0,mfs_nlim=0,te_hst=0,mfs_hst=0;
float mfs_set=0,te_set=0;

/* adquisition variables*/
long int ch0=0,ch1=0,ch2=0;
long int *pch0=&ch0,*pch1=&ch1,*pch2=&ch2;
float ch0_float=0,ch1_float=1,ch2_float=0;

/* estimator zoh variables */
float isD=0,isQ=0,frD=0,frQ=0,fsD=0,fsQ=0,mfs=0,te=0,ang=0,sect=0;
float frD_1=0,frQ_1=0,isD_1=0,isQ_1=0;
float wrrds=0,wrrds_1=0,wrrds_2=0,wrrds_3=0,wrrds_4=0;
float sectbis_aux=0,angbisfloat=0;
int sectbis=0;
float etz2_5=0.9960652825,etz1_6=0.9973751317;
float Rr=5.05,Lm=0.30563,P=2,Lr=0.32023;
float Lx=0.0091376,c=0.816496,stz=0;
int x=0;
double angbis=0,table[402];

/* functions  */
void c_int01(void);     /* Amelia int */
void c_int02(void);     /* Amelia int */
void c_int05(void);     /* Control */
void set_amd16qs(void); /* ADC inizialization  */
void set_amd16sa(void); /* ADC inizialization  */
void dio32(void);       /* DIO inizialization  */
void atn_table (void);  /* ATN table */
void get_ADC(void);     /* read 2 currents + wm */
void flux_torque_est_zoh(void); /* Flux and torque estimation zoh
*/
void dtc(void);                 /* dtc */
void save_data_DPRAM(void);
/*void save_data_DPRAM_2(void);*/
void send_active_state(void);   /* send the active state */
void save_active_state(void);   /* save the active state */
void bit1(void);
void bit0(void);
```

```
/************************************************************************/
/* Function : main
*/
/************************************************************************/
void main(void)
{

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");

      dio32();

/* null state to VSI */

      do {
            STOP=*pointerSTOP;
            }
      while (STOP==1);

if (STOP ==0)
{   atn_table();
      /* getting the reference values */
      mfs_set=*pointer_mfs_set;
      te_set=*pointer_te_set;
      mfs_hst=*pointer_mfs_hst;
      te_hst=*pointer_te_hst;
      mfs_plim=mfs_set+mfs_hst;
      mfs_nlim=mfs_set-mfs_hst;
      te_plim=te_set+te_hst;
      te_nlim=te_set-te_hst;

      set_amd16qs();
      set_amd16sa();

      asm("  LDI @ITTP,IF");

/* Enalbing interruptions INT0 +INT1 + INT5 of PC/C32 board */
      asm("  LDI  0013h,IE");
/* Globals int enable */
      asm("   OR 2000h,ST");
/* Start Cache */
      asm("  OR 0800h,ST");

}
```

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

/* Disable int */

      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");

}
/************************************************************************/
/*                End Function : main                                   */
/************************************************************************/

/************************************************************************/
/* Function : c_int01                                                   */
/*            AMELIA2_A interruption                                   */
/************************************************************************/
void c_int01(void)
{
asm("  PUSH AR0");
asm("  PUSH AR2");
asm("  PUSH R0");

/* disable int*/
asm(" ANDN 2000h,ST");

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

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  ASH -16,R0");
asm("  LDI @_pch2,AR2");
```

```
asm("  STI R0,*AR2");
asm("  LDI *+AR0(DR3),R0");

/* int 05 is set*/
asm(" OR 010h,IF");

asm("  POP R0");
asm("  POP AR2");
asm("  POP AR0");
/* erase this interrupt flag */
asm(" ANDN 1h,IF");

}
/*************************************************************************/
/*               End Function : c_int01                                  */
/*************************************************************************/

/*************************************************************************/
/* Function : c_int02                                                    */
/*            AMELIA2_B int                                              */
/*************************************************************************/
void c_int02(void)
{
asm("  PUSH AR0");
asm("  PUSH AR1");
asm("  PUSH AR2");
asm("  PUSH R0");
asm("  PUSH R1");
/* disable int*/
asm(" ANDN 2000h,ST");

asm("  LDI @IMR1,AR0");
asm("  LDI *AR0,R0");

asm("  LDI @CH0,AR0");
asm("  LDI @CH1,AR1");
asm("  LDI *AR0,R0");
asm("  LDI *AR1,R1");
asm("  ASH -16,R0");
asm("  LDI @_pch0,AR2");
asm("  STI R0,*AR2");
asm("  ASH -16,R1");
asm("  LDI @_pch1,AR2");
asm("  STI R1,*AR2");

asm("  POP R1");
asm("  POP R0");
asm("  POP AR2");
asm("  POP AR1");
asm("  POP AR0");

/* erase this interrupt flag */
asm(" ANDN 2h,IF");

}
/*************************************************************************/
```

```
/*                              End Function : c_int02              */
/*********************************************************************/

/*********************************************************************/
/* Funcio : c_int05 (166us)
/*********************************************************************/
void c_int05(void)
{
      bit1();
      get_ADC();
      flux_torque_est_zoh();
      dtc();
/*    bit0();*/
/*    for(d=0;d<215;d++)
      {
      }
      d=0;*/
      send_active_state();
      bit0();
      save_data_DPRAM();
/*    save_data_DPRAM_2();*/

}
/*********************************************************************/
/*               End Function : c_int05 (166us)
          */
/*********************************************************************/

/*********************************************************************/
/* Function : get_ADC (estimator)
/*********************************************************************/
void get_ADC (void)
{
      /* disable int*/
      asm(" ANDN 2000h,ST");

      /* get isA,isB,wm */

      /* 65535 / 6volts ; 1volt / 7.0922 amp */
/*    ch0_float=-(1.0*ch0*9.155273438e-5*7.1*1.03*6.11/7)+3.5e-3;
      ch1_float=-(1.0*ch1*9.155273438e-5*7.1*1.015*6.11/7)-3.0e-2;*/

      /* 65535 / 6volts ; 1volt / 10 amp */
      /* LEM probe */
      ch0_float=-(1.0*ch0*9.155273438e-5*10*4.85/5.15);
      /* Tektronic Probe */
      ch1_float=-(1.0*ch1*9.155273438e-5*10);

      /* 65535 / 4volts ; 6e-2volts / 1rpm  */
      ch2_float=-(1.0*(ch2-170.5)*6.103515625e-5/(1.10965*6e-4));

      /* enalbe global int*/
      asm(" OR 2000h,ST");
}
/*********************************************************************/
/*               Funcio : get_ADC (estimator)                        */
```

```
/*************************************************************************/
/*************************************************************************/
/* Function : flux_torque_est_zoh                                        */
/*************************************************************************/
  void flux_torque_est_zoh(void)
{   double sqr=0;
      float aux=0;
      int index=0;

      frD_1=frD;
      frQ_1=frQ;
      wrrds_4=wrrds_3;
      wrrds_3=wrrds_2;
      wrrds_2=wrrds_1;
      wrrds_1=wrrds;
      isD_1=isD;
      isQ_1=isQ;

      isD=1.2247*ch0_float;
/*    isQ=0.95*(1.4142135*ch1_float)+(0.7071067*ch0_float);*/
      isQ=(1.3435028*ch1_float)+(0.7071067*ch0_float);
/*    wrrds=(wrrds_4+wrrds_3+wrrds_2+wrrds_1+(2*PI/60*ch2_float))/5;*/
      wrrds=(wrrds_4+wrrds_3+wrrds_2+wrrds_1+(0.104719*ch2_float))/5;

/*frD=(((1-etz2_5)*((Lm*isD_1)-((Lr/Rr)*P*wrrds_1*frQ_1)))+(etz2_5*frD_1));
frQ=(((1-etz2_5)*((Lm*isQ_1)+((Lr/Rr)*P*wrrds_1*frD_1)))+(etz2_5*frQ_1));*/
/*frD=(((1-etz1_6)*((Lm*isD_1)-((Lr/Rr)*P*wrrds_1*frQ_1)))+(etz1_6*frD_1));
frQ=(((1-etz1_6)*((Lm*isQ_1)+((Lr/Rr)*P*wrrds_1*frD_1)))+(etz1_6*frQ_1));*/

frD=(0.80223847e-3*isD_1)-(0.332895663e-3*wrrds_1*frQ_1)+(etz1_6*frD_1);

frQ=(0.80223847e-3*isQ_1)+(0.332895663e-3*wrrds_1*frD_1)+(etz1_6*frQ_1);

/*    fsD=(isD*(Lx/Lm))+(frD*(Lm/Lr));
      fsQ=(isQ*(Lx/Lm))+(frQ*(Lm/Lr));*/
      fsD=(isD*0.0298975)+(frD*0.9544077);
      fsQ=(isQ*0.0298975)+(frQ*0.9544077);

/* may be it could be just used the square value */
      sqr=(fsD*fsD)+(fsQ*fsQ);
      mfs=sqrt(sqr);

      /* using the table */
      aux=fsQ/fsD;
      if (aux < 0)
            {aux=aux*(-1);
            }

      if( aux < 1 )
            {index=(int)(aux*100);
            angbis=table[index];
            angbisfloat=(float)angbis; }
      else if ( aux >= 100)
             {angbis=table[199];
              angbisfloat=(float)angbis; }
      else
```

```
            {index=(int)(aux+99);
             angbis=table[index];
             angbisfloat=(float)angbis; }
      if ( fsD > 0)
         {  if( fsQ < 0)
              {angbisfloat=(float)((-1)*angbis);}
         }
      else
            {
            if( fsQ > 0)
                  {angbisfloat=PI+((-1)*angbisfloat);}
            else
                  {angbisfloat=(angbisfloat-PI);}
            }
      if (-PI < angbisfloat ) sectbis=4;
      if ((-5*PI/6) < angbisfloat ) sectbis=5;
      if ((-PI/2) < angbisfloat) sectbis=6;
      if (-PI/6 < angbisfloat) sectbis=1;
      if (PI/6 < angbisfloat) sectbis=2;
      if (PI/2 < angbisfloat) sectbis=3;
      if ((5*PI/6) < angbisfloat) sectbis= 4;

/*    te=P*c*((fsD*isQ)-(fsQ*isD));*/
      te=1.632992*((fsD*isQ)-(fsQ*isD));
}
/*************************************************************************/
/*                      End function: flux_torque_est_zoh                */
/*************************************************************************/
/*************************************************************************/
/* Function:       dtc                                                   */
/*************************************************************************/

  void dtc (void)
{
      /* new reference values are read in savw_data_DPRAM
         before setting the FLAG=1 */

      if ( mfs > mfs_plim)
            mfs_e = 0;
      else if ( mfs < mfs_nlim)
            mfs_e = 1;

      if ( te > (te_plim))
            {te_e = 0;
            }
        /* te_e = 0; */
      else if ( te < te_nlim)
            {
            te_e = 2;

            }
      else
            {
            te_e = 1;

            }
```

```
switch(sectbis)
                  {
                  case 1:
                  e_fi_ti=0xa5;
                  e_fi_td=0x99;
                  e_fd_ti=0xa6;
                  e_fd_td=0x9a;
                  e_fi_ti_b=2;
                  e_fi_td_b=6;
                  e_fd_ti_b=3;
                  e_fd_td_b=5;
                  break;
                  case 2:
                  e_fi_ti=0xa6;
                  e_fi_td=0xa9;
                  e_fd_ti=0x96;
                  e_fd_td=0x99;
                  e_fi_ti_b=3;
                  e_fi_td_b=1;
                  e_fd_ti_b=4;
                  e_fd_td_b=6;
                  break;
                  case 3:
                  e_fi_ti=0x96;
                  e_fi_td=0xa5;
                  e_fd_ti=0x9a;
                  e_fd_td=0xa9;
                  e_fi_ti_b=4;
                  e_fi_td_b=2;
                  e_fd_ti_b=5;
                  e_fd_td_b=1;
                  break;
                  case 4:
                  e_fi_ti=0x9a;
                  e_fi_td=0xa6;
                  e_fd_ti=0x99;
                  e_fd_td=0xa5;
                  e_fi_ti_b=5;
                  e_fi_td_b=3;
                  e_fd_ti_b=6;
                  e_fd_td_b=2;
                  break;
                  case 5:
                  e_fi_ti=0x99;
                  e_fi_td=0x96;
                  e_fd_ti=0xa9;
                  e_fd_td=0xa6;
                  e_fi_ti_b=6;
                  e_fi_td_b=4;
                  e_fd_ti_b=1;
                  e_fd_td_b=3;
                  break;
                  case 6:
                  e_fi_ti=0xa9;
                  e_fi_td=0x9a;
```

```
                        e_fd_ti=0xa5;
                        e_fd_td=0x96;
                        e_fi_ti_b=1;
                        e_fi_td_b=5;
                        e_fd_ti_b=2;
                        e_fd_td_b=4;
                        break;
                        }
      switch(te_e)
                        {
                        case 0:
                        if ( mfs_e == 0 )
                              { e1 = e_fd_td;
                                e1_b = e_fd_td_b;  }
                        else
                              { e1 = e_fi_td;
                                e1_b = e_fi_td_b; }
                        break;
                        case 1:
                        if ( mfs_e == 0 )
                              { e1 = e0;
                              e1_b = 0; }
                        else
                              { e1 = e0;
                               e1_b = 0;}
                        break;
                        case 2:
                        if ( mfs_e == 0 )
                              { e1 = e_fd_ti;
                              e1_b = e_fd_ti_b; }
                        else
                              { e1 = e_fi_ti;
                               e1_b = e_fi_ti_b;}
                        break;
                        }

}
/*************************************************************************/
/*                            End function: dtc                          */
/*************************************************************************/
/*************************************************************************/
/* Function:              send_active_state                              */
/*************************************************************************/
  void send_active_state (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm("   LDI @_e1,R1");
      asm("   LSH 16,R1");
/* as long as save_active_state is not used */
/*    asm(" LDI @pointerdio,AR1");*/
/*    asm("   LDI *AR1,R1");*/
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");
```

```
      asm(" POP R1");
      asm(" POP AR1");
}
/*************************************************************************/
/*               End function: send_active_state                         */
/*************************************************************************/

/*************************************************************************/
/* Function : save_data_DPRAM */
/*************************************************************************/
void save_data_DPRAM (void)
{   float te_aux;

      te_aux=te+10;
      sectbis_aux=sectbis*(-1);
      e1_b=e1_b*(-1);

      *pointerf0=isD;
      *pointerf1=isQ;
      *pointerf2=te_aux;
/*    *pointerf3=te_aux;
      *pointerf4=11.2;
      *pointerf5=8.6;
      *pointerf6=te_e;
/*    *pointerf7=e1_b;
      *pointerf8=te;
/*    *pointerf9=te_e;

      /*10*/
/*    *pointerfa=wrrds_1;*/
/*    *pointerfb=te;
      *pointerfc=ang;
      *pointerfd=sect;
      *pointerfe=frD;
      *pointerff=frQ;
      *pointerf10=wrrds;
      *pointerf11=isDdef;
      *pointerf12=isQdef;
      *pointerf13=ange2;*/
      /*20*/

 /* new reference values are read in save_data_DPRAM
      before setting the FLAG=1 */
      mfs_set=*pointer_mfs_set;
      te_set=*pointer_te_set;
      mfs_hst=*pointer_mfs_hst;
      te_hst=*pointer_te_hst;
      mfs_plim=mfs_set+mfs_hst;
      mfs_nlim=mfs_set-mfs_hst;
      te_plim=te_set+te_hst;
      te_nlim=te_set-te_hst;


      FLAG=1;
      *pointer_FLAG = FLAG;
}
```

```
/*************************************************************************/
/*          End function: save_data_DPRAM                    */
/*************************************************************************/

/*************************************************************************/
/* Funcio : save_data_DPRAM_2
/*************************************************************************/

void save_data_DPRAM_2 (void)
{   float mfs_set_aux,mfs_aux,te_set_aux,te_aux,wrrds_aux;

      mfs_set_aux=mfs_set*(-1);
      mfs_aux=mfs*(-1);
      te_set_aux=te_set;
      te_aux=te;
      wrrds_aux=wrrds/(-10);

      *pointerf0=mfs_set_aux;
      *pointerf1=mfs_aux;
      *pointerf2=te_set_aux;
      *pointerf3=te_aux;
      *pointerf4=wrrds_aux;
 /* new reference values are read in save_data_DPRAM before setting the
FLAG=1 */
      mfs_set=*pointer_mfs_set;
      te_set=*pointer_te_set;
      mfs_hst=*pointer_mfs_hst;
      te_hst=*pointer_te_hst;
      mfs_plim=mfs_set+mfs_hst;
      mfs_nlim=mfs_set-mfs_hst;
      te_plim=te_set+te_hst;
      te_nlim=te_set-te_hst;


      FLAG=1;
      *pointer_FLAG = FLAG;

}
/*************************************************************************/
/*          End function: save_data_DPRAM_2                    */
/*************************************************************************/

/*************************************************************************/
/*                  save_active_state                    */
/*************************************************************************/
  void save_active_state (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI @pointerdio,AR1");
      asm("   LDI @_e1,R1");
      asm("   LSH 16,R1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
```

```
      asm(" POP AR1");
}
/*************************************************************************/
/*                            End:  save_active_state                    */
/*************************************************************************/

/*************************************************************************/
/*                            bit1                                       */
/*************************************************************************/
  void bit1 (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI 0000h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/*************************************************************************/
/*                            End function: bit1                         */
/*************************************************************************/
/*************************************************************************/
/*                                  bit0                                 */
/*************************************************************************/
  void bit0 (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI 00ffh,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");


}
/*************************************************************************/
/*                            End function: bit0                         */
/*************************************************************************/

/*************************************************************************/

/*************************************************************************/
/* Inizialization routines */
/*************************************************************************/

/*************************************************************************/
/* Function:  set_amd16qs                                                */
/*            It inizializates the board AM/D16QS.                       */
```

```
/*************************************************************************/
void set_amd16qs(void)
{
asm("  LDI @AMELIA0,AR0");

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

/* Reset AM/D16QS board */
asm("  LDI 0000h,R0");
asm("  STI R0,*+AR0(CMR)");

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

/* User Control register config.*/
/* Value 28E3h: TCLK0 clk. factor 1/8  MCLK_0 and MCLK_1 as an output */
/*asm("  LDI 28E0h,R0");*/ /* /1  48  32 */
asm("  LDI 28E1h,R0"); /* /2  24  16 */
/*asm("  LDI 28E2h,R0");*/ /* /4  12   8 */
/*asm("  LDI 28E3h,R0");*/   /* /8   6   4 */


asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CTR)");

/* Config. Serial Data Configuration Register */
/* 00c0h: Enable AMELIA2 */
asm("  LDI 00c0h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(SDC)");

/* Config. el Amelia Control Register */
/* Value 00F6h: out of reset, Master config., divide 384 */
/* enable  chanals 2 i 3 and M_CLOCK_0 */

/*asm("  LDI 00F6h,R0");  */     /* 32 16  8 4 */
asm("  LDI 00E6h,R0");        /* 48 24 12 6 */

asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CMR)");

/* Config. Configuration Register */
/* Value 8010h. It is cte */
asm("  LDI 8010h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CFR)");

/* Config. Enhanced Configuration Register */
/* 00A0h: level int. It is reset either reading ISR */
/* or reading the channels */

asm("  LDI 00A0h,R0");

asm("  LSH 16,R0");
asm("  STI R0,*+AR0(ECFR)");
```

```
/* Config. Enhanced Control Register */
/* Value 0003h. AMELIA2 int. when buffer FIFO is 100% full*/

/*asm("  LDI 0000h,R0");*/ /* 25% */
/*asm("  LDI 0001h,R0");*/
/*asm("  LDI 0002h,R0");*/
asm("  LDI 0003h,R0");     /* 100% */
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(ECTR)");

/* Config. Interrupt Mask Register */
/* Value 2000h int. when buffer if full at the specified %  */
asm("  LDI 2000h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(IMR)");


/* Wait loop calibration system */
/*asm("  LDI 0800h,R0");
asm("  LSH 13,R0");
asm("Calib_wait: SUBI 01h,R0");
asm("  BNZ Calib_wait");*/
asm("  LDI 0300h,R0");
asm("  LSH 15,R0");
asm("Calib_wait: SUBI 01h,R0");
asm("  BNZ Calib_wait");

/* Clean the FIFO (4 read for channel) */
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");

/* Clean any int.  (reading ISR) */
asm("  LDI  *+AR0(ISR),R0");
}
/**************************************************************************/
/*          End function :  set_amd16qs                                 */
/**************************************************************************/
```

```
/*************************************************************************/
/* Function:  set_amd16sa                                                */
/*            It inizializates the board AM/D16QS.                        */
/*************************************************************************/
void set_amd16sa(void)
{
asm("  LDI @DCR1,AR0");
asm("  LDI @DCR1_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @UCR,AR0");
asm("  LDI @UCR_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @ACR,AR0");
asm("  LDI @ACR_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @TIMER1,AR0");
asm("  LDI @TIMER1_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @CONFIG,AR0");
asm("  LDI @CONFIG_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @IMR1,AR0");
asm("  LDI @IMR1_DEF,R0");
asm("  STI R0,*AR0");

}
/*************************************************************************/
/*          End function :  set_amd16sa                                  */
/*************************************************************************/

/*************************************************************************/
/* Funcio : dio32                                                        */
/*          Ini. dio32                                                   */
/*************************************************************************/
void dio32(void)
{
      /* Dio32 as a master without any trigger */

/*    asm(" LDI 0241h,R1");*/
      asm(" LDI 023Ch,R1");
      asm("   LSH 16,R1");
      asm("   LDI @CONTROL,AR1");
      asm("   STI R1,*AR1");

      /* 32 bits as output  */

      asm(" LDI 1111h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @PORTREG,AR1");
      asm("   STI R1,*AR1");
```

```
      /* disconnect the bridge */
      asm(" LDI 00ffh,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");

      asm(" LDI 0000h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

}
/*************************************************************************/
/*          End function :  dio32                                        */
/*************************************************************************/

/*************************************************************************/
/* Funcio : atn_table                                                    */
/*          Creates a table with ATN  values                             */
/*************************************************************************/
void atn_table(void)
{
      double s=0,r=0;
      int k=0;

      for(k=0;k<=100;k++)
            {
            r=atan(s);
            table[k]=r;
            s=s+0.01;
            }
      for(k=100;k<=200;k++)
            {
            s=k-99;
            r=atan(s);
            table[k]=r;
            }
  }

/*************************************************************************/
/*                            End function :  atn_table                  */
/*************************************************************************/
```

A.1.2 - PCTH1_5c.c

```
/*************************************************************************/
/*    PCTH1_5C.C               Classical DTC                             */
/*************************************************************************/

/* Needed C files        */
#include <dos.h>
#include <math.h>
#include <conio.h>

/* Needed C files  for the DSP */
#include "c:\mon32\bc45\tic32.h"
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

/* Memory DPRAM positions for the set points + exchange values */
#define  FLAG                0x0c00009    // FLAG
#define  HST_MFS        0xc0000bl    // Flux set point_hst margin
#define  HST_TE         0xc0000cl    // Torque set point_hst margin
#define  MFS                 0xc0000dl    // Flux set point
#define  TE                  0xc0000el    // Torque set point
#define  STOP           0xc00012l    // SVM stop bit

#define  f0             0xc00048l
#define  f1             0xc00049l
#define  f2             0xc0004al
#define  f3             0xc0004bl
#define  f4             0xc0004cl
#define  f5             0xc0004dl
#define  f6             0xc0004el
#define  f7             0xc0004fl
#define  f8             0xc00050l
#define  f9             0xc00051l
/*10*/
#define  fa             0xc00052l
#define  fb             0xc00053l
#define  fc             0xc00054l
#define  fd             0xc00055l
#define  fe             0xc00056l
#define  ff             0xc00057l
#define  f10            0xc00058l
#define  f11            0xc00059l
#define  f12            0xc0005al
#define  f13            0xc0005bl
/*20*/
#define  f14            0xc0005cl
#define  f15            0xc0005dl

#define  PI 3.141592654

/* DSP file  */
#define  FILENAME "TH1_5C.OUT"
```

```
/* Global variables */

char x=0;
int s=0,j=0,k=0,i=0;
long int t=0 ;

/* Global adquisition variables */
float v0,v1,v2,v3,v4,v5,v6,v7,v8,v9,va,vb;
float vc,vd,ve,vf,v10,v11,v12,v13,v14,v15;
unsigned long stop=0,flag=0;                // Stop + flag  bit variable



/* set points */

float mfs_set=0,te_set=0,te_hst=0,mfs_hst=0;

double mfs_sug=0,te_set_d=0;
/* pointer file */

FILE *f_grf;
/* data matrix */
/*float dt [10][901];*/
float dt [9][901];
/* Functions definition */
void IniciDSP(void);
void EngegaDSP(void);
void AturaDSP(void);
void ReadDPRAM(void);
void Savedata(void);
void Setpoints(void);
void Setpoints1(void);
void Data2file(void);
void main (void)
{
clrscr();
printf("\nPress '1' to start");
x = getch();
      if (x=='1')
            {
            IniciDSP();
            delay(500);
            Setpoints();
            delay (200);
            EngegaDSP();
            delay (4000);
       /*   Setpoints1();
            delay (5000);
            Setpoints1();
            delay (5000);*/
            Setpoints1();
            delay (5000);
            printf("\nSaving data");
            for(t=0; t<1000000 ; t++)
                  {
```

```
                        for (s=0; s<5 ; s++)
                        {
                        }
                  s=0;
                  Get_DPRAM_Word_32(FLAG,&flag);
                  if (flag == 1 )
                        {
                              if ( j <= 900 )
                                    {
                                    ReadDPRAM();
                                    Savedata();
                                    j++;
                                    }
                        flag=0;
                        Put_DPRAM_Word_32(FLAG,flag);
                        }
                  }
            }
AturaDSP();
Data2file();
}


/*************************************************************** */
/*                 DSP  inizialization                          */
/***************************************************************/
void IniciDSP(void)
{
/* Seleccio data type for the DSP */
Set_Processor_Data_Type_Size_32();

/* DSP Inicialitzation. Board adress  (0x290), DPRMA adress (0xD00)*/
/*  bus ISA size  */
Select_Board(0x290, 0xd00, 8);

/* Enable DPRAM  */
Enable_DPRAM();

/* DSP reset  */
Global_Reset();

/* Load the file to be executed  */
Load_Object_File(FILENAME);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End of DSP inizialization                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/***************************************************************/
/*                Start up DSP                                  */
/***************************************************************/
void EngegaDSP(void)
{     stop=0;
      Put_DPRAM_Word_32(STOP, stop);
```

```
      /* Run file  */
      Run_From(Get_Entry_PC());
      delay(2000);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End start up DSP                                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Setpoints                                      */
/*****************************************************************/
void Setpoints(void)
{

      printf("\n\nTorque setpoint N/m: ");
      scanf("%f", &te_set);
      Put_DPRAM_Float_32(TE,te_set);
      printf("Torque set point hst margin: ");
      scanf("%f", &te_hst);
      Put_DPRAM_Float_32(HST_TE,te_hst);

      te_set_d=te_set;
      mfs_sug=0.3665*sqrt(te_set_d);
      printf("\nThe suggested optimized flux square value is %lf",
mfs_sug);
      printf("\nFlux set point: ");
      scanf("%f", &mfs_set);
      Put_DPRAM_Float_32(MFS,mfs_set);
      printf("Flux set point hst margin: ");
      scanf("%f", &mfs_hst);
      Put_DPRAM_Float_32(HST_MFS,mfs_hst);
/* old version */
/*    printf("\nFlux set point: ");
      scanf("%f", &mfs_set);
      Put_DPRAM_Float_32(MFS,mfs_set);
      printf("Flux set point hst margin: ");
      scanf("%f", &mfs_hst);
      Put_DPRAM_Float_32(HST_MFS,mfs_hst);
      printf("Torque setpoint N/m: ");
      scanf("%f", &te_set);
      Put_DPRAM_Float_32(TE,te_set);
      printf("Torque set point hst margin: ");
      scanf("%f", &te_hst);
      Put_DPRAM_Float_32(HST_TE,te_hst);*/

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End setpoints                                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Setpoints1                                     */
/*****************************************************************/
void Setpoints1(void)
{
```

```
      printf("\nTorque setpoint N/m: ");
      scanf("%f", &te_set);
      Put_DPRAM_Float_32(TE,te_set);
      te_set_d=te_set;
      /* because squared value */
      mfs_sug=0.3665*sqrt(te_set_d);
      printf("\nThe suggested optimized flux value is %lf", mfs_sug);
      printf("\nFlux set point: ");
      scanf("%f", &mfs_set);
      Put_DPRAM_Float_32(MFS,mfs_set);

/*old version */
/*    printf("\nFlux set point: ");
      scanf("%f", &mfs_set);
      Put_DPRAM_Float_32(MFS,mfs_set);
      printf("Torque setpoint N/m: ");
      scanf("%f", &te_set);
      Put_DPRAM_Float_32(TE,te_set);*/

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End setpoints1                                         */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Read DPRAM                                     */
/*****************************************************************/

void ReadDPRAM(void)
{
Get_DPRAM_Float_32(f0,&v0);
Get_DPRAM_Float_32(f1,&v1);
Get_DPRAM_Float_32(f2,&v2);
/*Get_DPRAM_Float_32(f3,&v3);
Get_DPRAM_Float_32(f4,&v4);
Get_DPRAM_Float_32(f5,&v5);
Get_DPRAM_Float_32(f6,&v6);
/*Get_DPRAM_Float_32(f7,&v7);
Get_DPRAM_Float_32(f8,&v8);
/*Get_DPRAM_Float_32(f9,&v9);
/*10*/
/*Get_DPRAM_Float_32(fa,&va);
Get_DPRAM_Float_32(fb,&vb);
Get_DPRAM_Float_32(fc,&vc);
Get_DPRAM_Float_32(fd,&vd);
Get_DPRAM_Float_32(fe,&ve);
Get_DPRAM_Float_32(ff,&vf);
Get_DPRAM_Float_32(f10,&v10);
Get_DPRAM_Float_32(f11,&v11);
Get_DPRAM_Float_32(f12,&v12);
Get_DPRAM_Float_32(f13,&v13);  */
/*20*/
/*Get_DPRAM_Float_32(f14,&v14);
Get_DPRAM_Float_32(f15,&v15);*/
```

```
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End read DPRAM                                            */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Savedata                                       */
/*****************************************************************/

void Savedata(void)
{
dt[0][j]=v0;
dt[1][j]=v1;
dt[2][j]=v2;
/*dt[3][j]=v3;
dt[4][j]=v4;
dt[5][j]=v5;
dt[6][j]=v6;
/*dt[7][j]=v7;
dt[8][j]=v8;
/*dt[9][j]=v9;
/*10*/
/*dt[10][j]=va;
dt[11][j]=vb;
dt[12][j]=vc;
dt[13][j]=vd;
dt[14][j]=ve;
dt[15][j]=vf;
dt[16][j]=v10;
dt[17][j]=v11;
dt[18][j]=v12;
dt[19][j]=v13;*/
/*20*/
/*dt[20][j]=v14;
dt[21][j]=v15;*/
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End save data                                             */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Stop DSP                                       */
/*****************************************************************/
void AturaDSP(void)
{     stop=1;
      Put_DPRAM_Word_32(STOP, stop);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End stop DSP                                              */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                        Data2file                                */
/*****************************************************************/
void Data2file(void)
{
```

```
if ((f_grf = fopen("\\DTC2.GRF", "w")) == 0)
      {
      printf("Unable to open file");
      return;
      }
else
      {
      for(k=0;k<j;k++)
            {
/*          fprintf(f_grf, "%e %e %e %e %e %e %e
\n",dt[0][k],dt[1][k],dt[2][k],dt[3][k],dt[4][k],dt[5][k],dt[6][k]);*/
            fprintf(f_grf, "%e %e %e \n",dt[0][k],dt[1][k],dt[2][k]);
            }
      fclose(f_grf);
}
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End Data2file */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

A.1.3 - TH1_9.c

```
/* File : TH1_9.C   */
/* Description: FLDTC */
/*____________________________*/
/* arc tangent function */


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


/* DPRAM Positions */
#define posicio_memoria_FLAG       0x0c00009 /*Flag*/
#define posicio_memoria_COUNTER    0x0c0000a
#define posicio_memoria_mfs_hst    0x0c0000b /*Flux set point_hst margin*/
#define posicio_memoria_te_hst     0x0c0000c /*Electr. torque set point_hst
margin*/
#define posicio_memoria_mfs_set    0x0c0000d /*Flux set point*/
#define posicio_memoria_te_set     0x0c0000e /*Electr. torque set point*/
#define posicio_memoria_STOP       0x0c00012 /*Stop bit*/
/* Start obtaining the DC */
#define posicio_memoria_START      0x0c00013 /*Start bit*/
/* Errors in obtaining the DC */
#define posicio_memoria_ERROR            0x0c00014 /*Number of errors*/
/* FLC1 references values */
#define posicio_memoria_FLUX_ANGLE_I  0x0c00015 /*Flux angle*/
#define posicio_memoria_TE_ERR_I  0x0c00016 /*Torque error i+1*/
#define posicio_memoria_WM             0x0c00017 /*Motor speed */
#define posicio_memoria_FI         0x0c00018 /*FI=0 flux decrease */
                                        /*FI=1 increase */
/* FLC2 refernces values */
#define posicio_memoria_TE_ERR_I_1  0x0c00019 /*Torque error i*/
#define posicio_memoria_COUNT1      0x0c0001a /*Increment Torque error
i+1*/

#define posicio_memoria_DC_AUX      0x0c0001d /*Last duty cycle*/
#define posicio_memoria_INC_DC      0x0c0001e /*FLC2 set point*/
#define posicio_memoria_EQ         0x0c0001f /*Equal signal*/

/* duty cycle */
#define posicio_memoria_DCFLC1     0x0c0001b  /*Duty cycle from FLC1*/
#define posicio_memoria_DCFLC2     0x0c0001c  /*Duty cycle from FLC2*/

#define posicio_memoria_f0  0x0c00048
#define posicio_memoria_f1  0x0c00049
#define posicio_memoria_f2  0x0c0004a
#define posicio_memoria_f3  0x0c0004b
#define posicio_memoria_f4  0x0c0004c
#define posicio_memoria_f5  0x0c0004d
#define posicio_memoria_f6  0x0c0004e
#define posicio_memoria_f7  0x0c0004f
#define posicio_memoria_f8  0x0c00050
#define posicio_memoria_f9  0x0c00051
```

```
/*10*/
#define posicio_memoria_fa  0x0c00052
#define posicio_memoria_fb  0x0c00053
#define posicio_memoria_fc  0x0c00054
#define posicio_memoria_fd  0x0c00055
#define posicio_memoria_fe  0x0c00056
#define posicio_memoria_ff  0x0c00057
#define posicio_memoria_f10  0x0c00058
#define posicio_memoria_f11  0x0c00059
#define posicio_memoria_f12  0x0c0005a
#define posicio_memoria_f13  0x0c0005b
/*20*/
#define posicio_memoria_f14  0x0c0005c
#define posicio_memoria_f15  0x0c0005d
#define posicio_memoria_f16  0x0c0005e
#define posicio_memoria_f17  0x0c0005f
#define posicio_memoria_f18  0x0c00060
#define posicio_memoria_f19  0x0c00061
#define posicio_memoria_f1a  0x0c00062
#define posicio_memoria_f1b  0x0c00063
#define posicio_memoria_f1c  0x0c00064
#define posicio_memoria_f1d  0x0c00065
/*30*/
#define posicio_memoria_f1e  0x0c00066
#define posicio_memoria_f1f  0x0c00067
#define posicio_memoria_f20  0x0c00068
#define posicio_memoria_f21  0x0c00069
#define posicio_memoria_f22  0x0c0006a
#define posicio_memoria_f23  0x0c0006b
#define posicio_memoria_f24  0x0c0006c
#define posicio_memoria_f25  0x0c0006d
#define posicio_memoria_f26  0x0c0006e
#define posicio_memoria_f27  0x0c0006f
/*40*/
#define posicio_memoria_f28  0x0c00070
#define posicio_memoria_f29  0x0c00071
#define posicio_memoria_f2a  0x0c00072
#define posicio_memoria_f2b  0x0c00073
#define posicio_memoria_f2c  0x0c00074
#define posicio_memoria_f2d  0x0c00075
#define posicio_memoria_f2e  0x0c00076
#define posicio_memoria_f2f  0x0c00077

#define PI 3.141592654

asm(" .data");

/* Direccio registre de control del timer 1  */
```

```
asm("TIMECTRL     .long 00808030h");

/* #define  RSTCTRL   00000601 */
asm("RSTCTRL      .set 00000601h");

/* Paraula de control que engega el timer 1  */
/* #define  SETCTRL  000006c1 */
asm("SETCTRL      .set 000006c1h");

/* Timer 1 register adresses */
asm("COUNTER      .long 00808034h");
asm("PERIOD .long 00808038h");

/* Direccio registre de configuracio de la DIO32  */

asm("COUNTDPRAM  .long 0c00016h");

/* Defineixo les adreces dels quatre canals anal•gics*/

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

/* #define PORTREG 00828007 */
asm("PORTREG      .long 00828007h");
/* Direccio registre de control de la DIO32  */
/* #define  CONTROL 00828005  */
asm("CONTROL      .long 00828005h");

asm("dio1    .long 00ff0000h");
asm("c1    .long 450");
asm("pointerdio  .long dio1");


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 0081A000h");

asm("ITTP         .word 06000000h");

asm("Mascara    .word   0000ffffh");

/* Map Registers 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");

/* Map Register AMELIA2 */

asm("TIMER1       .word 81B005h");
asm("UCR          .word 81B008h");
asm("ACR          .word 81B00Ah");
asm("CONFIG       .word 81B00Fh");
asm("IMR1         .word 81B00Bh");
asm("DCR1         .word 81B00Ch");
asm("CH0          .word 81B002h");
asm("CH1          .word 81B006h");

/* Constants */
asm("IMR1_DEF           .word 000010000h");
asm("DCR1_DEF           .word 0001e0000h");
asm("TIMER1_DEF         .word 0fd9b0000h");
asm("UCR_DEF            .word 0a4000000h");
asm("ACR_DEF            .word 000f20000h");
asm("CONFIG_DEF         .word 08dff0000h");

/* Program section called 'int0X' where the interruprion vector is placed*/
/* c_int0X is the name of the rotuine address */

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

asm("    .sect \".int02\"");
asm("    .word _c_int02");

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

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

/* End int label section */

asm("    .text");

/*  pointers  */
long * pointer_FLAG=(long *)posicio_memoria_FLAG;
long * pointer_COUNTER=(long *)posicio_memoria_COUNTER;
long * pointer_ERROR=(long *)posicio_memoria_ERROR;
float * pointer_mfs_set=(float *)posicio_memoria_mfs_set;
float * pointer_te_set=(float *)posicio_memoria_te_set;
```

```
float * pointer_mfs_hst=(float *)posicio_memoria_mfs_hst;
float * pointer_te_hst=(float *)posicio_memoria_te_hst;
long * pointerSTOP=(long *)posicio_memoria_STOP;

long * pointerSTART=(long *)posicio_memoria_START;
float * pointer_flux_angle_i=(float *)posicio_memoria_FLUX_ANGLE_I;
float * pointer_te_err_i=(float *)posicio_memoria_TE_ERR_I;
float * pointer_wm=(float *)posicio_memoria_WM;
long * pointerFI=(long *)posicio_memoria_FI;

long * pointer_count1=(long *)posicio_memoria_COUNT1;
float * pointer_dc_aux=(float *)posicio_memoria_DC_AUX;
float * pointer_inc_dc=(float *)posicio_memoria_INC_DC;
long * pointerEQ=(long *)posicio_memoria_EQ;

long * pointerDCFLC1=(long *)posicio_memoria_DCFLC1;
long * pointerDCFLC2=(long *)posicio_memoria_DCFLC2;

float * pointerf0=(float *)posicio_memoria_f0;
float * pointerf1=(float *)posicio_memoria_f1;
float * pointerf2=(float *)posicio_memoria_f2;
float * pointerf3=(float *)posicio_memoria_f3;
float * pointerf4=(float *)posicio_memoria_f4;
float * pointerf5=(float *)posicio_memoria_f5;
float * pointerf6=(float *)posicio_memoria_f6;
float * pointerf7=(float *)posicio_memoria_f7;
float * pointerf8=(float *)posicio_memoria_f8;
float * pointerf9=(float *)posicio_memoria_f9;
/*10*/
float * pointerfa=(float *)posicio_memoria_fa;
float * pointerfb=(float *)posicio_memoria_fb;
float * pointerfc=(float *)posicio_memoria_fc;
float * pointerfd=(float *)posicio_memoria_fd;
float * pointerfe=(float *)posicio_memoria_fe;
float * pointerff=(float *)posicio_memoria_ff;
float * pointerf10=(float *)posicio_memoria_f10;
float * pointerf11=(float *)posicio_memoria_f11;
float * pointerf12=(float *)posicio_memoria_f12;
float * pointerf13=(float *)posicio_memoria_f13;
/*20*/
float * pointerf14=(float *)posicio_memoria_f14;
float * pointerf15=(float *)posicio_memoria_f15;
float * pointerf16=(float *)posicio_memoria_f16;
float * pointerf17=(float *)posicio_memoria_f17;
float * pointerf18=(float *)posicio_memoria_f18;
float * pointerf19=(float *)posicio_memoria_f19;
float * pointerf1a=(float *)posicio_memoria_f1a;
float * pointerf1b=(float *)posicio_memoria_f1b;
float * pointerf1c=(float *)posicio_memoria_f1c;
float * pointerf1d=(float *)posicio_memoria_f1d;
/*30*/
float * pointerf1e=(float *)posicio_memoria_f1e;
float * pointerf1f=(float *)posicio_memoria_f1f;
float * pointerf20=(float *)posicio_memoria_f20;
float * pointerf21=(float *)posicio_memoria_f21;
float * pointerf22=(float *)posicio_memoria_f22;
```

```
float * pointerf23=(float *)posicio_memoria_f23;
float * pointerf24=(float *)posicio_memoria_f24;
float * pointerf25=(float *)posicio_memoria_f25;
float * pointerf26=(float *)posicio_memoria_f26;
float * pointerf27=(float *)posicio_memoria_f27;
/*40*/
float * pointerf28=(float *)posicio_memoria_f28;
float * pointerf29=(float *)posicio_memoria_f29;
float * pointerf2a=(float *)posicio_memoria_f2a;
float * pointerf2b=(float *)posicio_memoria_f2b;
float * pointerf2c=(float *)posicio_memoria_f2c;
float * pointerf2d=(float *)posicio_memoria_f2d;
float * pointerf2e=(float *)posicio_memoria_f2e;
float * pointerf2f=(float *)posicio_memoria_f2f;

int d=0,STOP=0,FLAG=0,s=0;

/* aux*/
int in1,in2,count1,index1,index2,index3;
float fin3,Tpc,Wpc;

/* DTC variables */
int e1=0x95,e2=0x95,e7=0x95,e0=0xbf;
int e1_b=0;
int e_fi_ti=0xaa,e_fd_ti=0xaa,e_fi_td=0xaa,e_fd_td=0xaa;
int e_fi_ti_b=0,e_fd_ti_b=0,e_fi_td_b=0,e_fd_td_b=0;

int te_e=0,mfs_e=0;
float te_plim=0,te_nlim=0,mfs_plim=0,mfs_nlim=0,te_hst=0,mfs_hst=0;
float mfs_set=0,te_set=0;

/* FLDTC variables */
int COUNTER_AUX=0,COUNTER=0,DC2=0,DC=0,ERROR=0,ERROR_FLAG=0,START=0;
int e1_b_1=0;
int EQ_aux=0,EQ=0;
float dc_aux=1,inc_dc=0;

/*FLC1 variables*/
int DCFLC1=0,FI=0;
float te_err_i=0,wm=0;

/* FLC2 variables */
int DCFLC2=0;
float te_err_i_1=0,inc_te_i=0;
float te_1=0;

/* adquisition variables*/
long int ch0=0,ch1=0,ch2=0;
long int *pch0=&ch0,*pch1=&ch1,*pch2=&ch2;
float ch0_float=0,ch1_float=1,ch2_float=0;

/* estimator zoh variables */
float isD=0,isQ=0,frD=0,frQ=0,fsD=0,fsQ=0,mfs=0,te=0,ang=0,sect=0;
float frD_1=0,frQ_1=0,isD_1=0,isQ_1=0;
float wrrds=0,wrrds_1=0,wrrds_2=0,wrrds_3=0,wrrds_4=0;
float sectbis_aux=0,angbisfloat=0;
```

```
int sectbis=0;
float etz1_25=0.99803070219,etz1_6=0.9973751317,etz2_5=0.9960652825;
float Rr=5.05,Lm=0.30563,P=2,Lr=0.32023;
float Lx=0.0091376,c=0.816496,stz=0;

double angbis=0,table[402];

/* Functions  */
void c_int01(void);     /* Amelia int */
void c_int02(void);     /* Amelia int */
void c_int05(void);     /* Control */
void c_int10(void);     /* Timer */

void set_amd16qs(void); /* ADC inizialization  */
void set_amd16sa(void); /* ADC inizialization  */
void dio32(void);       /* DIO inizialization  */
void set_c32_timer1(void);      /* TIMER 1 inizialization */
void atn_table (void);  /* ATN table */
void get_ADC(void);     /* read 2 currents + wm */
void flux_torque_est_zoh(void); /* Flux and torque estimation zoh */
void dtc(void);                 /* dtc */
void save_data_DPRAM(void);     /* saving data in DPRAM */
void save_data_DPRAM_2(void);   /* saving data in DPRAM_2 */
void send_active_state(void);   /* send the active state */
void save_active_state(void);   /* save the active state */
void program_timer1(void);     /* Timer 1 programing*/
void get_DC(void);             /* Get Duty cycle */
void bit1(void);
void bit0(void);

/************************************************************************/
/* Funcio : main                                                        */
/************************************************************************/
void main(void)
{

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");
/* null state to VSI */
      dio32();

      do {
            STOP=*pointerSTOP;
            }
      while (STOP==1);
```

```
if (STOP == 0)
{   atn_table();
      /* getting the reference values */
      mfs_set=*pointer_mfs_set;
      te_set=*pointer_te_set;
      Tpc=te_set*100/9.8;
      mfs_hst=*pointer_mfs_hst;
      te_hst=*pointer_te_hst;
      mfs_plim=mfs_set+mfs_hst;
      mfs_nlim=mfs_set-mfs_hst;
      te_plim=te_set+te_hst;
      te_nlim=te_set-te_hst;
      /* ini ADC */
      set_amd16qs();
      set_amd16sa();

      asm("  LDI @ITTP,IF");
/* Enalbing interruptions INT0 +INT1 + INT5 of PC/C32 board */
      asm("  LDI  0013h,IE");
/* Globals int enable */
      asm("   OR 2000h,ST");
/* Start Cache */
      asm("  OR 0800h,ST");

}
      do {
            STOP=*pointerSTOP;
            for(s=0;s<20;s++)
            {
            }
            s=0;
            }
      while (STOP==0);
/* dis int */
      asm("    AND 0000h,IE");
      asm("    AND 0000h,ST");
/* disconnect VSI */
      asm(" LDI 00ffh,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");
      *pointer_COUNTER=COUNTER;
      *pointer_ERROR=ERROR;

}
/*************************************************************************/
/*                            End function : main                        */
/*************************************************************************/

/*************************************************************************/
/* Funcio : c_int01                                                      */
/*          Amelia2 A int*/
/*************************************************************************/
void c_int01(void)
{
asm("  PUSH AR0");
```

```
asm("  PUSH AR2");
asm("  PUSH R0");

/* disable int*/
asm(" ANDN 2000h,ST");

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

asm("  LDI *+AR0(DR0),R0");
asm("  LDI *+AR0(DR1),R0");
asm("  LDI *+AR0(DR2),R0");
asm("  ASH -16,R0");
asm("  LDI @_pch2,AR2");
asm("  STI R0,*AR2");
asm("  LDI *+AR0(DR3),R0");

/* int 05 is set*/
asm(" OR 010h,IF");

asm("  POP R0");
asm("  POP AR2");
asm("  POP AR0");
/* erase this interrupt flag */
asm(" ANDN 1h,IF");

}
/*************************************************************************/
/*                            End Function : c_int01                     */
/*************************************************************************/

/*************************************************************************/
/* Funcio : c_int02                                                      */
/*          AMELIA2 B int            */
/*************************************************************************/
void c_int02(void)
{
asm("  PUSH AR0");
asm("  PUSH AR1");
asm("  PUSH AR2");
asm("  PUSH R0");
```

```
asm("  PUSH R1");
/* disable int*/
asm(" ANDN 2000h,ST");

asm("  LDI @IMR1,AR0");
asm("  LDI *AR0,R0");

asm("  LDI @CH0,AR0");
asm("  LDI @CH1,AR1");
asm("  LDI *AR0,R0");
asm("  LDI *AR1,R1");
asm("  ASH -16,R0");
asm("  LDI @_pch0,AR2");
asm("  STI R0,*AR2");
asm("  ASH -16,R1");
asm("  LDI @_pch1,AR2");
asm("  STI R1,*AR2");

asm("  POP R1");
asm("  POP R0");
asm("  POP AR2");
asm("  POP AR1");
asm("  POP AR0");

/* erase this interrupt flag */
asm(" ANDN 2h,IF");

}
/*************************************************************************/
/*                              End function : c_int02                   */
/*************************************************************************/


/*************************************************************************/
/* Funcio : c_int05 (166 us)                                             */
/*************************************************************************/
void c_int05 (void)
{ /*  *pointer_FLAG = 0;*/
      bit1();
      get_ADC();
      flux_torque_est_zoh();
      dtc();
      send_active_state();
      bit0();
      flc_set_points();
      *pointerSTART=1;
/*    save_data_DPRAM();*/
      save_data_DPRAM_2();
      for(d=0;d<150;d++)
            {
            }
      d=0;
      get_DC();
      program_timer1();
/*    DC=0;
      asm(" LDI @PERIOD,AR0");
```

```
      asm(" LDI @_DC,R0");
      asm(" STI R0,*AR0");
      asm(" LDI @TIMECTRL,AR0");
      asm(" LDI SETCTRL,R0");
      asm(" STI R0,*AR0");
      asm(" OR 200h,IE");*/


/*    bit1();  */
/*    save_data_DPRAM();*/

}
/************************************************************************/
/*               End c_int05 (166us)                                    */
/************************************************************************/

/************************************************************************/
/* Funcio : get_ADC (estimator)
/* Retorna : res  */
/************************************************************************/
void get_ADC (void)
{
      /* disable int*/
      asm(" ANDN 2000h,ST");

      /* get isA,isB,wm */

      /* 65535 / 6volts ; 1volt / 7.0922 amp */
/*    ch0_float=-(1.0*ch0*9.155273438e-5*7.1*1.03*6.11/7)+3.5e-3;
      ch1_float=-(1.0*ch1*9.155273438e-5*7.1*1.015*6.11/7)-3.0e-2;*/

      /* 65535 / 6volts ; 1volt / 10 amp */
      /* LEM probe */
      ch0_float=-(1.0*ch0*9.155273438e-5*10*4.85/5.15);
      /* Tektronic Probe */
      ch1_float=-(1.0*ch1*9.155273438e-5*10);

      /* 65535 / 4volts ; 6e-2volts / 1rpm  */
      ch2_float=-(1.0*(ch2-170.5)*6.103515625e-5/(1.10965*6e-4));

      /* enalbe global int*/
      asm(" OR 2000h,ST");
}
/************************************************************************/
/*               End function : get_ADC (estimator)                     */
/************************************************************************/
/************************************************************************/
/*                flux_torque_est_zoh                                   */
/************************************************************************/
  void flux_torque_est_zoh(void)
{   /* using the square value */
      double sqr=0;
      float aux=0;
      int index=0;

      frD_1=frD;
```

```
      frQ_1=frQ;
      wrrds_4=wrrds_3;
      wrrds_3=wrrds_2;
      wrrds_2=wrrds_1;
      wrrds_1=wrrds;
      isD_1=isD;
      isQ_1=isQ;

      te_1=te;

      isD=1.2247*ch0_float;
/*    isQ=0.95*(1.4142135*ch1_float)+(0.7071067*ch0_float);*/
      isQ=(1.3435028*ch1_float)+(0.7071067*ch0_float);
/*    wrrds=(wrrds_4+wrrds_3+wrrds_2+wrrds_1+(2*PI/60*ch2_float))/5;*/
      wrrds=(wrrds_4+wrrds_3+wrrds_2+wrrds_1+(0.104719*ch2_float))/5;

/*frD=(((1-etz2_5)*((Lm*isD_1)-((Lr/Rr)*P*wrrds_1*frQ_1)))+(etz2_5*frD_1));
frQ=(((1-etz2_5)*((Lm*isQ_1)+((Lr/Rr)*P*wrrds_1*frD_1)))+(etz2_5*frQ_1));*/
/*frD=(1.20256771e-3*isD_1)-(0.499015431e-3*wrrds_1*frQ_1)+(etz2_5*frD_1);
frQ=(1.20256771e-3*isQ_1)+(0.499015431e-3*wrrds_1*frD_1)+(etz2_5*frQ_1);*/
/*frD=(((1-etz1_6)*((Lm*isD_1)-((Lr/Rr)*P*wrrds_1*frQ_1)))+(etz1_6*frD_1));
frQ=(((1-etz1_6)*((Lm*isQ_1)+((Lr/Rr)*P*wrrds_1*frD_1)))+(etz1_6*frQ_1));*/

frD=(0.80223847e-3*isD_1)-(0.332895663e-3*wrrds_1*frQ_1)+(etz1_6*frD_1);
frQ=(0.80223847e-3*isQ_1)+(0.332895663e-3*wrrds_1*frD_1)+(etz1_6*frQ_1);

/*

frD=(((1-etz1_25)*((Lm*isD_1)-((Lr/Rr)*P*wrrds_1*frQ_1)))+(etz1_25*frD_1));

frQ=(((1-etz1_25)*((Lm*isQ_1)+((Lr/Rr)*P*wrrds_1*frD_1)))+(etz1_25*frQ_1));
*/



/*    fsD=(isD*(Lx/Lm))+(frD*(Lm/Lr));
      fsQ=(isQ*(Lx/Lm))+(frQ*(Lm/Lr));*/
      fsD=(isD*0.0298975)+(frD*0.9544077);
      fsQ=(isQ*0.0298975)+(frQ*0.9544077);

/* may be it could be just used the square value */
      sqr=(fsD*fsD)+(fsQ*fsQ);
      mfs=sqrt(sqr);
/* using the square value */
/*    mfs=(fsD*fsD)+(fsQ*fsQ);*/

      /* using the table */
      aux=fsQ/fsD;
      if (aux < 0)
            {aux=aux*(-1);
            }

      if( aux < 1 )
            {index=(int)(aux*100);
            angbis=table[index];
            angbisfloat=(float)angbis; }
```

```
      else if ( aux >= 100)
             {angbis=table[199];
              angbisfloat=(float)angbis; }
      else
            {index=(int)(aux+99);
             angbis=table[index];
             angbisfloat=(float)angbis; }

      if ( fsD > 0)
         {  if( fsQ < 0)
              {angbisfloat=(float)((-1)*angbis);}
         }
      else
            {
            if( fsQ > 0)
                  {angbisfloat=PI+((-1)*angbisfloat);}
            else
                  {angbisfloat=(angbisfloat-PI);}
            }

      if (-PI < angbisfloat ) sectbis=4;
      if ((-5*PI/6) < angbisfloat ) sectbis=5;
      if ((-PI/2) < angbisfloat) sectbis=6;
      if (-PI/6 < angbisfloat) sectbis=1;
      if (PI/6 < angbisfloat) sectbis=2;
      if (PI/2 < angbisfloat) sectbis=3;
      if ((5*PI/6) < angbisfloat) sectbis= 4;

/*    te=P*c*((fsD*isQ)-(fsQ*isD));*/
      te=1.632992*((fsD*isQ)-(fsQ*isD));
}
/*****************************************************************************/
/*                      End function: flux_torque_est_zoh                    */
/*****************************************************************************/
/*****************************************************************************/
/*                     dtc                                                   */
/*****************************************************************************/

  void dtc (void)
{   /* needed for FLDTC */
      e1_b_1=e1_b;

      if ( mfs > mfs_plim)
            mfs_e = 0;
      else if ( mfs < mfs_nlim)
            mfs_e = 1;

      if ( te > (te_plim))
            te_e = 0;
      else
            te_e = 2;


      switch(sectbis)
                        {
                        case 1:
```

```
e_fi_ti=0xa5;
e_fi_td=0x99;
e_fd_ti=0xa6;
e_fd_td=0x9a;
e_fi_ti_b=2;
e_fi_td_b=6;
e_fd_ti_b=3;
e_fd_td_b=5;
break;
case 2:
e_fi_ti=0xa6;
e_fi_td=0xa9;
e_fd_ti=0x96;
e_fd_td=0x99;
e_fi_ti_b=3;
e_fi_td_b=1;
e_fd_ti_b=4;
e_fd_td_b=6;
break;
case 3:
e_fi_ti=0x96;
e_fi_td=0xa5;
e_fd_ti=0x9a;
e_fd_td=0xa9;
e_fi_ti_b=4;
e_fi_td_b=2;
e_fd_ti_b=5;
e_fd_td_b=1;
break;
case 4:
e_fi_ti=0x9a;
e_fi_td=0xa6;
e_fd_ti=0x99;
e_fd_td=0xa5;
e_fi_ti_b=5;
e_fi_td_b=3;
e_fd_ti_b=6;
e_fd_td_b=2;
break;
case 5:
e_fi_ti=0x99;
e_fi_td=0x96;
e_fd_ti=0xa9;
e_fd_td=0xa6;
e_fi_ti_b=6;
e_fi_td_b=4;
e_fd_ti_b=1;
e_fd_td_b=3;
break;
case 6:
e_fi_ti=0xa9;
e_fi_td=0x9a;
e_fd_ti=0xa5;
e_fd_td=0x96;
e_fi_ti_b=1;
e_fi_td_b=5;
```

```
                        e_fd_ti_b=2;
                        e_fd_td_b=4;
                        break;
                        }
      switch(te_e)
                        {
                        case 0:
                        if ( mfs_e == 0 )
                              { e1 = e_fd_td;
                                e1_b = e_fd_td_b;  }
                        else
                              { e1 = e_fi_td;
                                e1_b = e_fi_td_b; }
                        break;
                        case 1:
                        if ( mfs_e == 0 )
                              { e1 = e0;
                              e1_b = 0; }
                        else
                              { e1 = e0;
                               e1_b = 0;}
                        break;
                        case 2:
                        if ( mfs_e == 0 )
                              { e1 = e_fd_ti;
                              e1_b = e_fd_ti_b; }
                        else
                              { e1 = e_fi_ti;
                               e1_b = e_fi_ti_b;}
                        break;
                        }
      /* needed for FLDTC */
      if (e1_b == e1_b_1)
            EQ=1;
      else
            EQ=0;
}
/*************************************************************************/
/*                       End function: dtc                               */
/*************************************************************************/
/*************************************************************************/
/*    flc_set_points                                                     */
/*************************************************************************/
  void flc_set_points (void)
/*  MULTIPLY 0.683 FOR MOTOR 2 */
/*  MULTIPLY 0.5 FOR 250us */
/*  MULTIPLY 0.7 FOR 166us */

{     if (EQ == 0)
            {
            /* FLC1 */
            te_err_i=(te_set-te)*0.31;
            if (te_err_i > 1.4)
                  te_err_i = 1.4;
            if (te_err_i < -1.4)
                  te_err_i = -1.4;
```

```
            index1=((te_err_i+0.1)/0.2)+7;
            /* rd 2 degress */
            /*    fin2=(fin2*180/PI)+180+30;*/
            angbisfloat=(angbisfloat*57.29)+210;
            in2=angbisfloat/60;
            angbisfloat=angbisfloat-60*in2;
            if (angbisfloat > 60)
                  angbisfloat = 60;
            if (angbisfloat < 0)
                  angbisfloat = 0;
            index2= (angbisfloat+3)/6;
      /*  Wpc=wrrds*100/146; */

            Wpc=wrrds*0.685;
            if (Tpc >= Wpc)
                  {fin3=Tpc+Wpc+Wpc;
                  in2=mfs_e+10;
                  *pointerFI=in2;
                  }
            else
                  {fin3=Tpc+Tpc+Wpc;
                  in2=mfs_e;
                  *pointerFI=in2;
                  }
            if (fin3 > 280)
                  fin3 = 280;
            if (fin3 < 100)
                  fin3 = 100;
/*          index3=(fin3-100+7.5)/15;*/
            index3=(fin3-92.5)/15;
/*          count1=(11*13*index1)+(13*index2)+(index3);*/
            count1=(143*index1)+(13*index2)+(index3);
            *pointer_count1=count1;
            *pointerEQ=0;
            }
            else
            {
            /* FLC2 */
            te_err_i_1=(te_set-te_1)*0.31;
            inc_te_i=(te-te_1)*0.31;
/*          te_err_i_1=0.3;
            inc_te_i=-0.5;*/
            if ( inc_te_i > 0.7 )
                  inc_te_i = 0.7;
            if (inc_te_i < -0.7)
                  inc_te_i = -0.7;
            in1=(inc_te_i+0.025)/0.05+14;

            if (te_err_i_1 > 0.7)
                  te_err_i_1 = 0.7;
            if (te_err_i_1 < -0.7)
                  te_err_i_1 = -0.7;
            in2=(te_err_i_1+0.025)/0.05+14;
            count1=(29*in1)+in2;

            *pointer_count1=count1;
```

```
            *pointer_dc_aux=dc_aux;
            *pointerEQ=1;
            }
}
/*************************************************************************/
/*     End function: flc_set_points                    */
/*************************************************************************/

/*************************************************************************/
/* Function : get_DC   */
/*************************************************************************/

void get_DC(void)
{   COUNTER++;
      START=*pointerSTART;
      if (START==1)
      {
      ERROR=ERROR+1;
      ERROR_FLAG=1;
      }
      else
      {
      ERROR_FLAG=0;
            if (EQ == 0)
            {
                  DC=*pointerDCFLC1;
                  DC2=DC;
            }
            else
            {
                  DC=*pointerDCFLC2;
                  DC2=DC;
               /* inc_dc=*pointer_inc_dc;*/
            }

      }

}
/*************************************************************************/
/*               End get_DC   */
/*************************************************************************/


/*************************************************************************/
/* Function : program_timer1   */
/*************************************************************************/

void program_timer1(void)
{
      /* 1ct=80ns=80e-9s. */
      /* 125us=1562.5ct*/
      /* 10us=8 % 125*/
      /* dc=1 => all active state. 1562 ct.*/
      /* dc=0.92 => 1437ct  */
      /* dc=0.08 => 125ct */
      /* 3125 for 250us */
```

```
      /* 2083 fpr 166us */

      if ( DC > 2000  )
      {
      /*do not program the timer 1. Do not enable its interrupt*/
      /*stop the timer wherever it was*/
      asm(" LDI @TIMECTRL,AR0");
      asm(" LDI RSTCTRL,R0");
      asm(" STI R0,*AR0");
/*    send_active_state();*/
      dc_aux=1.0;
      }
      else if ( ERROR_FLAG == 1)
      {

      DC=600;
      /*program the timer*/
      asm(" LDI @PERIOD,AR0");
      asm(" LDI @_DC,R0");
      asm(" STI R0,*AR0");
      /*start the timer*/
      asm(" LDI @TIMECTRL,AR0");
      asm(" LDI SETCTRL,R0");
      asm(" STI R0,*AR0");
/*    send_active_state();*/
/*    dc_aux=(600+1000)/2083.0;*/
      dc_aux=0.768;
      /*Enable timer 1 interupt */
      asm(" OR 200h,IE");
      }
      else if ( DC< 1084)
      {
      /*do not program the timer 1. Do not enable its interrupt*/
      /*stop the timer wherever it was*/
      asm(" LDI @TIMECTRL,AR0");
      asm(" LDI RSTCTRL,R0");
      asm(" STI R0,*AR0");
      /* SEND just a null state*/
      asm("   LDI @_e0,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");
/*    dc_aux=1000/2083.0;   */
      dc_aux=0.48;

      }
      else
      {
      /* 70/166*2083 = 878 */
      DC=DC2-1082;
/*    DC=600;    */
      /*program the timer*/
      asm(" LDI @PERIOD,AR0");
      asm(" LDI @_DC,R0");
      asm(" STI R0,*AR0");
      /*start the timer*/
```

```
      asm(" LDI @TIMECTRL,AR0");
      asm(" LDI SETCTRL,R0");
      asm(" STI R0,*AR0");
/*    send_active_state();*/
/*    dc_aux=(DC+878)/2083.0;*/
      dc_aux=DC2/2083.0;
      /*Enable timer 1 interupt */
      asm(" OR 200h,IE");
      }
}
/*************************************************************************/
/*               End program_timer1                                      */
/*************************************************************************/
/*************************************************************************/
/*                         send_active_state                             */
/*************************************************************************/
  void send_active_state (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm("   LDI @_e1,R1");
      asm("   LSH 16,R1");
/* as long as save_active_state is not used */
/*    asm(" LDI @pointerdio,AR1");*/
/*    asm("   LDI *AR1,R1");*/
      asm("   LDI @posicio_DIO,AR1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/*************************************************************************/
/*                      End function: send_active_state                  */
/*************************************************************************/

/*************************************************************************/
/* Funcio : save_data_DPRAM (estimator) */
/* Retorna : res  */
/*************************************************************************/
void save_data_DPRAM (void)
{   float e1_b_aux,DC_b,te_aux;

      sectbis_aux=sectbis*(-1);
      e1_b_aux=e1_b*(-1);
      DC_b=DC2/2083.0;
      te_aux=te;


      *pointerf0=te;
      *pointerf1=te_e;
      *pointerf2=dc_aux;
      *pointerf3=e1_b_aux;
      *pointerf4=sectbis_aux;
      *pointerf5=mfs;
      *pointerf6=mfs_e;
```

```
      *pointerf7=EQ;
/*    *pointerf8=;
      *pointerf9=;*/
      /*10*/
/*    *pointerfa=11;
      *pointerf0=e1_b;
      *pointerf1=sectbis_aux;
      *pointerf2=te_e;
      *pointerf3=mfs_e;
      *pointerf4=te_set;
      *pointerf5=fsD;
      *pointerf6=fsQ;
      *pointerf7=mfs;
      *pointerf8=te;
      *pointerf9=mfs_set;*/
      /*10*/
/*    *pointerfa=wrrds_1;  */
/*    *pointerfb=te;
      *pointerfc=ang;
      *pointerfd=sect;
      *pointerfe=frD;
      *pointerff=frQ;
      *pointerf10=wrrds;
      *pointerf11=isDdef;
      *pointerf12=isQdef;
      *pointerf13=ange2;*/
      /*20*/
/*    *pointer_FLAG = 1;*/
}
/*************************************************************************/
/*          End function: save_data_DPRAM                    */
/*************************************************************************/


/*************************************************************************/
/* Funcio : save_data_DPRAM_2 (estimator) */
/* Retorna : res  */
/*************************************************************************/

void save_data_DPRAM_2 (void)
{   float mfs_set_aux,mfs_aux,te_set_aux,te_aux,wrrds_aux;

      mfs_set_aux=mfs_set*(-1);
      mfs_aux=mfs*(-1);
      te_set_aux=te_set;
      te_aux=te;
      wrrds_aux=wrrds/(-10);

      *pointerf0=mfs_set_aux;
      *pointerf1=mfs_aux;
      *pointerf2=te_set_aux;
      *pointerf3=te_aux;
      *pointerf4=wrrds_aux;

}
/*************************************************************************/
```

```
/*          End function: save_data_DPRAM_2                              */
/***************************************************************************/
/***************************************************************************/
/*                              bit1                                       */
/***************************************************************************/
  void bit1 (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI 0000h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/***************************************************************************/
/*                End function: bit1                                       */
/***************************************************************************/
/***************************************************************************/
/*          bit0                                                           */
/***************************************************************************/
  void bit0 (void)
{
      asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI 00ffh,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/***************************************************************************/
/*                End function: bit0                                       */
/***************************************************************************/

/***************************************************************************/
/* Function : c_int10 (timer 1 interruption) */
/***************************************************************************/
  void c_int10(void)
{     /* disable int*/
      asm(" ANDN 2000h,ST");
      /* clean the int */
      asm(" ANDN 200h,IF");
      asm(" PUSH AR1");
      asm(" PUSH R1");

      /* stop the timer1 */
      asm(" LDI @TIMECTRL,AR1");
      asm(" LDI RSTCTRL,R1");
      asm(" STI R1,*AR1");
```

```
      /* send the null state */
      asm(" LDI @_e0,R1");
      asm(" LSH 16,R1");
      asm(" LDI @posicio_DIO,AR1");
      asm(" STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/*************************************************************************/
/*               End c_int10 (timer 1 interruption)                      */
/*************************************************************************/

/*************************************************************************/
/*    save_active_state                                                  */
/*************************************************************************/
  void save_active_state (void)
{   asm(" PUSH AR1");
      asm(" PUSH R1");

      asm(" LDI @pointerdio,AR1");
      asm("   LDI @_e1,R1");
      asm("   LSH 16,R1");
      asm("   STI R1,*AR1");

      asm(" POP R1");
      asm(" POP AR1");
}
/*************************************************************************/
/*                End:  save_active_state                                */
/*************************************************************************/
/*************************************************************************/
/* Inizialization tasks. Exectuted just once.                            */
/*************************************************************************/



 /*************************************************************************/
/* Function:  set_amd16qs                                                */
/*            It inizializates the board AM/D16QS.                       */
/*************************************************************************/
void set_amd16qs(void)
{
asm("  LDI @AMELIA0,AR0");

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

/* reset AM/D16QS board*/
asm("  LDI 0000h,R0");
asm("  STI R0,*+AR0(CMR)");

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

```
/* User Control register config.*/
/* Value 28E3h: TCLK0 clk. factor 1/8  MCLK_0 and MCLK_1 as an output */

/*asm("  LDI 28E0h,R0");*/ /* /1  48  32 */
asm("  LDI 28E1h,R0"); /* /2  24  16 */
/*asm("  LDI 28E2h,R0");*/ /* /4  12   8 */
/*asm("  LDI 28E3h,R0");*/   /* /8   6   4 */


asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CTR)");
/* Config. Serial Data Configuration Register */
/* 00c0h: Enable AMELIA2 */
asm("  LDI 00c0h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(SDC)");

/* Config. el Amelia Control Register */
/* Value 00F6h: out of reset, Master config., divide 384 */
/* enable  chanals 2 i 3 and M_CLOCK_0 */

/*asm("  LDI 00F6h,R0");*/       /* 32 16  8 4 */
asm("  LDI 00E6h,R0");        /* 48 24 12 6 */

asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CMR)");
/* Config. Configuration Register */
/* Value 8010h. It is cte */
asm("  LDI 8010h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(CFR)");

/* Config. Enhanced Configuration Register */
/* 00A0h: level int. It is reset either reading ISR */
/* or reading the channels */

asm("  LDI 00A0h,R0");

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

/* Config. Enhanced Control Register */
/* Value 0003h. AMELIA2 int. when buffer FIFO is 100% full*/

/*asm("  LDI 0000h,R0");*/ /* 25% */
/*asm("  LDI 0001h,R0");*/
/*asm("  LDI 0002h,R0");*/
asm("  LDI 0003h,R0");     /* 100% */
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(ECTR)");

/* Config. Interrupt Mask Register */
/* Value 2000h int. when buffer if full at the specified %  */
asm("  LDI 2000h,R0");
asm("  LSH 16,R0");
asm("  STI R0,*+AR0(IMR)");
```

```
/* Wait loop calibration system */
/*asm("  LDI 0800h,R0");
asm("  LSH 13,R0");
asm("Calib_wait: SUBI 01h,R0");
asm("  BNZ Calib_wait");*/
asm("  LDI 0300h,R0");
asm("  LSH 15,R0");
asm("Calib_wait: SUBI 01h,R0");
asm("  BNZ Calib_wait");
/* Clean the FIFO (4 read for channel) */
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");
/* Clean any int.  (reading ISR) */
asm("  LDI  *+AR0(ISR),R0");
}
/************************************************************************/
/*          End function :  set_amd16qs                                 */
/************************************************************************/


/************************************************************************/
/* Function:  set_amd16sa                                               */
/*            It inizializates the board AM/D16QS.                      */
/************************************************************************/
void set_amd16sa(void)
{
asm("  LDI @DCR1,AR0");
asm("  LDI @DCR1_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @UCR,AR0");
asm("  LDI @UCR_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @ACR,AR0");
asm("  LDI @ACR_DEF,R0");
asm("  STI R0,*AR0");
```

```
asm("  LDI @TIMER1,AR0");
asm("  LDI @TIMER1_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @CONFIG,AR0");
asm("  LDI @CONFIG_DEF,R0");
asm("  STI R0,*AR0");

asm("  LDI @IMR1,AR0");
asm("  LDI @IMR1_DEF,R0");
asm("  STI R0,*AR0");

}
/************************************************************************/
/*          End function :  set_amd16sa                                 */
/************************************************************************/

/************************************************************************/
/* Funcio : dio32                                                       */
/*          Inizializates the DIO 32                                    */
/************************************************************************/
void dio32(void)
{

      /* Dio32 as a master without any trigger */

/*    asm(" LDI 0241h,R1");*/
      asm(" LDI 023Ch,R1");
      asm("   LSH 16,R1");
      asm("   LDI @CONTROL,AR1");
      asm("   STI R1,*AR1");

      /* 32 bits as output  */
      asm(" LDI 1111h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @PORTREG,AR1");
      asm("   STI R1,*AR1");

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

      asm(" LDI 0000h,R1");
      asm("   LSH 16,R1");
      asm("   LDI @posicio_DIO2,AR1");
      asm("   STI R1,*AR1");

}

/************************************************************************/
/*          End function :  dio_32                                      */
/************************************************************************/
```

```
/*************************************************************************/
/* Funcio : atn_table                                                    */
/*          Creates a table with ATN  values                             */
/*************************************************************************/
void atn_table(void)
{
      double s=0,r=0;
      int k=0;

      for(k=0;k<=100;k++)
            {
            r=atan(s);
            table[k]=r;
            s=s+0.01;
            }
      for(k=100;k<=200;k++)
            {
            s=k-99;
            r=atan(s);
            table[k]=r;
            }
  }

/*************************************************************************/
/*                            End function :  atn_table                  */
/*************************************************************************/
```

A.1.4 - PCTH1_9.c

```
/*****************************************************************/
/*    PCTH1_9:  FLDTC 166us */
/*****************************************************************/

/* Needed C files        */
#include <dos.h>
#include <math.h>
#include <conio.h>
#include "c:\mon32\bc45\tic32.h"
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
/* Memory DPRAM positions for the set points + exchange values */
#define  FLAG           0x0c00009    // FLAG
#define  COUNTER        0x0c0000a
#define  HST_MFS        0xc0000bl    // Flux set point_hst margin
#define  HST_TE         0xc0000cl    // Torque set point_hst margin
#define  MFS            0xc0000dl    // Flux set point
#define  TE             0xc0000el    // Torque set point
#define  STOP           0xc00012l    // SVM stop bit
/* Start obtaining the DC */
#define  START          0xc00013l    // SVM stop bit
/* Errors in obtaining the DC */
#define  ERROR          0xc00014l    // SVM stop bit
/* FLC1 references values */
#define FLUX_ANGLE_I    0x0c00015 /*Flux angle*/
#define TE_ERR_I        0x0c00016 /*Torque error i+1*/
#define WM              0x0c00017 /*Motor speed */
#define FI              0x0c00018 /*FI=0 flux decrease */
                                  /*FI=1 increase */
/* FLC2 refernces values */

#define COUNT1  0x0c0001a /*Increment Torque error i+1*/
#define DC_AUX  0x0c0001d /* Last DC*/
#define INC_DC 0x0c0001e /* flc2 set point */
#define EQ 0x0c0001f
/* duty cycle */
#define DCFLC1          0x0c0001b  /*Duty cycle from FLC1*/
#define DCFLC2          0x0c0001c  /*Duty cycle from FLC2*/

#define  f0       0xc00048l
#define  f1       0xc00049l
#define  f2       0xc0004al
#define  f3       0xc0004bl
#define  f4       0xc0004cl
#define  f5       0xc0004dl
#define  f6       0xc0004el
#define  f7       0xc0004fl
#define  f8       0xc00050l
#define  f9       0xc00051l
/*10*/
#define  fa       0xc00052l
#define  fb       0xc00053l
```

```
#define  fc       0xc00054l
#define  fd       0xc00055l
#define  fe       0xc00056l
#define  ff       0xc00057l
#define  f10      0xc00058l
#define  f11      0xc00059l
#define  f12      0xc0005al
#define  f13      0xc0005bl
/*20*/
#define  f14      0xc0005cl
#define  f15      0xc0005dl

#define  PI 3.141592654

#define C_INC_DC  842
#define C_FI2           2150
#define C_FI2bis  2150
#define C_FD2           2150
#define C_FD2bis  2150

/* DSP file  */
#define  FILENAME "TH1_9.OUT"

/* Global variables */

char x;
int flg_err=0,q=0,r=0,j=0,k=0,i=0;
long int t=0;

/* Global adquisition variables */
float v0,v1,v2,v3,v4,v5,v6,v7,v8,v9,va,vb;
float vc,vd,ve,vf,v10,v11,v12,v13,v14,v15;

unsigned long stop=0,flag=0,counter=0,error=0,start=0;



/* set points */

float mfs_set=0,te_set=0,te_hst=0,mfs_hst=0;

/* FLDTC variables */
   unsigned long dc2=3125,dc1=3125;
   int eq,fi,index1,index2,index3,ct,in1,in2,in3,count, count1;
   float Wpc,Tpc,fin1,fin2,fin3,data;
   float dc_aux,inc_dc,duty_ratio=1;
   float wmrds=0;

   FILE *dades;

   /*Array where the read file data is stored*/
   float struct_INC_DC[C_INC_DC],struct_FI2[C_FI2],struct_FD2[C_FD2];
   float struct_FI2bis[C_FI2bis],struct_FD2bis[C_FD2bis];
/* pointer file */

FILE *f_grf;
```

```
/* data matrix */
float dt [5][300];
/*  float dt [5][300];*/
/* Functions definition */
void Read_FLC(void);
void IniciDSP(void);
void Setpoints(void);
void EngegaDSP(void);
void ReadDPRAM(void);
void Savedata(void);
void Find_DC(void);
void AturaDSP(void);
void Data2file(void);

void main (void)
{
clrscr();
printf("\nPress '1' to start\n");
x = getch();
if (x=='1')
      {
      Read_FLC();
      IniciDSP();
      delay(500);
      Setpoints();
      delay (500);
      EngegaDSP();
      delay (500);
      for(t=0; t<500000 ; t++)
            {
            delay(0.01);
            r++;
            Get_DPRAM_Word_32(START,&start);
              if (start == 1 )
                  {
                  Find_DC();
                  start=0;
                  Put_DPRAM_Word_32(START,start);
         /*       for(q=0; q<10 ; q++)
                  {
                  }
                  q=0;*/
               /* Get_DPRAM_Word_32(FLAG,&flag);
                  if (flag == 1 )
                        {          */
                              if ( j <= 299 && r > 20000)
                                    {

                                    ReadDPRAM();
                                    Savedata(); /* Store the data array */
                                    j++;
                                    }
            /*          }
            /*    else
                        {
```

```
                              flg_err++;
                         }     */
                    }
               }
     }
AturaDSP();
printf("\nEND ");
delay(2000);
Get_DPRAM_Word_32(ERROR,&error);
printf("\nThe number of errors is %li",error);
Get_DPRAM_Word_32(COUNTER,&counter);
printf("\nThe number of times is %li",counter);
printf("\nThe number of flag real errors is %i",flg_err);

Data2file();
getche();
}

/*****************************************************************/
/*                 DSP  inizialization                            */
/*****************************************************************/
void IniciDSP(void)
{
/* Seleccio data type for the DSP */
Set_Processor_Data_Type_Size_32();

/* DSP Inicialitzation. Board adress  (0x290), DPRMA adress (0xD00)*/
/*                      bus ISA size  */
Select_Board(0x290, 0xd00, 8);

/* Enable DPRAM  */
Enable_DPRAM();

/* DSP reset  */
Global_Reset();

/* Load the file to be executed  */
Load_Object_File(FILENAME);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End of DSP inizialization                       */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Engega DSP                                      */
/*****************************************************************/
void EngegaDSP(void)
{     stop=0;
      Put_DPRAM_Word_32(STOP, stop);
      /* Run file  */
      Run_From(Get_Entry_PC());
      delay(2000);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*                End Engega DSP                                    */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Setpoints                                      */
/*****************************************************************/
void Setpoints(void)
{   double mfs_sug=0,te_set_d=0;

      printf("\n\nTorque setpoint N/m: ");
      scanf("%f", &te_set);
      Put_DPRAM_Float_32(TE,te_set);
      Tpc=te_set*100/9.8;
      printf("Torque set point hst margin: ");
      scanf("%f", &te_hst);
      Put_DPRAM_Float_32(HST_TE,te_hst);

      te_set_d=te_set;
      /* because squared value */
/*    mfs_sug=(0.3665*sqrt(te_set_d)*0.3665*sqrt(te_set_d));
      printf("\nThe suggested optimized flux square value is %lf \n",
mfs_sug);*/

      mfs_sug=0.3665*sqrt(te_set_d);
      printf("\nThe suggested optimized flux value is %lf \n", mfs_sug);
      printf("\nFlux set point: ");
      scanf("%f", &mfs_set);
/*    mfs_set=1.05;*/
      Put_DPRAM_Float_32(MFS,mfs_set);
      printf("Flux set point hst margin: ");
      scanf("%f", &mfs_hst);
      Put_DPRAM_Float_32(HST_MFS,mfs_hst);

}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End setpoints                                     */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/


/*****************************************************************/
/*                Read DPRAM                                     */
/*****************************************************************/

void ReadDPRAM(void)
{
Get_DPRAM_Float_32(f0,&v0);
Get_DPRAM_Float_32(f1,&v1);
Get_DPRAM_Float_32(f2,&v2);
Get_DPRAM_Float_32(f3,&v3);
Get_DPRAM_Float_32(f4,&v4);
/*Get_DPRAM_Float_32(f5,&v5);
Get_DPRAM_Float_32(f6,&v6);
Get_DPRAM_Float_32(f7,&v7);
/*Get_DPRAM_Float_32(f8,&v8);
Get_DPRAM_Float_32(f9,&v9);
/*10*/
```

```
/*Get_DPRAM_Float_32(fa,&va);
Get_DPRAM_Float_32(fb,&vb);
Get_DPRAM_Float_32(fc,&vc);
Get_DPRAM_Float_32(fd,&vd);
Get_DPRAM_Float_32(fe,&ve);
Get_DPRAM_Float_32(ff,&vf);
Get_DPRAM_Float_32(f10,&v10);
Get_DPRAM_Float_32(f11,&v11);
Get_DPRAM_Float_32(f12,&v12);
Get_DPRAM_Float_32(f13,&v13);  */
/*20*/
/*Get_DPRAM_Float_32(f14,&v14);
Get_DPRAM_Float_32(f15,&v15);*/


}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End read DPRAM                                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                Savedata                                       */
/*****************************************************************/

void Savedata(void)
{
dt[0][j]=v0;
dt[1][j]=v1;
dt[2][j]=v2;
dt[3][j]=v3;
dt[4][j]=v4;
/*dt[5][j]=v5;
dt[6][j]=v6;
dt[7][j]=v7;
/*dt[8][j]=v8;
dt[9][j]=v9;
/*10*/
/*dt[10][j]=va;
dt[11][j]=vb;
dt[12][j]=vc;
dt[13][j]=vd;
dt[14][j]=ve;
dt[15][j]=vf;
dt[16][j]=v10;
dt[17][j]=v11;
dt[18][j]=v12;
dt[19][j]=v13;*/
/*20*/
/*dt[20][j]=v14;
dt[21][j]=v15;*/
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End save data                                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*****************************************************************/
/*                        Find_DC                                    */
```

```
/*****************************************************************/
void Find_DC(void)
{
Get_DPRAM_Word_32(EQ,&eq);

if (eq == 0)
      {
      /* FLC1 */
      Get_DPRAM_Word_32(COUNT1,&count1);
      Get_DPRAM_Word_32(FI,&fi);

/* fi=1 and Tpc >=Wpc 10. Total fi=11*/
/* fi=0 and Tpc >=Wpc 10. Total fi=10*/
/* fi=1 and Tpc < Wpc 00. Total fi=01*/
/* fi=0 and Tpc < Wpc 00. Total fi=00*/

      if (fi == 11)
            duty_ratio=struct_FI2[count1];
      else if (fi == 1)
            duty_ratio=struct_FI2bis[count1];
      else if (fi == 10)
            duty_ratio=struct_FD2[count1];
      else
            duty_ratio=struct_FD2bis[count1];

/*    if (duty_ratio < 0.45 )
            duty_ratio=0.45;
      if (duty_ratio > 0.55)
            duty_ratio=0.55;*/

      /* 83us=1041,666667ct 125us=1562.5ct 166us=2083.33ct 250us=3125ct */
      dc1=duty_ratio*2083;
      if (dc1<1850)
            dc1=1850;
      Put_DPRAM_Word_32(DCFLC1,dc1);
      }
else
      {
      /* FLC2 */
      Get_DPRAM_Word_32(COUNT1,&count1);
      Get_DPRAM_Float_32(DC_AUX,&dc_aux);
      inc_dc=struct_INC_DC[count1];
      duty_ratio=inc_dc+dc_aux;
/*    if (duty_ratio < 0.2 )
            duty_ratio=0.2;
      if (duty_ratio > 0.85)
            duty_ratio=0.85;*/
      dc2=duty_ratio*2083;
      if (dc2<1850)
            dc2=1850;

      Put_DPRAM_Word_32(DCFLC2,dc2);
/*    Put_DPRAM_Float_32(INC_DC,inc_dc);   */
      }
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

```
/*                End Find_DC                                          */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

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

/*****************************************************************/
/*                         Data2file                              */
/*****************************************************************/
void Data2file(void)
{
if ((f_grf = fopen("\\FLDTC11.GRF", "w")) == 0)
      {
      printf("Unable to open file");
      return;
      }
else
      {
      for(k=0;k<j;k++)
            {
/*          fprintf(f_grf, "%e %e %e %e %e %e %e
%e\n",dt[0][k],dt[1][k],dt[2][k],dt[3][k],dt[4][k],dt[5][k],dt[6][k],dt[7][
k]);*/
            fprintf(f_grf, " %e %e %e %e
%e\n",dt[0][k],dt[1][k],dt[2][k],dt[3][k],dt[4][k]);
            }
      fclose(f_grf);
}
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                End Data2file */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

/*****************************************************************/
/*                            Read_FLC                            */
/*****************************************************************/
void Read_FLC(void)
{  clrscr();
   count=0;
   count1=0;
   data=0;
   /*open the file (it must be in the root */
   dades=fopen("INC_DC.txt","rt");

   /*while not file end*/
   while (!feof(dades))
   {
      fscanf(dades,"%f",&data);      /* read file float */
```

```
      struct_INC_DC[count]=data;    /* save it to the structure */
        /*printf("\nThe input %d of the file, is %f",count,data);  */
      count++;
   }
   fclose(dades);                  /*the file is closed*/

   printf("\nThe file INC_DC has %d inputs",count);
/* The data from the structures is shown */
/*   while (count1<count)
   {
      printf("\nThe data input %d of the struct is
%f",count1,estructura[count1]);
      count1++;
   } */

   /* Second file*/

   count=0;
   count1=0;
   data=0;
   /*open the file (it must be in the root */
   dades=fopen("FD2.txt","rt");

   /*while not file end*/
   while (!feof(dades))
   {
      fscanf(dades,"%f",&data);      /* read file float */
      struct_FD2[count]=data;    /* save it to the structure */
        /*printf("\nThe input %d of the file, is %f",count,data);  */
      count++;
   }
   fclose(dades);                  /*the file is closed*/

   printf("\nThe file FD2 has %d inputs",count);

   /* Third file*/

   count=0;
   count1=0;
   data=0;
   /*open the file (it must be in the root */
   dades=fopen("FI2.txt","rt");

   /*while not file end*/
   while (!feof(dades))
   {
      fscanf(dades,"%f",&data);      /* read file float */
      struct_FI2[count]=data;    /* save it to the structure */
        /*printf("\nThe input %d of the file, is %f",count,data);  */
      count++;
   }
   fclose(dades);                  /*the file is closed*/

   printf("\nThe file FI2 has %d inputs",count);

   /* Fourth file*/
```

```
   count=0;
   count1=0;
   data=0;
   /*open the file (it must be in the root */
   dades=fopen("FD2bis.txt","rt");

   /*while not file end*/
   while (!feof(dades))
   {
      fscanf(dades,"%f",&data);      /* read file float */
      struct_FD2bis[count]=data;    /* save it to the structure */
        /*printf("\nThe input %d of the file, is %f",count,data);  */
      count++;
   }
   fclose(dades);                  /*the file is closed*/

   printf("\nThe file FD2bis has %d inputs",count);

   /* Fifth file*/

   count=0;
   count1=0;
   data=0;
   /*open the file (it must be in the root */
   dades=fopen("FI2bis.txt","rt");

   /*while not file end*/
   while (!feof(dades))
   {
      fscanf(dades,"%f",&data);      /* read file float */
      struct_FI2bis[count]=data;    /* save it to the structure */
        /*printf("\nThe input %d of the file, is %f",count,data);  */
      count++;
   }
   fclose(dades);                  /*the file is closed*/
   printf("\nThe file FI2bis has %d inputs",count);
}
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
/*                              End Read_FLC                                  */
/*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/
```

A.2 - Fuzzy rules.

A.2.1 - FLC 1.

The Fuzzy Logic controller 1 system comprises four groups of rules, each of which contains 46 rules (see section 3.2.1). Two of them are used when the stator flux is smaller than its reference value (Flux increase) and the other two in the opposite case (Flux decrease). In any case just one fuzzy system is used per iteration, and it depends on the working point.

All four groups of 46 rules are listed below.

A.2.1.1 - Flux increase, Wpc<Tpc.

1. If (t-e is n-l) and (s-f-p is s) and (T+2w is s) then (dc is vl) (1)
2. If (t-e is n-l) and (s-f-p is m) and (T+2w is s) then (dc is vl) (1)
3. If (t-e is n-l) and (s-f-p is l) and (T+2w is s) then (dc is m) (1)
4. If (t-e is n-m) and (s-f-p is s) and (T+2w is s) then (dc is s) (1)
5. If (t-e is n-m) and (s-f-p is m) and (T+2w is s) then (dc is s) (1)
6. If (t-e is n-m) and (s-f-p is l) and (T+2w is s) then (dc is s) (1)
7. If (t-e is n-s) and (s-f-p is s) and (T+2w is s) then (dc is z) (1)
8. If (t-e is n-s) and (s-f-p is m) and (T+2w is s) then (dc is z) (1)
9. If (t-e is n-s) and (s-f-p is l) and (T+2w is s) then (dc is z) (1)
10. If (t-e is s) and (s-f-p is s) and (T+2w is s) then (dc is s) (1)
11. If (t-e is s) and (s-f-p is m) and (T+2w is s) then (dc is s) (1)
12. If (t-e is s) and (s-f-p is l) and (T+2w is s) then (dc is s) (1)
13. If (t-e is m) and (s-f-p is s) and (T+2w is s) then (dc is s) (1)
14. If (t-e is m) and (s-f-p is m) and (T+2w is s) then (dc is m) (1)
15. If (t-e is m) and (s-f-p is l) and (T+2w is s) then (dc is m) (1)
16. If (t-e is l) then (dc is vl) (1)
17. If (t-e is n-l) and (s-f-p is s) and (T+2w is m) then (dc is vl) (1)
18. If (t-e is n-l) and (s-f-p is m) and (T+2w is m) then (dc is vl) (1)
19. If (t-e is n-l) and (s-f-p is l) and (T+2w is m) then (dc is l) (1)
20. If (t-e is n-m) and (s-f-p is s) and (T+2w is m) then (dc is s) (1)
21. If (t-e is n-m) and (s-f-p is m) and (T+2w is m) then (dc is s) (1)
22. If (t-e is n-m) and (s-f-p is l) and (T+2w is m) then (dc is s) (1)
23. If (t-e is n-s) and (s-f-p is s) and (T+2w is m) then (dc is m) (1)
24. If (t-e is n-s) and (s-f-p is m) and (T+2w is m) then (dc is m) (1)
25. If (t-e is n-s) and (s-f-p is l) and (T+2w is m) then (dc is m) (1)
26. If (t-e is s) and (s-f-p is s) and (T+2w is m) then (dc is m) (1)
27. If (t-e is s) and (s-f-p is m) and (T+2w is m) then (dc is m) (1)
28. If (t-e is s) and (s-f-p is l) and (T+2w is m) then (dc is l) (1)
29. If (t-e is m) and (s-f-p is s) and (T+2w is m) then (dc is m) (1)
30. If (t-e is m) and (s-f-p is m) and (T+2w is m) then (dc is l) (1)
31. If (t-e is m) and (s-f-p is l) and (T+2w is m) then (dc is l) (1)
32. If (t-e is n-l) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
33. If (t-e is n-l) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
34. If (t-e is n-l) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
35. If (t-e is n-m) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
36. If (t-e is n-m) and (s-f-p is m) and (T+2w is l) then (dc is z) (1)
37. If (t-e is n-m) and (s-f-p is l) and (T+2w is l) then (dc is z) (1)
38. If (t-e is n-s) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
39. If (t-e is n-s) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
40. If (t-e is n-s) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
41. If (t-e is s) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
42. If (t-e is s) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
43. If (t-e is s) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
44. If (t-e is m) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
45. If (t-e is m) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
46. If (t-e is m) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)

A.2.1.2 - Flux increase, Wpc>Tpc.

1. If (t-e is n-l) and (s-f-p is s) and (2T+w is s) then (dc is vl) (1)
2. If (t-e is n-l) and (s-f-p is m) and (2T+w is s) then (dc is vl) (1)
3. If (t-e is n-l) and (s-f-p is l) and (2T+w is s) then (dc is m) (1)
4. If (t-e is n-m) and (s-f-p is s) and (2T+w is s) then (dc is s) (1)
5. If (t-e is n-m) and (s-f-p is m) and (2T+w is s) then (dc is s) (1)
6. If (t-e is n-m) and (s-f-p is l) and (2T+w is s) then (dc is s) (1)
7. If (t-e is n-s) and (s-f-p is s) and (2T+w is s) then (dc is z) (1)
8. If (t-e is n-s) and (s-f-p is m) and (2T+w is s) then (dc is z) (1)
9. If (t-e is n-s) and (s-f-p is l) and (2T+w is s) then (dc is z) (1)
10. If (t-e is s) and (s-f-p is s) and (2T+w is s) then (dc is s) (1)
11. If (t-e is s) and (s-f-p is m) and (2T+w is s) then (dc is s) (1)
12. If (t-e is s) and (s-f-p is l) and (2T+w is s) then (dc is s) (1)
13. If (t-e is m) and (s-f-p is s) and (2T+w is s) then (dc is s) (1)
14. If (t-e is m) and (s-f-p is m) and (2T+w is s) then (dc is m) (1)
15. If (t-e is m) and (s-f-p is l) and (2T+w is s) then (dc is m) (1)
16. If (t-e is l) then (dc is vl) (1)
17. If (t-e is n-l) and (s-f-p is s) and (2T+w is m) then (dc is vl) (1)
18. If (t-e is n-l) and (s-f-p is m) and (2T+w is m) then (dc is vl) (1)
19. If (t-e is n-l) and (s-f-p is l) and (2T+w is m) then (dc is l) (1)
20. If (t-e is n-m) and (s-f-p is s) and (2T+w is m) then (dc is s) (1)
21. If (t-e is n-m) and (s-f-p is m) and (2T+w is m) then (dc is s) (1)
22. If (t-e is n-m) and (s-f-p is l) and (2T+w is m) then (dc is s) (1)
23. If (t-e is n-s) and (s-f-p is s) and (2T+w is m) then (dc is m) (1)
24. If (t-e is n-s) and (s-f-p is m) and (2T+w is m) then (dc is m) (1)
25. If (t-e is n-s) and (s-f-p is l) and (2T+w is m) then (dc is m) (1)
26. If (t-e is s) and (s-f-p is s) and (2T+w is m) then (dc is m) (1)
27. If (t-e is s) and (s-f-p is m) and (2T+w is m) then (dc is m) (1)
28. If (t-e is s) and (s-f-p is l) and (2T+w is m) then (dc is l) (1)
29. If (t-e is m) and (s-f-p is s) and (2T+w is m) then (dc is m) (1)
30. If (t-e is m) and (s-f-p is m) and (2T+w is m) then (dc is l) (1)
31. If (t-e is m) and (s-f-p is l) and (2T+w is m) then (dc is l) (1)
32. If (t-e is n-l) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
33. If (t-e is n-l) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
34. If (t-e is n-l) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
35. If (t-e is n-m) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
36. If (t-e is n-m) and (s-f-p is m) and (2T+w is l) then (dc is z) (1)
37. If (t-e is n-m) and (s-f-p is l) and (2T+w is l) then (dc is z) (1)
38. If (t-e is n-s) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
39. If (t-e is n-s) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
40. If (t-e is n-s) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
41. If (t-e is s) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
42. If (t-e is s) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
43. If (t-e is s) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
44. If (t-e is m) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
45. If (t-e is m) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
46. If (t-e is m) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)

A.2.1.3 - Flux decrease, Wpc<Tpc.

1. If (t-e is n-l) and (s-f-p is s) and (T+2w is s) then (dc is m) (1)
2. If (t-e is n-l) and (s-f-p is m) and (T+2w is s) then (dc is vl) (1)
3. If (t-e is n-l) and (s-f-p is l) and (T+2w is s) then (dc is vl) (1)
4. If (t-e is n-m) and (s-f-p is s) and (T+2w is s) then (dc is z) (1)
5. If (t-e is n-m) and (s-f-p is m) and (T+2w is s) then (dc is z) (1)
6. If (t-e is n-m) and (s-f-p is l) and (T+2w is s) then (dc is s) (1)
7. If (t-e is n-s) and (s-f-p is s) and (T+2w is s) then (dc is z) (1)
8. If (t-e is n-s) and (s-f-p is m) and (T+2w is s) then (dc is z) (1)
9. If (t-e is n-s) and (s-f-p is l) and (T+2w is s) then (dc is z) (1)
10. If (t-e is s) and (s-f-p is s) and (T+2w is s) then (dc is s) (1)
11. If (t-e is s) and (s-f-p is m) and (T+2w is s) then (dc is s) (1)
12. If (t-e is s) and (s-f-p is l) and (T+2w is s) then (dc is s) (1)
13. If (t-e is m) and (s-f-p is s) and (T+2w is s) then (dc is m) (1)
14. If (t-e is m) and (s-f-p is m) and (T+2w is s) then (dc is m) (1)
15. If (t-e is m) and (s-f-p is l) and (T+2w is s) then (dc is s) (1)
16. If (t-e is l) then (dc is vl) (1)
17. If (t-e is n-l) and (s-f-p is s) and (T+2w is m) then (dc is l) (1)
18. If (t-e is n-l) and (s-f-p is m) and (T+2w is m) then (dc is vl) (1)
19. If (t-e is n-l) and (s-f-p is l) and (T+2w is m) then (dc is vl) (1)
20. If (t-e is n-m) and (s-f-p is s) and (T+2w is m) then (dc is z) (1)
21. If (t-e is n-m) and (s-f-p is m) and (T+2w is m) then (dc is z) (1)
22. If (t-e is n-m) and (s-f-p is l) and (T+2w is m) then (dc is z) (1)
23. If (t-e is n-s) and (s-f-p is s) and (T+2w is m) then (dc is m) (1)
24. If (t-e is n-s) and (s-f-p is m) and (T+2w is m) then (dc is m) (1)
25. If (t-e is n-s) and (s-f-p is l) and (T+2w is m) then (dc is m) (1)
26. If (t-e is s) and (s-f-p is s) and (T+2w is m) then (dc is vl) (1)
27. If (t-e is s) and (s-f-p is m) and (T+2w is m) then (dc is m) (1)
28. If (t-e is s) and (s-f-p is l) and (T+2w is m) then (dc is l) (1)
29. If (t-e is m) and (s-f-p is s) and (T+2w is m) then (dc is vl) (1)
30. If (t-e is m) and (s-f-p is m) and (T+2w is m) then (dc is l) (1)
31. If (t-e is m) and (s-f-p is l) and (T+2w is m) then (dc is m) (1)
32. If (t-e is n-l) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
33. If (t-e is n-l) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
34. If (t-e is n-l) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
35. If (t-e is n-m) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
36. If (t-e is n-m) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
37. If (t-e is n-m) and (s-f-p is l) and (T+2w is l) then (dc is z) (1)
38. If (t-e is n-s) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
39. If (t-e is n-s) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
40. If (t-e is n-s) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
41. If (t-e is s) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
42. If (t-e is s) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
43. If (t-e is s) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)
44. If (t-e is m) and (s-f-p is s) and (T+2w is l) then (dc is vl) (1)
45. If (t-e is m) and (s-f-p is m) and (T+2w is l) then (dc is vl) (1)
46. If (t-e is m) and (s-f-p is l) and (T+2w is l) then (dc is vl) (1)

A.2.1.4 - Flux decrease, Wpc>Tpc.

1. If (t-e is n-l) and (s-f-p is s) and (2T+w is s) then (dc is m) (1)
2. If (t-e is n-l) and (s-f-p is m) and (2T+w is s) then (dc is vl) (1)
3. If (t-e is n-l) and (s-f-p is l) and (2T+w is s) then (dc is vl) (1)
4. If (t-e is n-m) and (s-f-p is s) and (2T+w is s) then (dc is z) (1)
5. If (t-e is n-m) and (s-f-p is m) and (2T+w is s) then (dc is z) (1)
6. If (t-e is n-m) and (s-f-p is l) and (2T+w is s) then (dc is s) (1)
7. If (t-e is n-s) and (s-f-p is s) and (2T+w is s) then (dc is z) (1)
8. If (t-e is n-s) and (s-f-p is m) and (2T+w is s) then (dc is z) (1)
9. If (t-e is n-s) and (s-f-p is l) and (2T+w is s) then (dc is z) (1)
10. If (t-e is s) and (s-f-p is s) and (2T+w is s) then (dc is s) (1)
11. If (t-e is s) and (s-f-p is m) and (2T+w is s) then (dc is s) (1)
12. If (t-e is s) and (s-f-p is l) and (2T+w is s) then (dc is s) (1)
13. If (t-e is m) and (s-f-p is s) and (2T+w is s) then (dc is m) (1)
14. If (t-e is m) and (s-f-p is m) and (2T+w is s) then (dc is m) (1)
15. If (t-e is m) and (s-f-p is l) and (2T+w is s) then (dc is s) (1)
16. If (t-e is l) then (dc is vl) (1)
17. If (t-e is n-l) and (s-f-p is s) and (2T+w is m) then (dc is l) (1)
18. If (t-e is n-l) and (s-f-p is m) and (2T+w is m) then (dc is vl) (1)
19. If (t-e is n-l) and (s-f-p is l) and (2T+w is m) then (dc is vl) (1)
20. If (t-e is n-m) and (s-f-p is s) and (2T+w is m) then (dc is z) (1)
21. If (t-e is n-m) and (s-f-p is m) and (2T+w is m) then (dc is z) (1)
22. If (t-e is n-m) and (s-f-p is l) and (2T+w is m) then (dc is z) (1)
23. If (t-e is n-s) and (s-f-p is s) and (2T+w is m) then (dc is m) (1)
24. If (t-e is n-s) and (s-f-p is m) and (2T+w is m) then (dc is m) (1)
25. If (t-e is n-s) and (s-f-p is l) and (2T+w is m) then (dc is m) (1)
26. If (t-e is s) and (s-f-p is s) and (2T+w is m) then (dc is vl) (1)
27. If (t-e is s) and (s-f-p is m) and (2T+w is m) then (dc is m) (1)
28. If (t-e is s) and (s-f-p is l) and (2T+w is m) then (dc is l) (1)
29. If (t-e is m) and (s-f-p is s) and (2T+w is m) then (dc is vl) (1)
30. If (t-e is m) and (s-f-p is m) and (2T+w is m) then (dc is l) (1)
31. If (t-e is m) and (s-f-p is l) and (2T+w is m) then (dc is m) (1)
32. If (t-e is n-l) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
33. If (t-e is n-l) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
34. If (t-e is n-l) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
35. If (t-e is n-m) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
36. If (t-e is n-m) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
37. If (t-e is n-m) and (s-f-p is l) and (2T+w is l) then (dc is z) (1)
38. If (t-e is n-s) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
39. If (t-e is n-s) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
40. If (t-e is n-s) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
41. If (t-e is s) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
42. If (t-e is s) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
43. If (t-e is s) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)
44. If (t-e is m) and (s-f-p is s) and (2T+w is l) then (dc is vl) (1)
45. If (t-e is m) and (s-f-p is m) and (2T+w is l) then (dc is vl) (1)
46. If (t-e is m) and (s-f-p is l) and (2T+w is l) then (dc is vl) (1)

A.2.2 - FLC 2.

The fuzzy logic controller 2 system contains 37 rules (see section 3.2.2).

All 37 rules are listed below.

1. If (inc-Te is n-l) then (inc-dc is vl) (1)
2. If (inc-Te is n-m) and (E-Te-ant is n-l) then (inc-dc is s) (1)
3. If (inc-Te is n-m) and (E-Te-ant is n-m) then (inc-dc is m) (1)
4. If (inc-Te is n-m) and (E-Te-ant is n-s) then (inc-dc is l) (1)
5. If (inc-Te is n-m) and (E-Te-ant is s) then (inc-dc is l) (1)
6. If (inc-Te is n-m) and (E-Te-ant is m) then (inc-dc is l) (1)
7. If (inc-Te is n-m) and (E-Te-ant is l) then (inc-dc is vl) (1)
8. If (inc-Te is n-s) and (E-Te-ant is n-l) then (inc-dc is n-s) (1)
9. If (inc-Te is n-s) and (E-Te-ant is n-m) then (inc-dc is n-s) (1)
10. If (inc-Te is n-s) and (E-Te-ant is n-s) then (inc-dc is z) (1)
11. If (inc-Te is n-s) and (E-Te-ant is s) then (inc-dc is s) (1)
12. If (inc-Te is n-s) and (E-Te-ant is m) then (inc-dc is m) (1)
13. If (inc-Te is n-s) and (E-Te-ant is l) then (inc-dc is l) (1)
14. If (inc-Te is z) and (E-Te-ant is n-l) then (inc-dc is n-m) (1)
15. If (inc-Te is z) and (E-Te-ant is n-m) then (inc-dc is n-m) (1)
16. If (inc-Te is z) and (E-Te-ant is n-s) then (inc-dc is n-s) (1)
17. If (inc-Te is z) and (E-Te-ant is s) then (inc-dc is z) (1)
18. If (inc-Te is z) and (E-Te-ant is m) then (inc-dc is s) (1)
19. If (inc-Te is z) and (E-Te-ant is l) then (inc-dc is m) (1)
20. If (inc-Te is s) and (E-Te-ant is n-l) then (inc-dc is n-l) (1)
21. If (inc-Te is s) and (E-Te-ant is n-m) then (inc-dc is n-m) (1)
22. If (inc-Te is s) and (E-Te-ant is n-s) then (inc-dc is n-s) (1)
23. If (inc-Te is s) and (E-Te-ant is s) then (inc-dc is n-s) (1)
24. If (inc-Te is s) and (E-Te-ant is m) then (inc-dc is z) (1)
25. If (inc-Te is s) and (E-Te-ant is l) then (inc-dc is s) (1)
26. If (inc-Te is m) and (E-Te-ant is n-l) then (inc-dc is n-vl) (1)
27. If (inc-Te is m) and (E-Te-ant is n-m) then (inc-dc is n-l) (1)
28. If (inc-Te is m) and (E-Te-ant is n-s) then (inc-dc is n-m) (1)
29. If (inc-Te is m) and (E-Te-ant is s) then (inc-dc is n-m) (1)
30. If (inc-Te is m) and (E-Te-ant is m) then (inc-dc is n-s) (1)
31. If (inc-Te is m) and (E-Te-ant is l) then (inc-dc is n-s) (1)
32. If (inc-Te is l) and (E-Te-ant is n-l) then (inc-dc is n-vl) (1)
33. If (inc-Te is l) and (E-Te-ant is n-m) then (inc-dc is n-vl) (1)
34. If (inc-Te is l) and (E-Te-ant is n-s) then (inc-dc is n-vl) (1)
35. If (inc-Te is l) and (E-Te-ant is s) then (inc-dc is n-vl) (1)
36. If (inc-Te is l) and (E-Te-ant is m) then (inc-dc is n-l) (1)
37. If (inc-Te is l) and (E-Te-ant is l) then (inc-dc is z) (1)

A.3 - Experimental motor drive pictures.

A.3.1 - Workbench. Induction motor_1.5kW. DC motor.

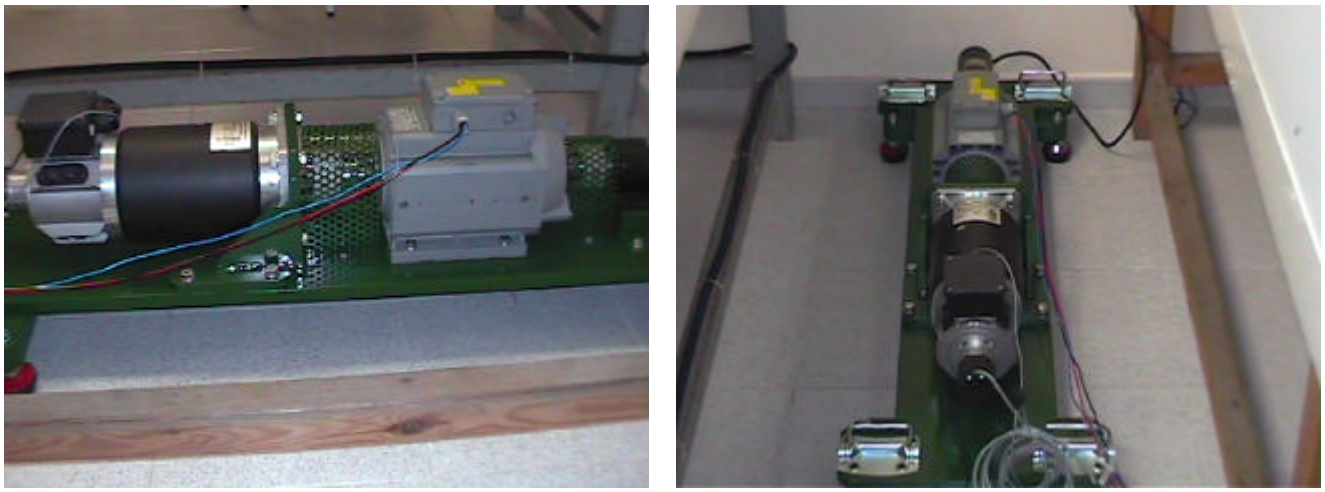

Figure A.1. It is shown the entire workbench with the encoder, DC motor, AC motor_1.5kW and the dynamometer.

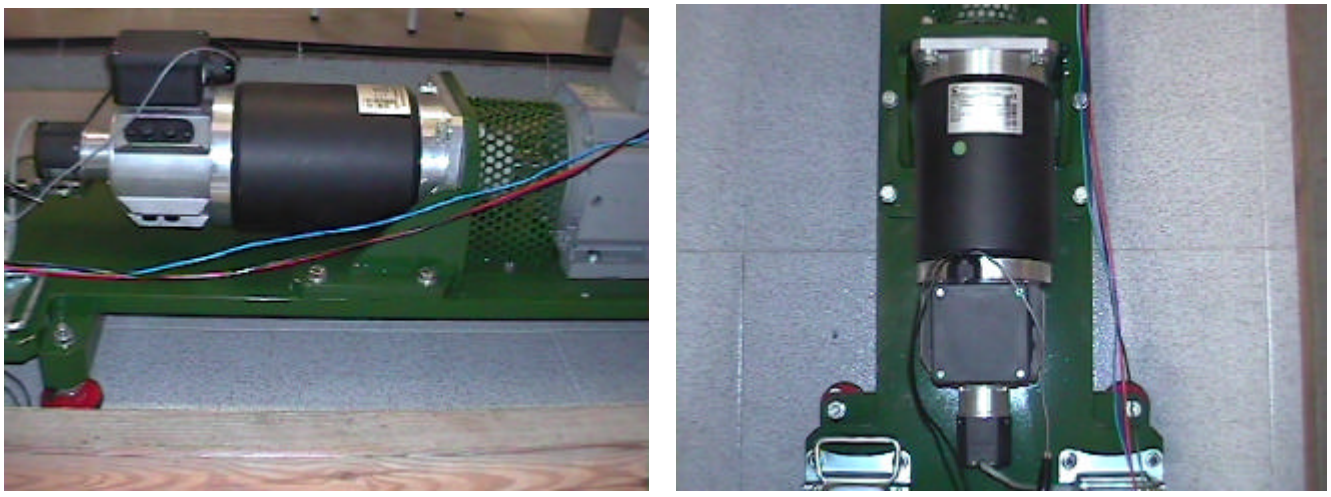

Figure A.2. Encoder and DC motor detail. It should be remembered that the DC motor works as a generator, i.e. as a load.

(Next page shows figures A.3 and A.4).

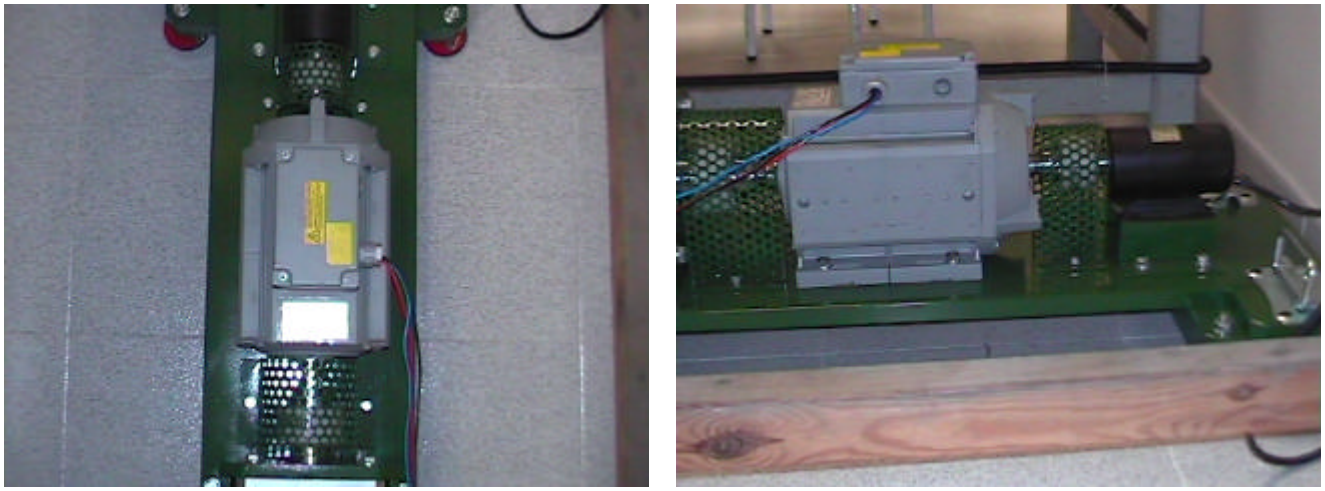

Figure A.3. AC motor_1.5kW and dynamo detail.

A.3.2 - Resistors.

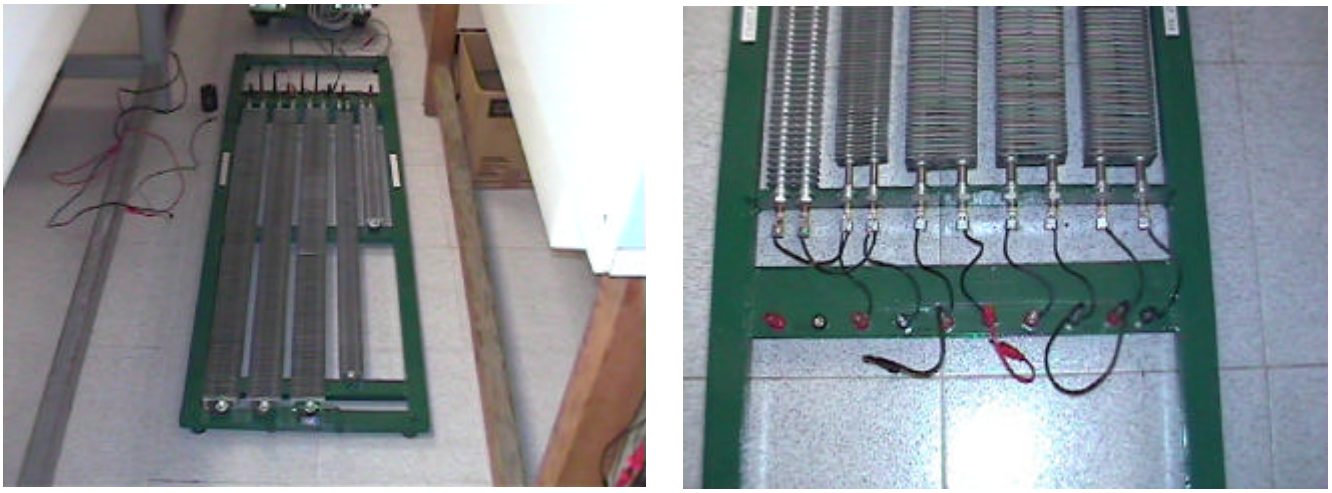

Figure A.4. Resistors used as a load of the DC generator.

(Next page shows figures A.5 and A.6).

A.3.3 - Voltage Source Inverter.

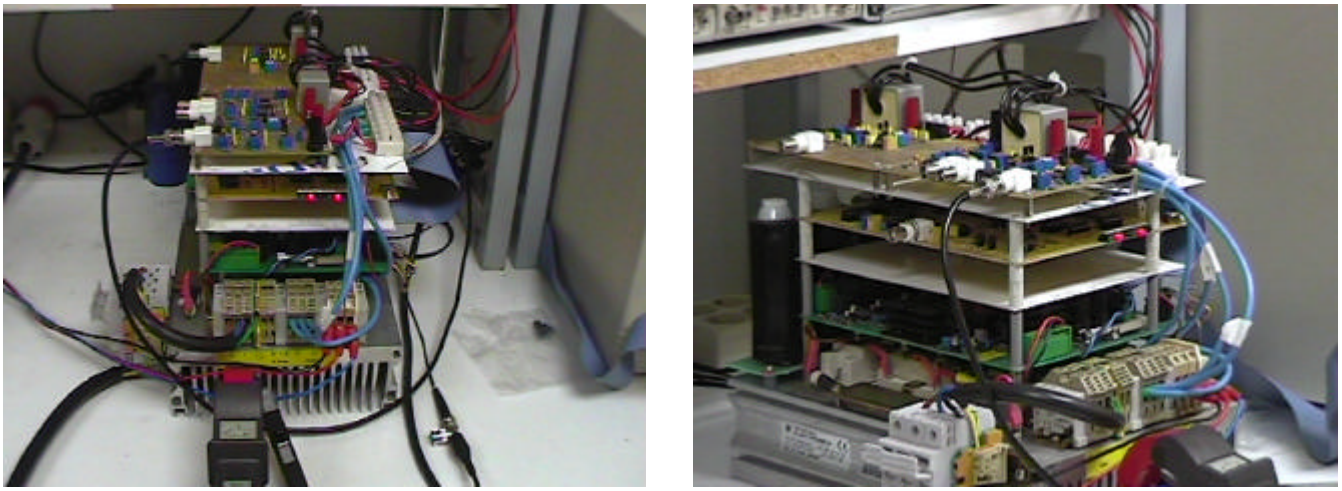

Figure A.5. It is shown the Voltage Source Inverter with its protections board and signal adaptation board.

A.3.4 - Workstation.

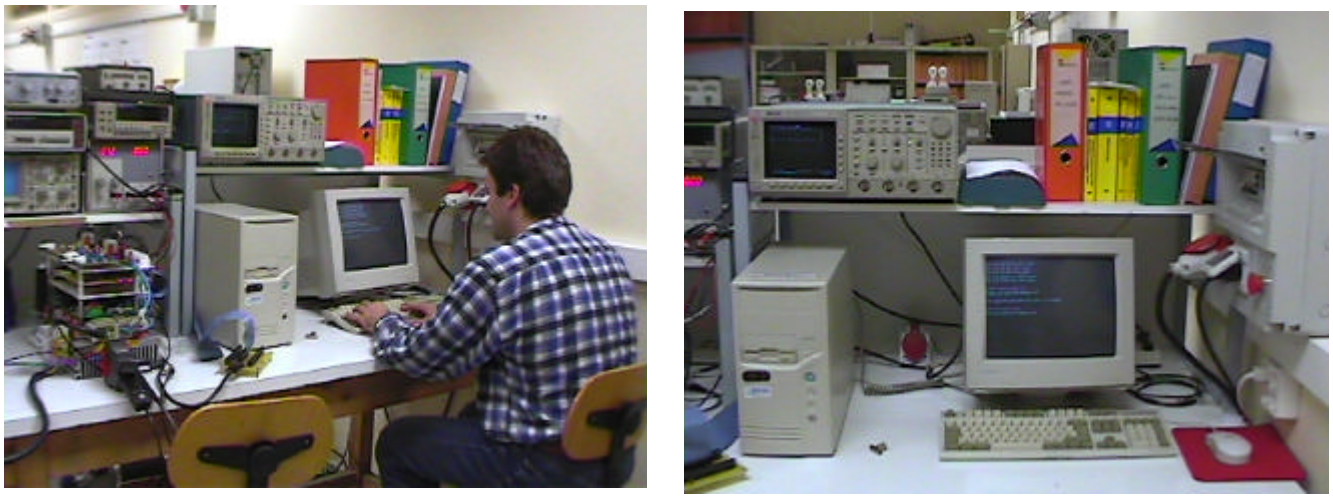

Figure A.6. It is shown the entire workstation.