ภาคผนวก ข
โปรแกรมควบคุมตัวประมวลผลสัญญาณดิจิตอล

โปรแกรมควบคุมตัวประมวลผลสัญญาณดิจิตอล

```
MODULE/RAM/SEG=USER_PM1/ABS=0x30     Main_Program;


{*******************************************************************************
********
* Include General System Parameters and Libraries                                   *
********************************************************************************
*******}
#include <main.h>;
#include <aux_dac.h>;
#include <pwmf32x.h>;
#include <trigono.h>;
#include <Mathfun.h>;
#include <pi.h>;
{*******************************************************************************
********
* Constants Defined in the Module                                                   *
********************************************************************************
*******}
.CONST Delta              =0x148;            {Angle increment 64 pr.rev}
.CONST TwoPiOverThree       =0xffff/3;                  {Hex equivalent of 2pi/3  }
{*******************************************************************************
********
* Local Variable Defind in this Module                                      *
********************************************************************************
********}
#define PI_SF16 0
.VAR/RAM/PM/CIRC/SEG=USER_PM1 PI_Coef16[3];
.INIT PI_Coef16: 0xE00200,0x200100,0x7FFF00;
.VAR/RAM/DM/CIRC/SEG=USER_DM PI_Delay16[2];
```

```
.INIT PI_Delay16: 0x0000,0x0000;


.VAR/DM/RAM/SEG=USER_DM AD_IN;                {volt/herzt command (0-1)}
.INIT AD_IN : 0x6000;                         {Corresponds to 1.0    }


.VAR/DM/RAM/SEG=USER_DM SETUPTIME;
.INIT SETUPTIME :0x0000;
.VAR/DM/RAM/SEG=USER_DM PIODATA1_REF;
.INIT PIODATA1_REF :0x0000;
.VAR/DM/RAM/SEG=USER_DM delt;
.INIT delt :0x0000;


.VAR/DM/RAM/SEG=USER_DM Theta;                          {Current angle       }
.INIT Theta : 0x0000;
.VAR/DM/RAM/SEG=USER_DM soft;                 {Soft start        }
.INIT Soft : 0x0000;
.VAR/DM/RAM/SEG=USER_DM VrefA;                {Voltage demand                }
.VAR/DM/RAM/SEG=USER_DM VrefB;
.VAR/DM/RAM/SEG=USER_DM VrefC;
.INIT VrefA : 0x0000;
.INIT VrefB : 0x0000;
.INIT VrefC : 0x0000;
.VAR/DM/RAM/SEG=USER_DM v;
.INIT v : 0x0;
.VAR/DM/RAM/SEG=USER_DM vref;
.INIT vref :0x000;
.VAR/DM/RAM/SEG=USER_DM vmod;
.INIT vmod : 0x000;
.VAR/DM/RAM/SEG=USER_DM Soft_start;
.INIT Soft_start : 0x0;
.VAR/DM/RAM/SEG=USER_DM Store;
```

```
.INIT Store : 0x0;
.VAR/DM/RAM/SEG=USER_DM v_boost;
.INIT v_boost : 0x000;
.VAR/DM/RAM/SEG=USER_DM SinCos[2];          {Sine Cosine Buffer     }
.VAR/DM/RAM/SEG=USER_DM Vabc[3];            {3 Phase Buffer         }

{****************************************************************************
*************}
{ Start of program code                                                     }
{****************************************************************************
*************}
Startup:
        write_auxdac_init;
        ax1=80{255};
        dm(auxTM0)=ax1;
        dm(auxTM1)=ax1;

        PWM_INIT(PWMSYNC_ISR,PWMTRIP_ISR);

        {ICNTL=0x03;}
        IFC = 0x80;                 { Clear any pending IRQ2 inter.}
        ay0 = 0x0200;                       { unmask irq2 interrupts.      }
        ar = IMASK;
        ar = ar or ay0;
        IMASK = ar;                         { IRQ2 ints fully enabled here}
init_pi16(pi_delay16,0x0000);

Main:                                       { Wait for interrupt to occur   }
        jump Main;

        rts;
```

```
{*********************************************************************************
*************}
{ PWM Interrupt Service Routine
                                   }
{*********************************************************************************
*************}

PWMSYNC_ISR:

{****  BOOST  ****}

        set_DAG_registers_for_trigonometric;
ay0=dm(soft_start);ax0=0x7F00;
                ar=ax0-ay0;if LT af=pass ax0;
                ar=ax0-ay0;if GT af=pass ay0;
                ar=pass af;
dm(store)=ar;my1=dm(store);

        ax0=0x2497;{0x2140}ay0=0x1D00;ar=ax0-ay0;
        dm(vref)=ar;

        ar=dm(adcaux);ay0=0x1D00;ar=ar-ay0;if LE ar=pass 0;
        dm(v)=ar;

        ax0=dm(vref);ay0=dm(v);
        ena AR_SAT;ar=ax0-ay0;if LE ar=pass 0;dis AR_SAT;
        Pi16(PI_Delay16, PI_Coef16, PI_SF16);
        ar=sr1;
        my0=27000;mr=ar*my0(ss);
        sr=lshift mr1 by 8(hi);
        dm(vmod)=ar;


mr=ar*my1(ss);
        ay0=mr1;ax0=30{120};
                ar=ax0-ay0;if LT af=pass ax0;
                ar=ax0-ay0;if GT af=pass ay0;
                ar=pass af;


        dm(v_boost)=ar;
        write_auxpwm(auxch0,v_boost);
        write_auxpwm(auxch1,v_boost);


ar=dm(soft_start);ay1=0x7F00;ar=ar-ay1;if GE jump pas;
ar=dm(soft_start);ar=ar+1;
dm(soft_start)=ar;jump pas1;
pas:
ax0=0x7F01;
dm(soft_start)=ax0;
pas1:
nop;


{****inverter****}
                        my0=0x7FFF{dm(AD_IN)};


                        mr=0;              {Clear mr          }
                        mr1=dm(Theta);           {Preload Theta       }
                        mx0={dm(delt)}Delta;
                        mr=mr+mx0*my0(ss); {Compute new angle & store   }
                        dm(Theta)=mr1;


                        Sin(mr1);          {Result in the ar register   }
```

```
mr=ar*my0 (ss);             {Multiply by AD_IN for VrefA  }
{mr=mr1*my1(ss);}
dm(VrefA)=mr1;

ax1=dm(Theta);              {compute angle for phase B    }
{mx0=TwoPiOverThree;my0=0x4000;mr=mx0*my0(SS);}
ay1=TwoPiOverThree;
ar=ax1-ay1;
Sin(ar);             {Result in the ar register      }
mr=ar*my0 (ss);             {Multiply by AD_IN for VrefB  }
{mr=mr1*my1(ss);}
dm(VrefB)=mr1;

ax1=dm(Theta);              {Compute angle for phase C    }
ay1=TwoPiOverThree;
ar=ax1+ay1;
Sin(ar);             {Result in the ar register      }
mr=ar*my0 (ss);             {Multiply by AD_IN for VrefC  }
{mr=mr1*my1(ss);}
dm(VrefC)=mr1;



ax0=dm(VrefA);
ax1=dm(VrefB);
ay0=dm(VrefC);

PWM_update_demanded_Voltage(ax0,ax1,ay0);




nop;
```

```
        rti;

{******************************************************************************
*************}
{ PWM Trip Interrupt Service Routine
                                   }
{******************************************************************************
*************}

PWMTRIP_ISR:
        nop;
        rti;

.ENDMOD;
```